


3일차


Web and Http

1. Web

- a. 웹페이지는 object(HTML file, JPEG image, Java applet, audio file 등) 으로 이루어진다.
- b. HTML file을 기본으로 한다.
- c. URL 형태를 지님.

`www.someschool.edu/someDept/pic.gif`


host name


path name

2. HTTP (Hypertext transfer protocol)

a. client-server model

- i. client : client는 HTTP protocol 을 통해서 request 를 보내고 response 를 받음.
- ii. server : web server는 HTTP protocol 을 이용하여 request에 대한 response를 object 안에 넣어 보냄

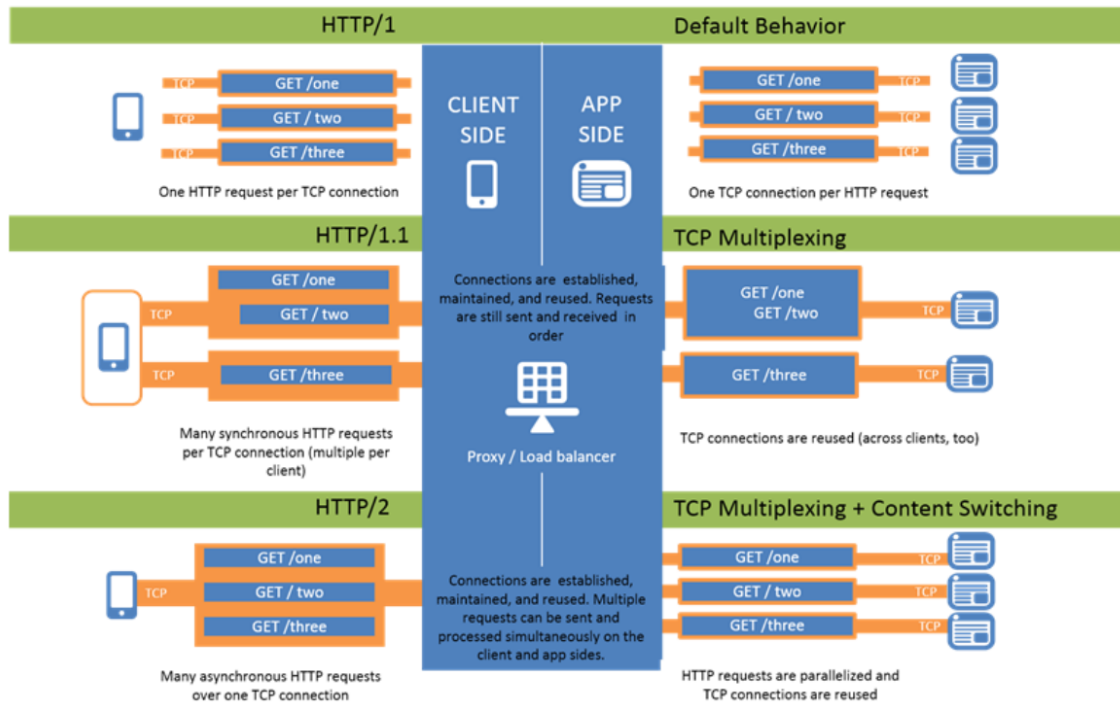
b. HTTP uses TCP

- i. client는 80번 포트로 서버에 TCP connection을 시작 (socket 생성)
- ii. server는 client로부터 온 TCP connection 을 수락
- iii. client 와 server는 메시지 교환
- iv. TCP connection 종료

c. HTTP is “stateless”

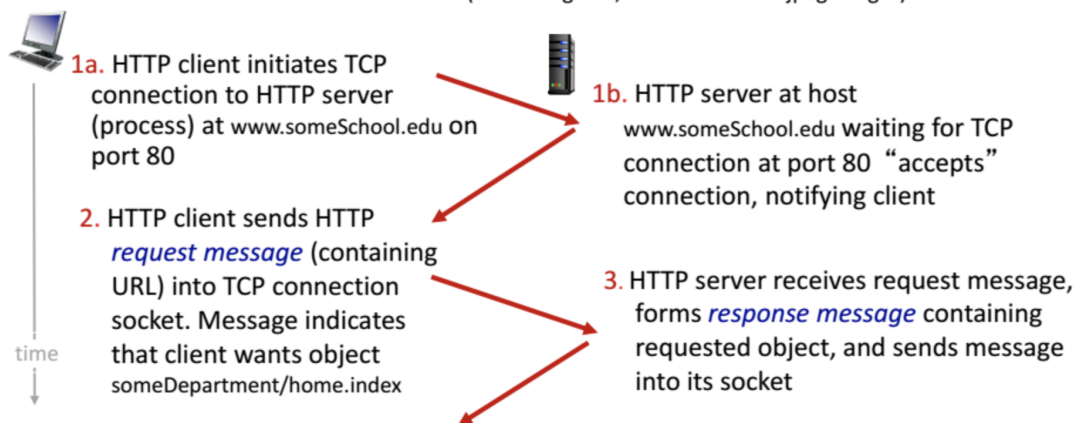
- a. server 는 client 가 과거에 어떤 내용을 request 했는지에 대한 정보를 저장하지 않음.
- b. server나 client가 죽으면 내용이 다 사라지므로 inconsistent 하다.

3. HTTP Connections

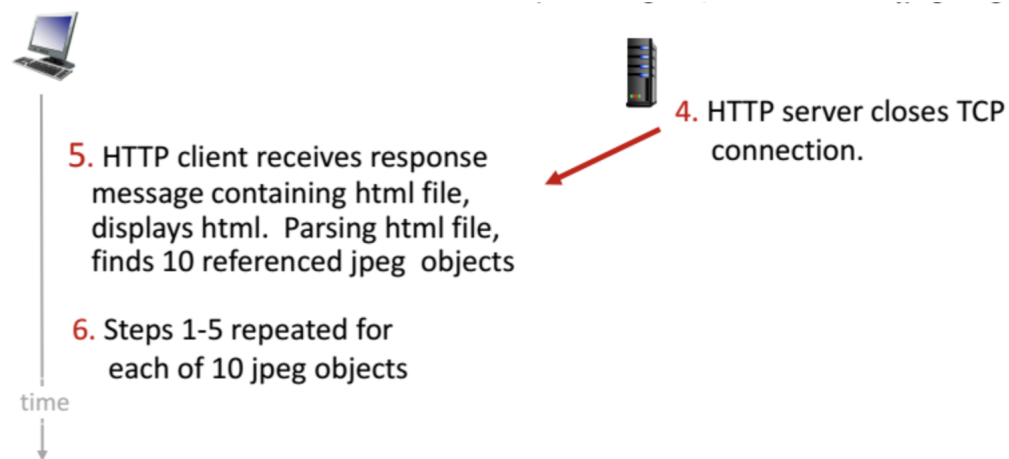


1. Non-persistent HTTP (HTTP 1.0)

- TCP connection이 만들어지면 하나의 object 로만 sent 가능함
- 여러개의 object를 받으려면 multiple connections 가 필요
- RTT : client 에서 server 까지 packet 이 왕복하는 시간
- HTTP response time
 - 1 RTT 는 TCP connection 연결에 필요
 - 연결 후 object를 요청 하고 받는데 1 RTT 필요 (connection 유지 하지 않는 것의 단점)
 - file size 가 클수록 file trasmission time 이 증가
 - 결과적으로
$$\text{Non-persistent HTTP response time} = 2\text{RTT} + \text{file transmission time}$$



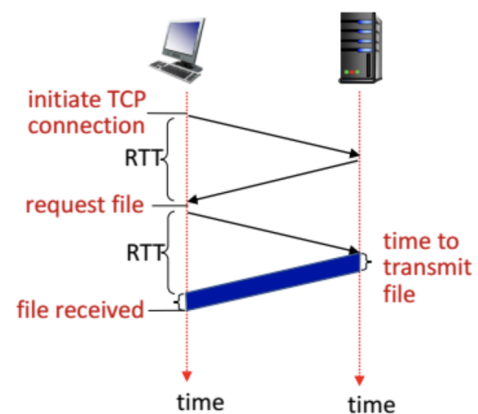
1a - 1b - 2 : three hand shaking → TCP 통신에서 정확한 전송을 보장하기 위해 상대방 컴퓨터와의 초기 환경 수립 과정



RTT (definition): time for a small packet to travel from client to server and back

HTTP response time (per object):

- one RTT to initiate TCP connection
- one RTT for HTTP request and first few bytes of HTTP response to return
- object/file transmission time



Non-persistent HTTP response time = 2RTT+ file transmission time

page9

b. Persistent HTTP (HTTP 1.1)

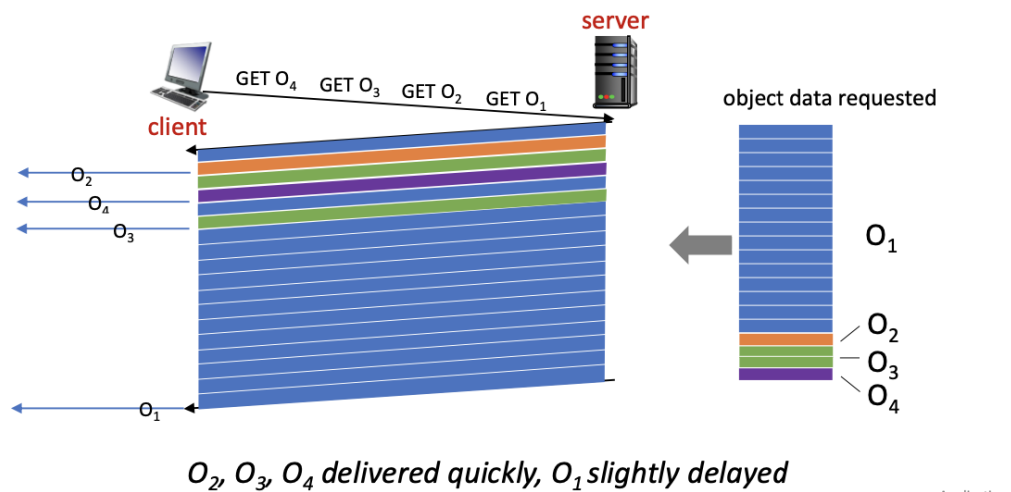
- multiple object 한개를 TCP Connection으로 보낼 수 있음

- b. sender 를 끊지 않고 됨 → client 와 server 가 지속적으로 메시지를 주고받음
- c. FCFS (first-come-first-served scheduling) 으로 작업하기 때문에
HOL(head-of-line blocking) 문제가 발생 → 첫번째 오브젝트부터 처리를 하는데 이후 오브젝트들은 너무 오래 걸리는 문제



c. HTTP 2

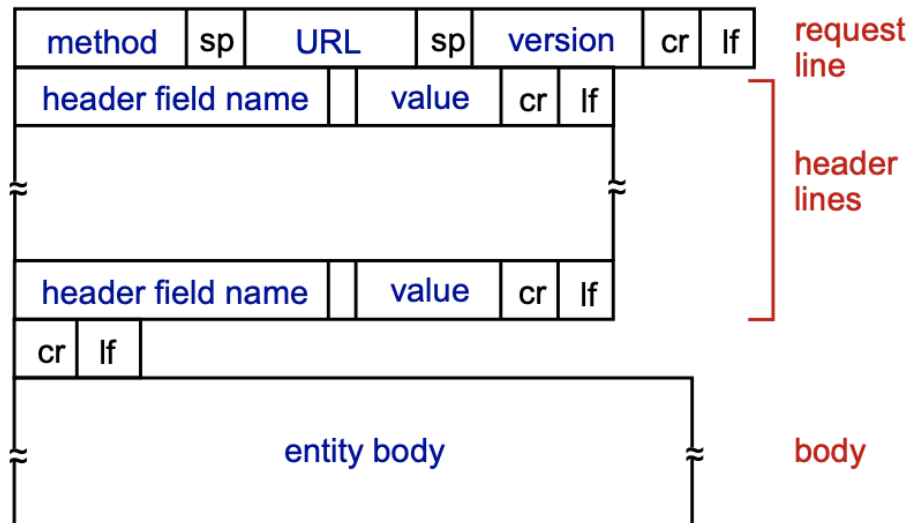
- a. 병렬적으로 보냄 → 처리시간 개선
- b. HOL blocking 을 해결하고자 object를 frame 로 나누고 Round Robin 방식으로 전송



4. HTTP request message

- a. 두가지 메시지 타입 : request, response

b. HTTP Request Message



i. ASCII 코드 사용

• ASCII (human-readable format)

```

request line (GET, POST, HEAD commands) → GET /index.html HTTP/1.1\r\n
                                                                    ↑
                                                                    carriage return character
                                                                    line-feed character
header lines → Host: www-net.cs.umass.edu\r\n
               User-Agent: Firefox/3.6.10\r\n
               Accept: text/html,application/xhtml+xml\r\n
               Accept-Language: en-us,en;q=0.5\r\n
               Accept-Encoding: gzip,deflate\r\n
               Accept-Charset: ISO-8859-1,utf-8;q=0.7\r\n
               Keep-Alive: 115\r\n
               Connection: keep-alive\r\n
               \r\n
carriage return, line feed at start of line indicates end of header lines

```

c. Method type

- i. POST
- ii. GET
- iii. HEAD
- iv. PUT
- v. DELETE

d. HTTP Response Message

```

status line (protocol status code status phrase) → HTTP/1.1 200 OK\r\n
                                                    Date: Sun, 26 Sep 2010 20:09:20 GMT\r\n
                                                    Server: Apache/2.0.52 (CentOS)\r\n
                                                    Last-Modified: Tue, 30 Oct 2007 17:00:02 GMT\r\n
header lines → ETag: "17dc6-a5c-bf716880"\r\n
                Accept-Ranges: bytes\r\n
                Content-Length: 2652\r\n
                Keep-Alive: timeout=10, max=100\r\n
                Connection: Keep-Alive\r\n
                Content-Type: text/html; charset=ISO-8859-1\r\n
data, e.g., requested HTML file → \r\n
                                   data data data data data ...
  
```

e. HTTP Response Status Codes

200 OK

- request succeeded, requested object later in this message

301 Moved Permanently

- requested object moved, new location specified later in this message (in Location: field)

400 Bad Request

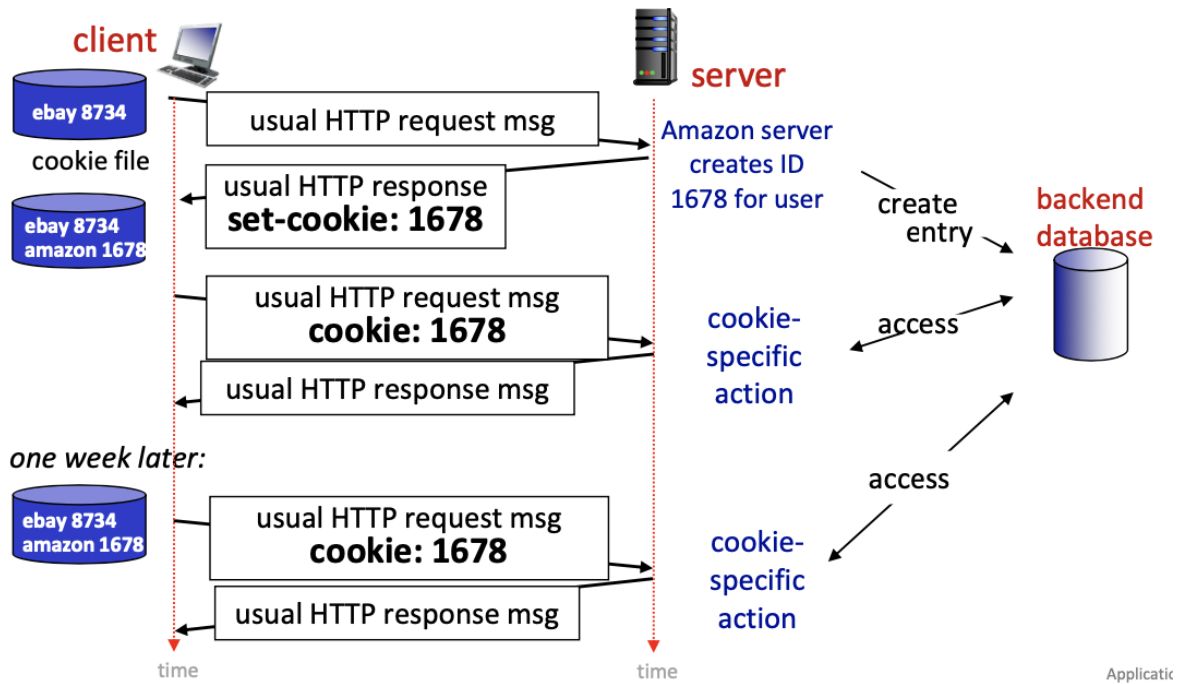
- request msg not understood by server

404 Not Found

- requested document not found on this server

505 HTTP Version Not Supported

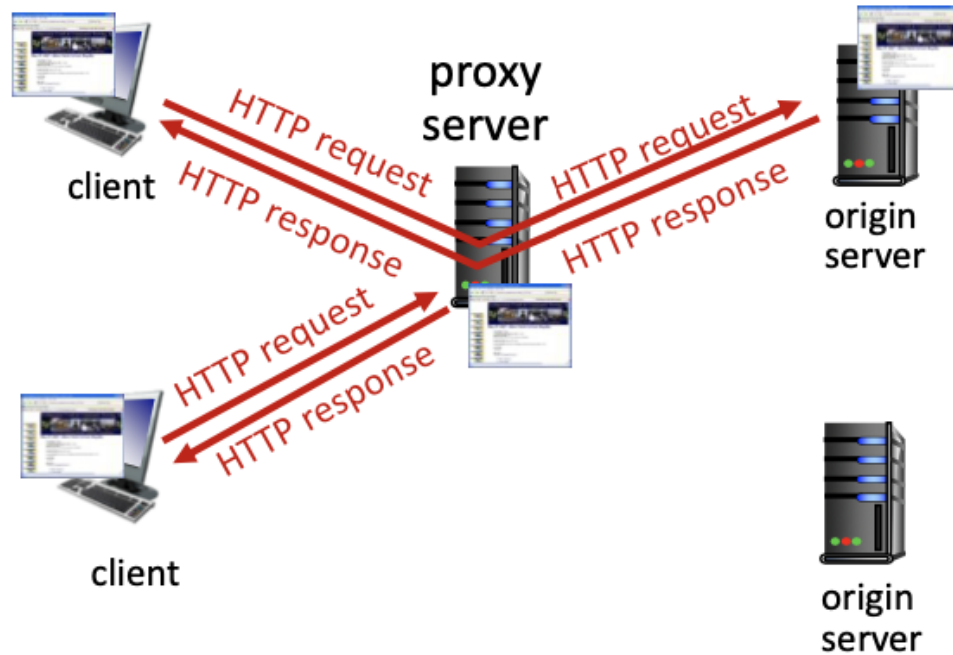
Cookies



1. 웹 사이트와 클라이언트 브라우저는 트랜잭션 사이 상태를 유지하기 위해 쿠키를 사용한다.
2. client local 에 저장
3. header 에 부가적으로 바꿔서 넣음
4. cookie 를 통해 재방문 여부(state)를 알 수 있음

Web caches (proxy servers)

1. origin server 까지 가지 않고 proxy server 에서 처리를 하려고 함 (왜? 가까워서 빠르니깐!)



- a. browser 가 HTTP request 를 cache에 보냄
 - b. proxy server에서 request 를 확인 후 있으면 주고 없으면 origin server 에 요청하여 받아서 보냄
 - c. proxy server 가 client, server 역할을 다한다. client 와 가까이 있어야 한다 (propagation delay 를 줄이기 위해)
2. Why web caching?
- a. for client : 응답 시간 감소
 - b. for server : 더 많은 유저에게 제공 가능 (서버 부담이 줄기 때문)
 - c. for local ISP : link bandwidth 를 효율적으로 활용 가능 (서버부담이 줄기 때문)
3. conditional GET
- a. cache 에 유효 날짜가 있는데 이를 지나면 response 요청 보냄. → 같으면 304 not modified 다르면 200 ok 후 보냄
 - b. 이후 client 가 요청하면 보내줌

E-mail → SMTP, IMAP, POP 프로토콜 사용하는 정도만 알면 될듯

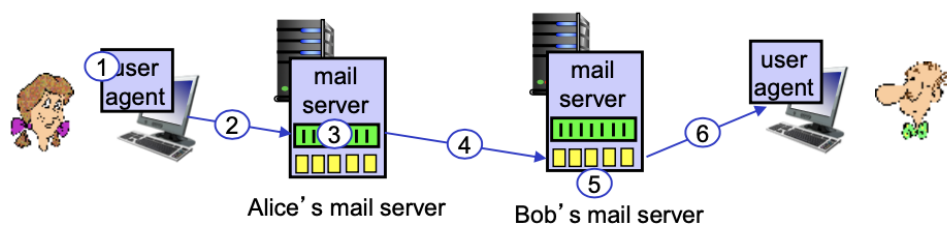
1. Three major components

a. user agents

- i. aka 'mail reader'
- ii. 읽기, 수정, 구성
- iii. 나가고 들어오는 message를 mail server 에 저장
- iv. e.g. Outlook, iPhone mail client

b. mail servers

- i. mailbox 유지 user를 위해 들어오는 message 저장
- ii. 나가는 mail message 의 message queue 구성
- iii. SMTP 사용 하여 mail server 끼리 메세지 보냄



c. SMTP protocol

- i. TCP 사용
- ii. port 25
- iii. direct transfer : 서버끼리 직접 보냄
- iv. 3개의 단계
 1. three handshaking
 2. 메세지 주고 받음
 3. closure
- v. command : ASCII text / response : status code and phrase

DNS : Domain Name System

1. DNS

a. 분산 데이터베이스 : 많은 name server의 계층구조

2. DNS Service

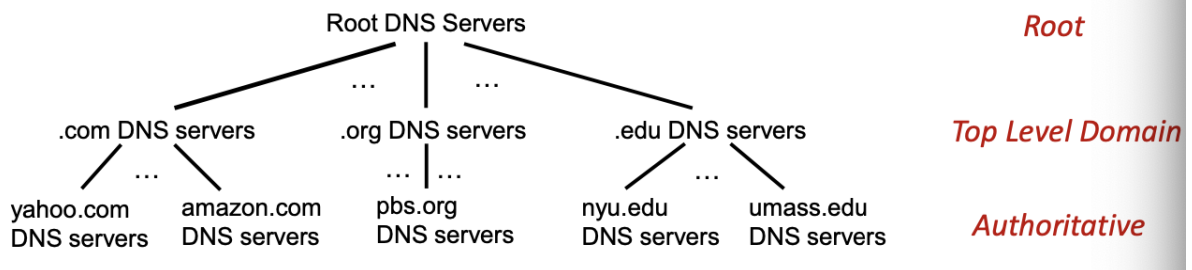
a. hostname 을 IP주소로 번역

b. 같은 호스트라도 여러개의 이름 가능 (alias)

3. 왜 DNS를 중앙집중형으로 안 만들까?

a. DNS Server가 죽으면 복구 불가

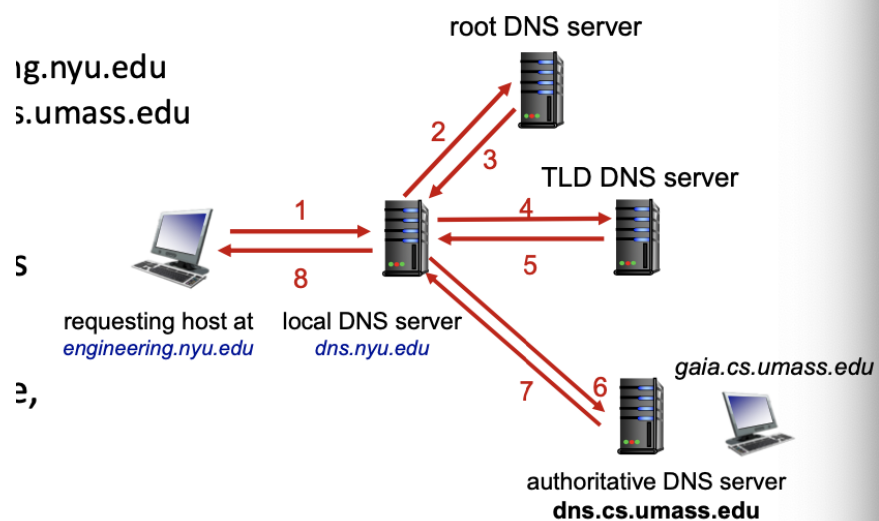
b. traffic이 몰릴 수 있으므로



4. 전 세계적으로 root dns 는 13개

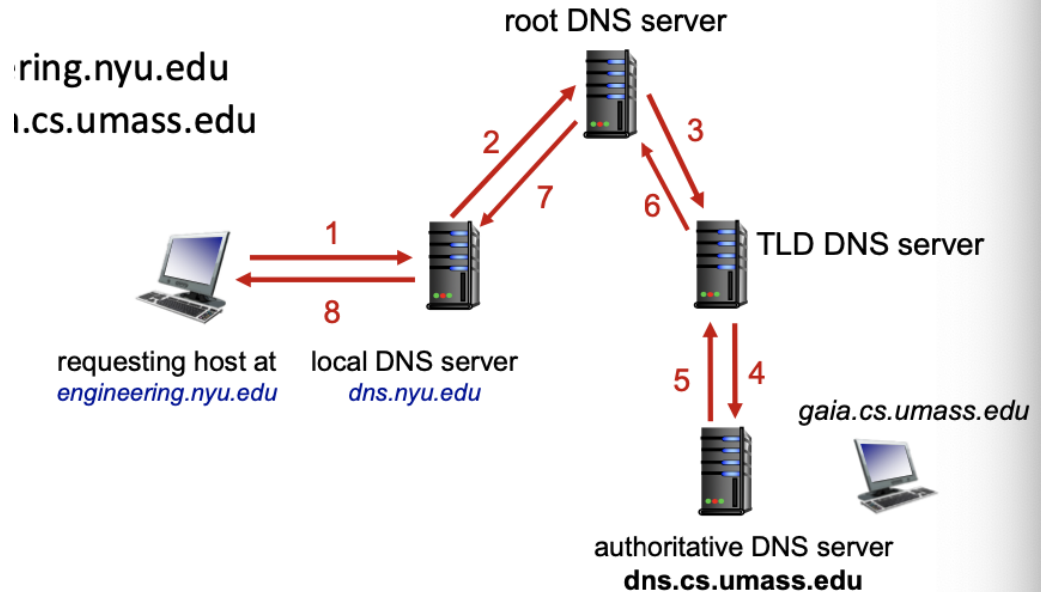
5. DNS name resolution

a. iterated query



i.

b. recursive query



i.

6. Caching DNS → 시간을 가지고 caching 함

7. DNS Records

DNS: distributed database storing resource records (RR)

RR format: (name, value, type, ttl)

type=A

- name is hostname
- value is IP address

type=NS

- name is domain (e.g., `foo.com`)
- value is hostname of authoritative name server for this domain

type=CNAME

- name is alias name for some "canonical" (the real) name
- `www.ibm.com` is really `serveeast.backup2.ibm.com`
- value is canonical name

type=MX

- value is name of mailserver associated with name

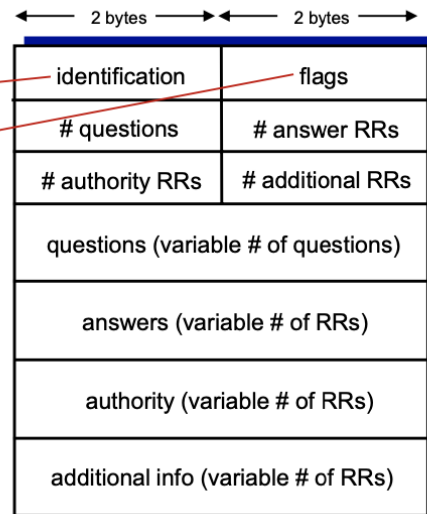
ttl : 유효시간

8. DNS Protocol and Messages

DNS *query* and *reply* messages, both have same *format*:

message header:

- **identification**: 16 bit # for query,
reply to query uses same #
- **flags**:
 - query or reply
 - recursion desired
 - recursion available
 - reply is authoritative



flags 안에는 query 혹은 reply 가 들어감

9. DNS Security → DDOS 공격을 받을 수 있음.