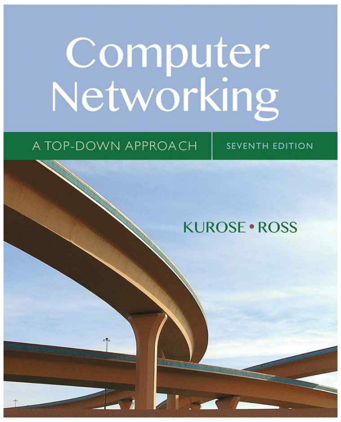
Chapter 2
Application Layer



Seongwook Youn
Department of Software
Korea National University of Transportation

The slides are adapted from the publisher's material

Computer Networking: A Top Down Approach
6th edition
Jim Kurose, Keith Ross
Addison-Wesley
April 2016

Chapter 2: outline

2.1 principles of network applications
- app architectures
- app requirements

2.2 Web and HTTP

2.3 electronic mail
- SMTP, POP3, IMAP

2.4 DNS

2.5 P2P applications

2.6 video streaming and content distribution networks (CDNs)

전자 메일

Electronic mail

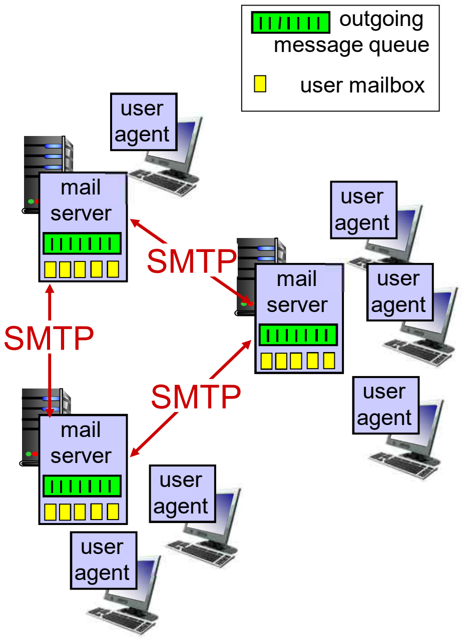
Three major components:

- user agents
- mail servers
- simple mail transfer protocol: SMTP

User = computer

User Agent

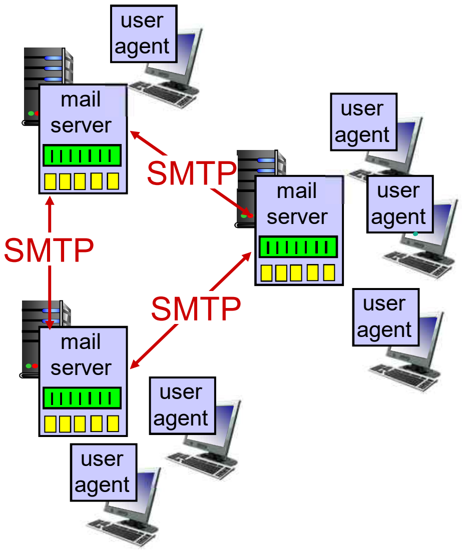
- a.k.a. "mail reader"
- 작성, 조립 composing, editing, reading mail messages
- e.g., Outlook, Thunderbird, iPhone mail client
- outgoing, incoming messages stored on server



Electronic mail: mail servers

mail servers:

- *mailbox* contains 도착하는 incoming messages for user
- *message queue* of 나가는 outgoing (to be sent) mail messages
- simple mail transfer protocol *SMTP protocol* between mail servers to send email messages
  - client: sending mail server
  - "server": receiving mail server



messages는 message queue를 통해 다른 메일 서버의 mailbox로 들어가서 메시지를 주고 받습니다.

Electronic Mail: SMTP [RFC 2821]

- uses TCP to reliably (신뢰할 수 있어) transfer email message from client to server, port 25
- direct transfer: sending server to receiving server
- three phases of transfer
  - handshaking (greeting) "너에게 보낼 메시지가 있어~" 손흔들어줌
  - transfer of messages
  - closure
- command/response interaction (like HTTP, FTP)
  - commands: ASCII text
  - response: status code and phrase
- messages must be in 7-bit ASCI

Scenario: Alice sends message to Bob

1) Alice uses UA (user agent) to compose message "to" `bob@someschool.edu`

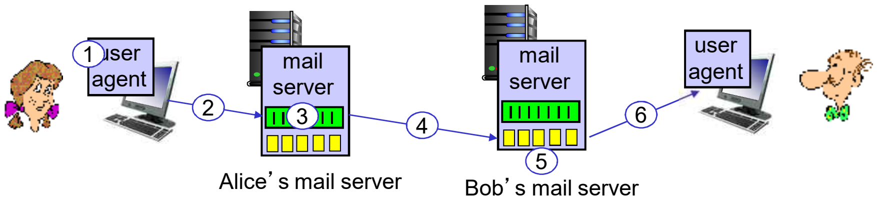
2) Alice's UA sends message to her mail server; message placed in message queue

3) client side of SMTP opens TCP connection with Bob's mail server

4) SMTP client sends Alice's message over the TCP connection 연결하고 보내고 닫습니다.

5) Bob's mail server places the message in Bob's mailbox

6) Bob invokes (부르다) his user agent to read message



Sample SMTP interaction

server <-> client

```
S: 220 hamburger.edu
C: HELO crepes.fr
S: 250  Hello crepes.fr, pleased to meet you
C: MAIL FROM: <alice@crepes.fr>
S: 250 alice@crepes.fr... Sender ok
C: RCPT TO: <bob@hamburger.edu>
S: 250 bob@hamburger.edu ... Recipient ok
C: DATA --> 메시지가 끝났음을 알림
S: 354 Enter mail, end with "." on a line by itself
C: Do you like ketchup?
C: How about pickles?
C: .
S: 250 Message accepted for delivery
C: QUIT 끝!
S: 221 hamburger.edu closing connection  서버가 닫습니다.
```

SMTP: final words

끈질긴, 집요한

- SMTP uses persistent connections
- SMTP requires message (header & body) to be in 7-bit ASCII
- SMTP server uses `CRLF.CRLF` to determine end of message

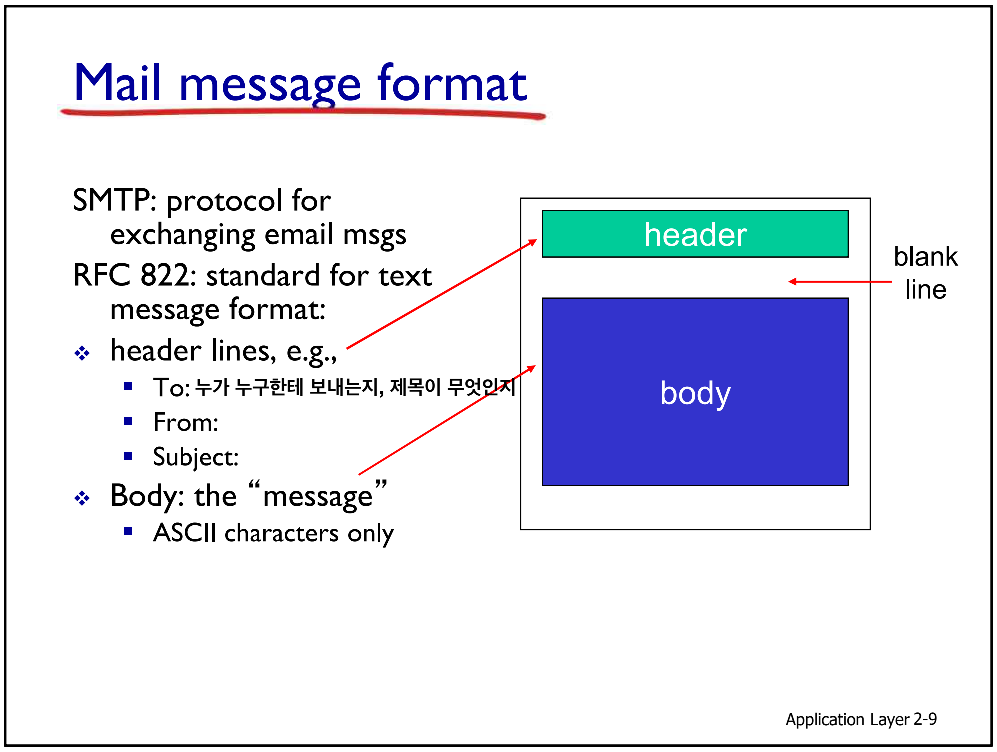
comparison with HTTP:

- HTTP: pull
- SMTP: push
- both have ASCII command/response interaction, status codes
- HTTP: each object encapsulated in its own response msg

  요약하다

  각각의 object가 고유의 response를 가지고 보내집니다.
- SMTP: multiple objects sent in multipart msg

  여러 개의 object가 여러 개로 보내집니다.



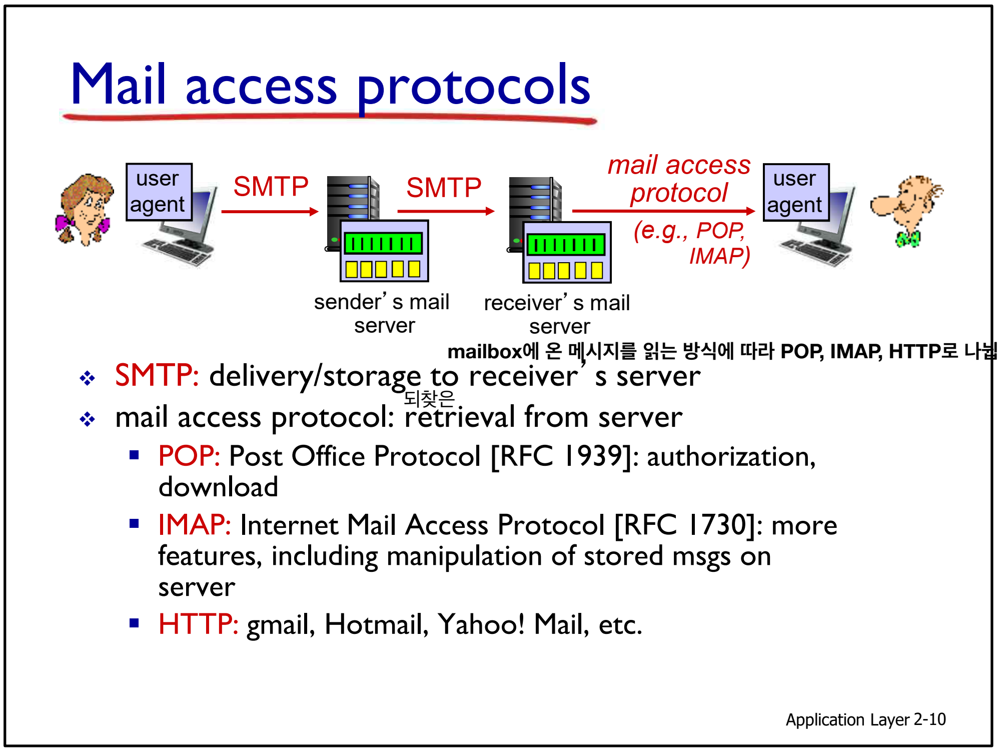
Mail message format

SMTP: protocol for exchanging email msgs

RFC 822: standard for text message format:

- header lines, e.g.,
  - To: 누가 누구한테 보내는지, 제목이 무엇인지
  - From:
  - Subject:
- Body: the "message"
  - ASCII characters only

header
blank line
body

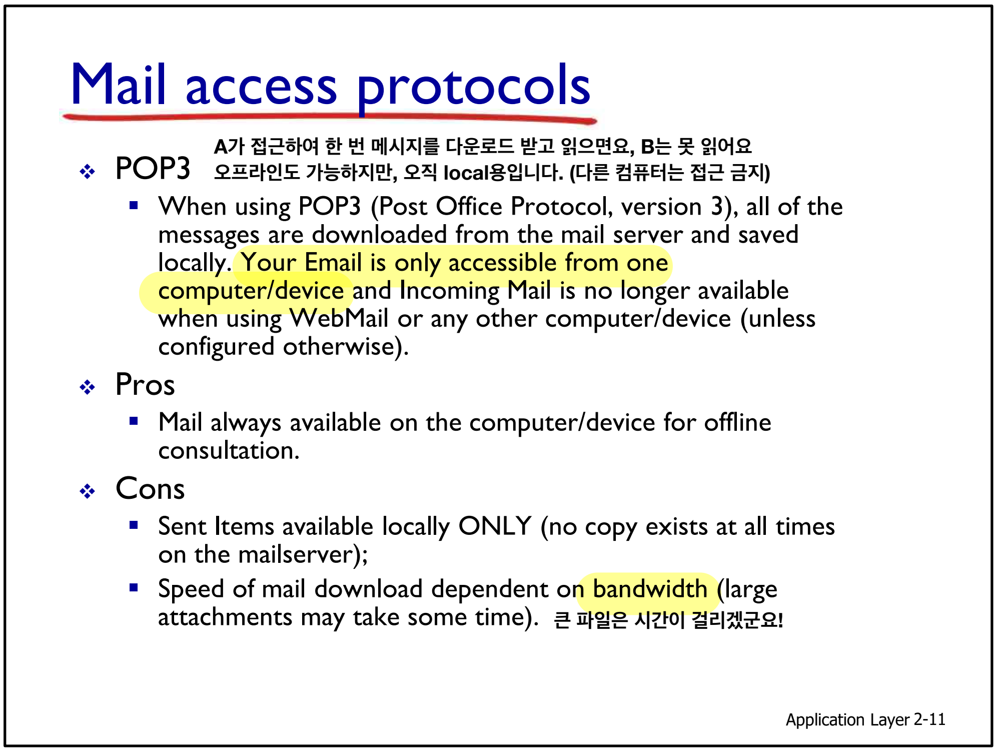


Mail access protocols

user agent
SMTP
SMTP
mail access protocol
(e.g., POP, IMAP)
user agent
sender's mail server
receiver's mail server

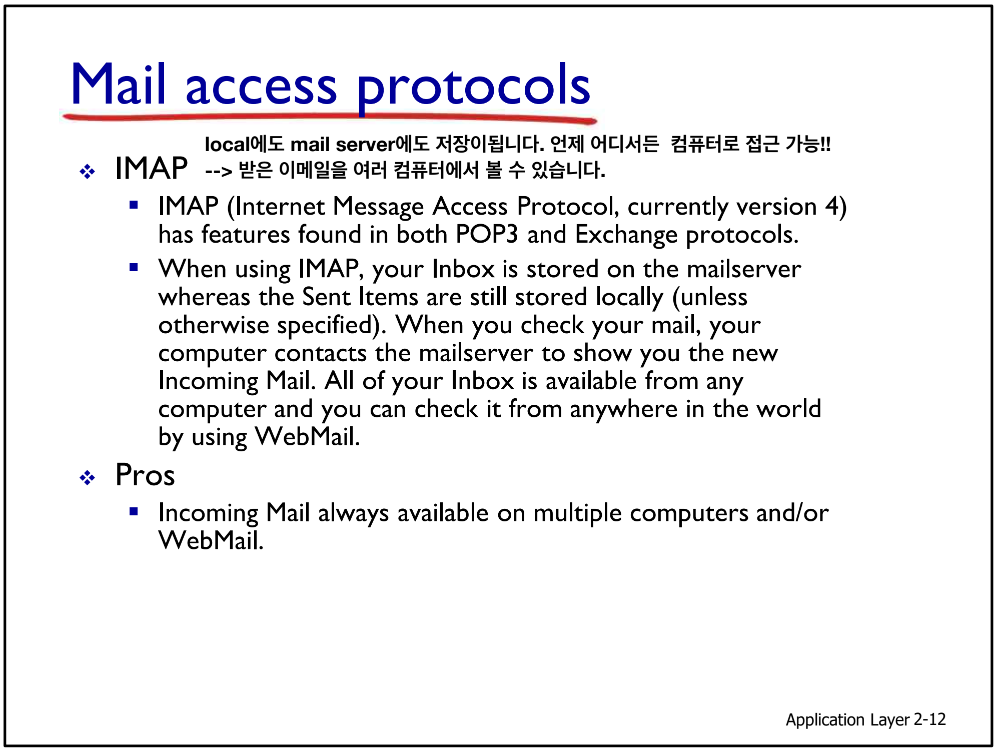
mailbox에 온 메시지를 읽는 방식에 따라 POP, IMAP, HTTP로 나뉩니다.

- SMTP: delivery/storage to receiver's server
- mail access protocol: 되찾은 retrieval from server
  - POP: Post Office Protocol [RFC 1939]: authorization, download
  - IMAP: Internet Mail Access Protocol [RFC 1730]: more features, including manipulation of stored msgs on server
  - HTTP: gmail, Hotmail, Yahoo! Mail, etc.



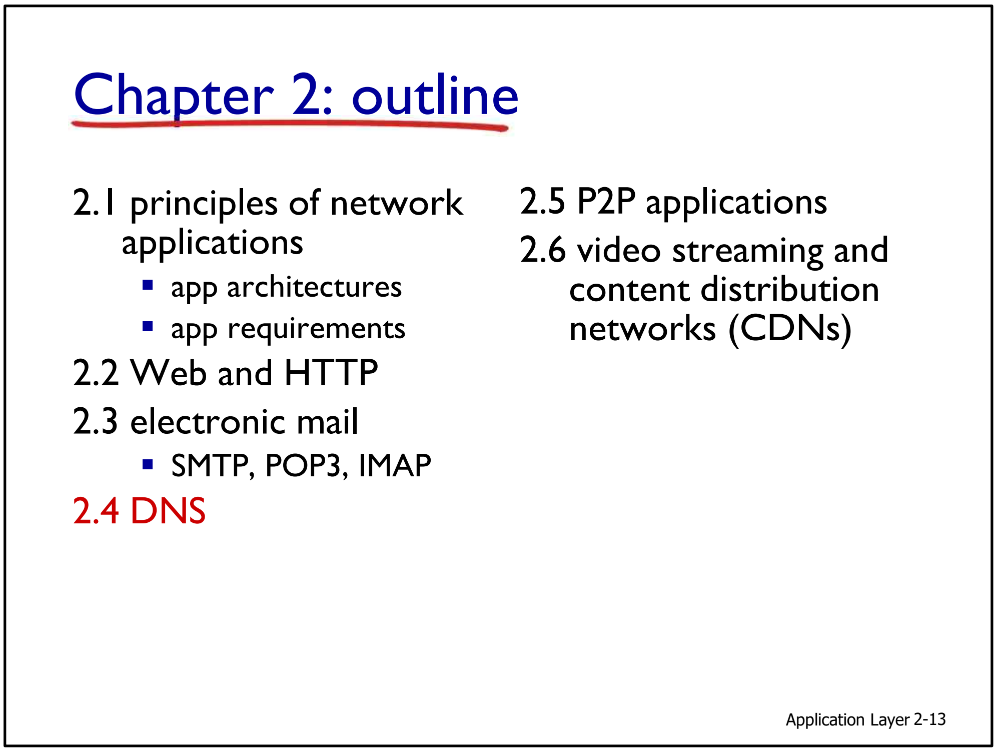
Mail access protocols

- POP3
  A가 접근하여 한 번 메시지를 다운로드 받고 읽으면요, B는 못 읽어요
  오프라인도 가능하지만, 오직 local용입니다. (다른 컴퓨터는 접근 금지)
  - When using POP3 (Post Office Protocol, version 3), all of the messages are downloaded from the mail server and saved locally. Your Email is only accessible from one computer/device and Incoming Mail is no longer available when using WebMail or any other computer/device (unless configured otherwise).
- Pros
  - Mail always available on the computer/device for offline consultation.
- Cons
  - Sent Items available locally ONLY (no copy exists at all times on the mailserver);
  - Speed of mail download dependent on bandwidth (large attachments may take some time). 큰 파일은 시간이 걸리겠군요!



Mail access protocols

- IMAP
  local에도 mail server에도 저장이됩니다. 언제 어디서든 컴퓨터로 접근 가능!!
  --> 받은 이메일을 여러 컴퓨터에서 볼 수 있습니다.
  - IMAP (Internet Message Access Protocol, currently version 4) has features found in both POP3 and Exchange protocols.
  - When using IMAP, your Inbox is stored on the mailserver whereas the Sent Items are still stored locally (unless otherwise specified). When you check your mail, your computer contacts the mailserver to show you the new Incoming Mail. All of your Inbox is available from any computer and you can check it from anywhere in the world by using WebMail.
- Pros
  - Incoming Mail always available on multiple computers and/or WebMail.



Chapter 2: outline

2.1 principles of network applications
- app architectures
- app requirements

2.2 Web and HTTP

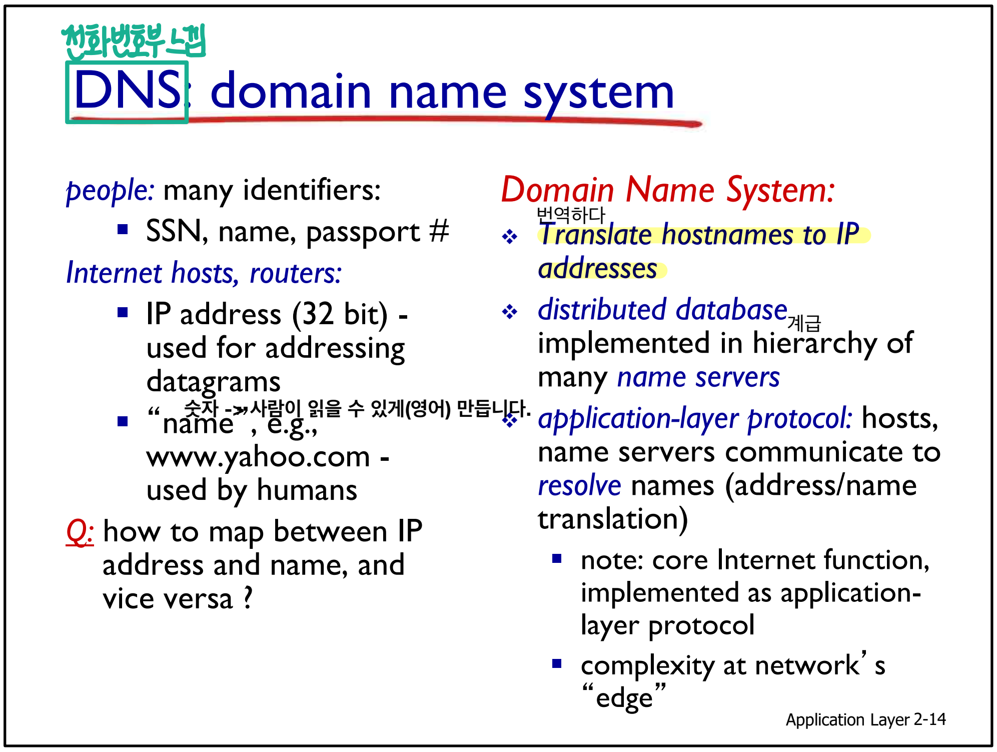
2.3 electronic mail
- SMTP, POP3, IMAP

2.4 DNS

2.5 P2P applications

2.6 video streaming and content distribution networks (CDNs)

Application Layer 2-13



전화번호부 느낌

DNS: domain name system

people: many identifiers:
- SSN, name, passport #

Internet hosts, routers:
- IP address (32 bit) - used for addressing datagrams
- 숫자 -> 사람이 읽을 수 있게(영어) 만듭니다.
  "name", e.g., www.yahoo.com - used by humans

Q: how to map between IP address and name, and vice versa ?

Domain Name System:
- 번역하다
  Translate hostnames to IP addresses
- *distributed database* 계급 implemented in hierarchy of many *name servers*
- *application-layer protocol:* hosts, name servers communicate to *resolve* names (address/name translation)
  - note: core Internet function, implemented as application-layer protocol
  - complexity at network's "edge"

Application Layer 2-14

왜 분산 하는지 알아두세요.

DNS: services, structure

DNS services

host이름을 IP 주소로 전환합니다.

- hostname to IP address translation
- host aliasing을 이용합니다.

  원래 이름 우리가 알기 쉽게 만들어 놓은 것

  - canonical, alias names
    - relay1.west-coast.enterprise.com (Canonical host)
    - ➔ two aliases (enterprise.com and www.enterprise.com)
- mail server aliasing

  결국 로드를 분산시킵니다.
- load distribution
  - replicated Web servers: many IP addresses correspond to one name

왜 분산이냐고? 하나에 집중했다가 고장나면 어떡해요.

why not centralize DNS?

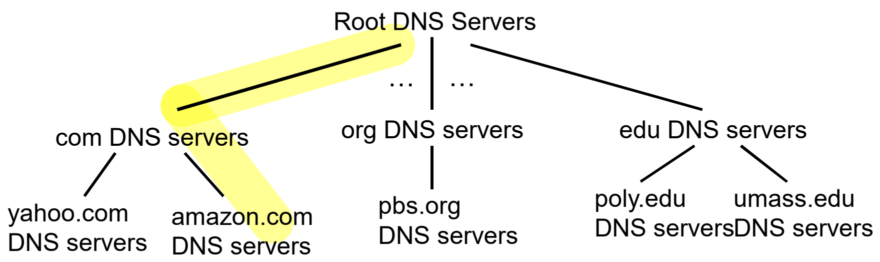
- single point of failure

  트래픽이 몰리면 어떡해요
- traffic volume
- distant centralized database
- maintenance

A: doesn’t scale!

분산적이고 계층적인 데이터베이스다.

DNS: a distributed, hierarchical database



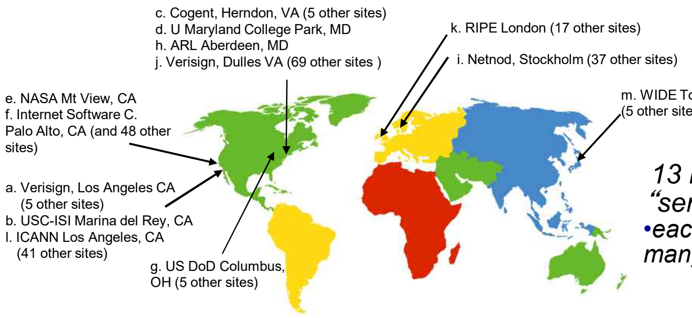
client wants IP for www.amazon.com; 1st approx:

- client queries root server to find com DNS server
- client queries .com DNS server to get amazon.com DNS server
- client queries amazon.com DNS server to get IP address for www.amazon.com

[0] 아마존 IP를 클라이언트가 원한다면, Root Server에게 물어봅니다.
[1] com DNS 서버를 알게 됩니다.
[2] amazone ip 주소로 받게 됩니다.
=> 계층적임을 보여줌

DNS: root name servers

- contacted by local name server that can not resolve name
- root name server:
  - contacts authoritative name server if name mapping not known
  - gets mapping
  - returns mapping to local name server


c. Cogent, Herndon, VA (5 other sites)
d. U Maryland College Park, MD
h. ARL Aberdeen, MD
j. Verisign, Dulles VA (69 other sites)
k. RIPE London (17 other sites)
i. Netnod, Stockholm (37 other sites)
e. NASA Mt View, CA
f. Internet Software C.
Palo Alto, CA (and 48 other sites)
m. WIDE Tokyo
(5 other sites)
a. Verisign, Los Angeles CA
(5 other sites)
b. USC-ISI Marina del Rey, CA
l. ICANN Los Angeles, CA
(41 other sites)
g. US DoD Columbus, OH (5 other sites)

13 logical root name “servers” worldwide
•each “server” replicated many times

TLD, authoritative servers

그냥 그렇구나.

top-level domain (TLD) servers:

- responsible for com, org, net, edu, aero, jobs, museums, and all top-level country domains, e.g.: uk, fr, ca, jp
- Network Solutions maintains servers for .com TLD
- Edu cause for .edu TLD

authoritative DNS servers:

- organization’s own DNS server(s), providing authoritative hostname to IP mappings for organization’s named hosts
- can be maintained by organization or service provider

Local DNS name server

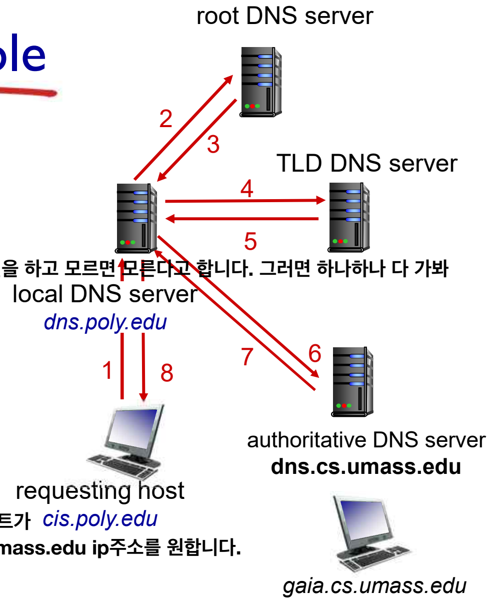
- does not strictly belong to hierarchy
- each ISP (residential ISP, company, university) has one
  - also called "default name server"
- when host makes DNS query, query is sent to its local DNS server **DNS 쿼리 ->local DNS로 보내집니다.**
  - has local cache of recent name-to-address translation pairs (but may be out of date!)
  - acts as proxy, forwards query into hierarchy

DNS name resolution example

- host at cis.poly.edu wants IP address for gaia.cs.umass.edu

iterated query:

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



[1] 알면 서버와 연결을 하고 모르면 모른다고 합니다. 그러면 하나하나 다 가봐야 합니다.

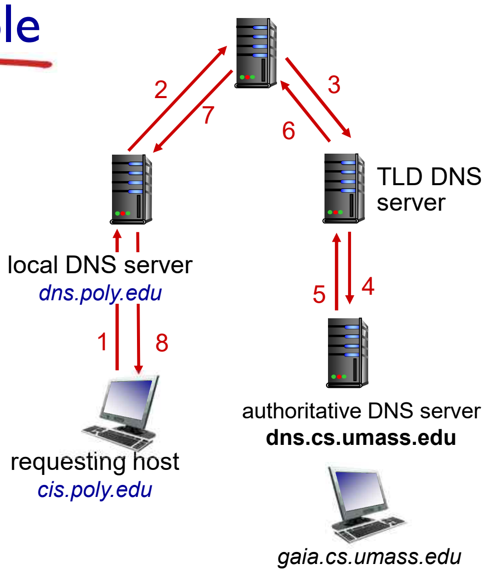
[0] 이 호스트가 gaia.cs.umass.edu ip주소를 원합니다.

DNS name resolution example

recursive query:

- puts burden of name resolution on contacted name server
- heavy load at upper levels of hierarchy?

위의 레벨로 갈수록 노드가 많이 몰릴 것입니다.



DNS: caching, updating records

한 번 물어보면, 일정 시간 동안 cashes로 남겨둡니다. 또 올 수도 있으니깐!!

- once (any) name server learns mapping, it *caches* mapping
  - cache entries timeout (disappear) after some time (TTL)
  - TLD servers typically cached in local name servers
    - thus root name servers not often visited

자동 소멸이 될 때까지는 알려지지 않을 채로 남겨 있습니다.

- cached entries may be *out-of-date* (best effort name-to-address translation!)
  - if name host changes IP address, may not be known Internet-wide until all TTLs expire
- update/notify mechanisms proposed IETF standard
  - RFC 2136 그래서 업데이트하고 알려주는 것이 필요합니다.

알아둬야 할 것 같음

DNS records

일종의 분산 DB입니다.

DNS: distributed db storing resource records (RR)

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

time to leave: 얼마 동안 유지 되는지

type=A

- **`name`** is hostname
- **`value`** is IP address

EX - (relay1.bar.foo.com, 145.37.93.126, A)

type=NS

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

EX - (foo.com, dns.foo.com, NS)

type=CNAME

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

EX - (www.ibm.com, servereast.backup2.ibm.com, CNAME)

type=MX 그 서버와 관련된 메일 서버입니다.

- **`value`** is name of mail server associated with **`name`**

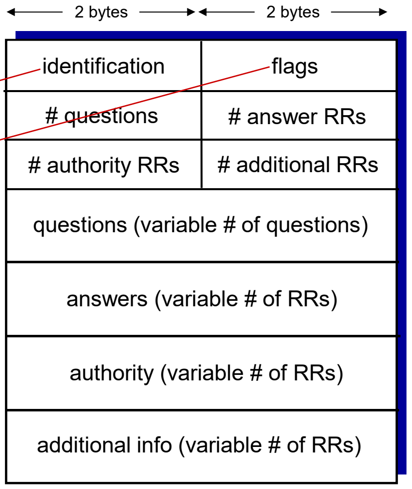
EX - (foo.com, mail.bar.foo.com, MX)

DNS protocol, messages

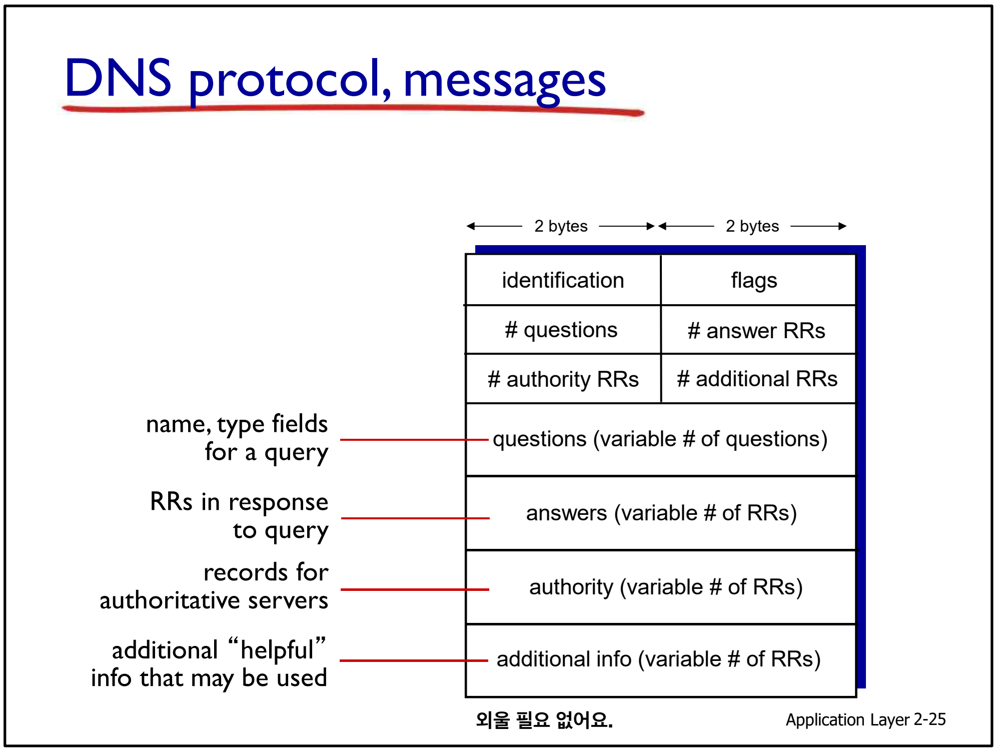
- *query* and *reply* messages, both with same *message format*

msg header

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



2 bytes	2 bytes
identification	flags
# questions	# answer RRs
# authority RRs	# additional RRs
questions (variable # of questions)	
answers (variable # of RRs)	
authority (variable # of RRs)	
additional info (variable # of RRs)	

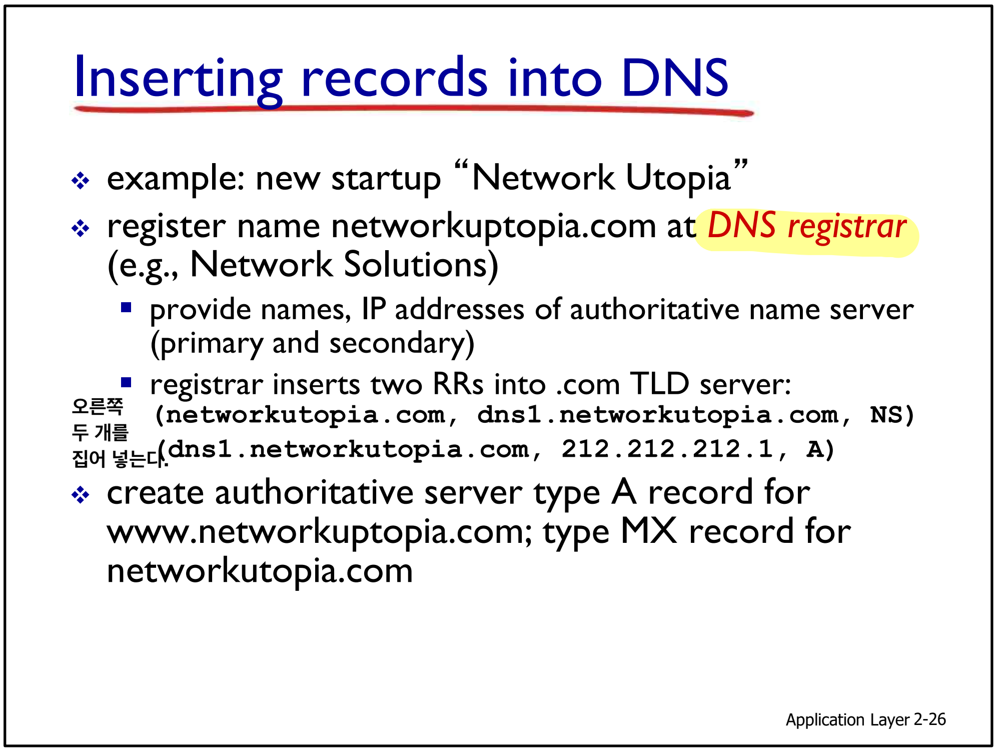


DNS protocol, messages

2 bytes	2 bytes
identification	flags
# questions	# answer RRs
# authority RRs	# additional RRs
questions (variable # of questions)	
answers (variable # of RRs)	
authority (variable # of RRs)	
additional info (variable # of RRs)	

name, type fields for a query
RRs in response to query
records for authoritative servers
additional "helpful" info that may be used

외울 필요 없어요.



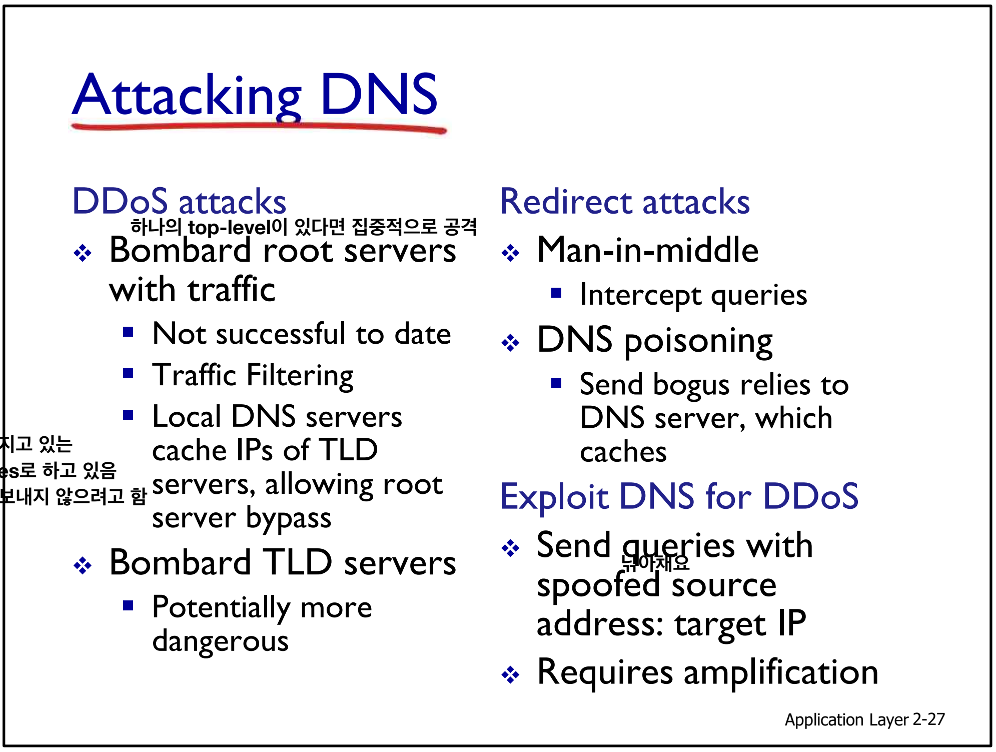
Inserting records into DNS

- example: new startup "Network Utopia"
- register name networkuptopia.com at *DNS registrar* (e.g., Network Solutions)
  - provide names, IP addresses of authoritative name server (primary and secondary)
  - registrar inserts two RRs into .com TLD server:

```
(networkutopia.com, dns1.networkutopia.com, NS)
(dns1.networkutopia.com, 212.212.212.1, A)
```

오른쪽 두 개를 집어 넣는다.

- create authoritative server type A record for www.networkuptopia.com; type MX record for networkutopia.com



Attacking DNS

DDoS attacks

하나의 top-level이 있다면 집중적으로 공격

- Bombard root servers with traffic
  - Not successful to date
  - Traffic Filtering
  - Local DNS servers cache IPs of TLD servers, allowing root server bypass
- Bombard TLD servers
  - Potentially more dangerous

웬만하면
local server가
root server가 가지고 있는
ip 주소들을 cashes로 하고 있음
root server까지 보내지 않으려고 함

Redirect attacks

- Man-in-middle
  - Intercept queries
- DNS poisoning
  - Send bogus relies to DNS server, which caches

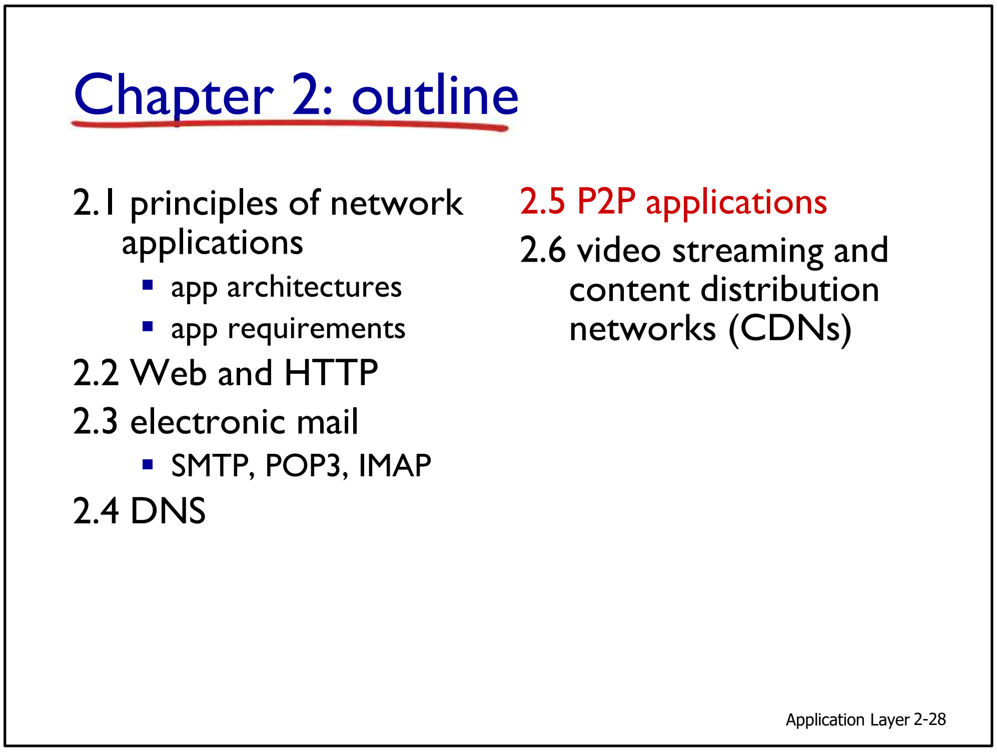
Exploit DNS for DDoS

- Send queries with spoofed source address: target IP

뉘아채요

- Requires amplification

Application Layer 2-27



Chapter 2: outline

2.1 principles of network applications
- app architectures
- app requirements

2.2 Web and HTTP

2.3 electronic mail
- SMTP, POP3, IMAP

2.4 DNS

2.5 P2P applications

2.6 video streaming and content distribution networks (CDNs)

Application Layer 2-28