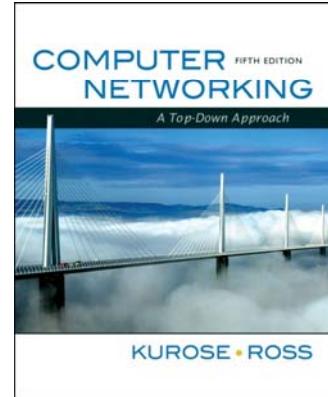


Chapter 2

Application Layer



A note on the use of these ppt slides:

We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) in substantially unaltered form, that you mention their source (after all, we'd like people to use our book!)
- If you post any slides in substantially unaltered form on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

Thanks and enjoy! JFK/KWR

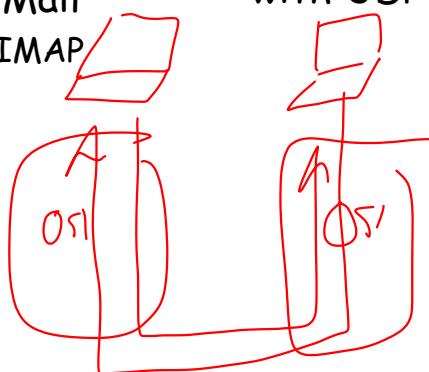
All material copyright 1996-2009
J.F Kurose and K.W. Ross, All Rights Reserved

*Computer Networking:
A Top Down Approach,
5th edition.
Jim Kurose, Keith Ross
Addison-Wesley, April
2009.*

2: Application Layer 1

Chapter 2: Application layer

- 2.1 Principles of network applications
- 2.2 Web and HTTP
- 2.3 FTP
- 2.4 Electronic Mail
 - SMTP, POP3, IMAP
- 2.5 DNS
- 2.6 P2P applications
- 2.7 Socket programming with TCP
- 2.8 Socket programming with UDP



2: Application Layer 2

Chapter 2: Application Layer

- Our goals:
- conceptual, implementation aspects of network application protocols
 - transport-layer service models
 - client-server paradigm
 - peer-to-peer paradigm
- learn about protocols by examining popular application-level protocols
 - HTTP
 - FTP
 - SMTP / POP3 / IMAP
 - DNS
- programming network applications
 - socket API

2: Application Layer 3

Some network apps

- e-mail
- web
- instant messaging
- remote login
- P2P file sharing
- multi-user network games
- streaming stored video clips
- voice over IP
- real-time video conferencing
- grid computing

2: Application Layer 4

protocol ດາວໂຫຼນທະບຽນເຕີຍໃຈໃດສໍາຜະກິນ

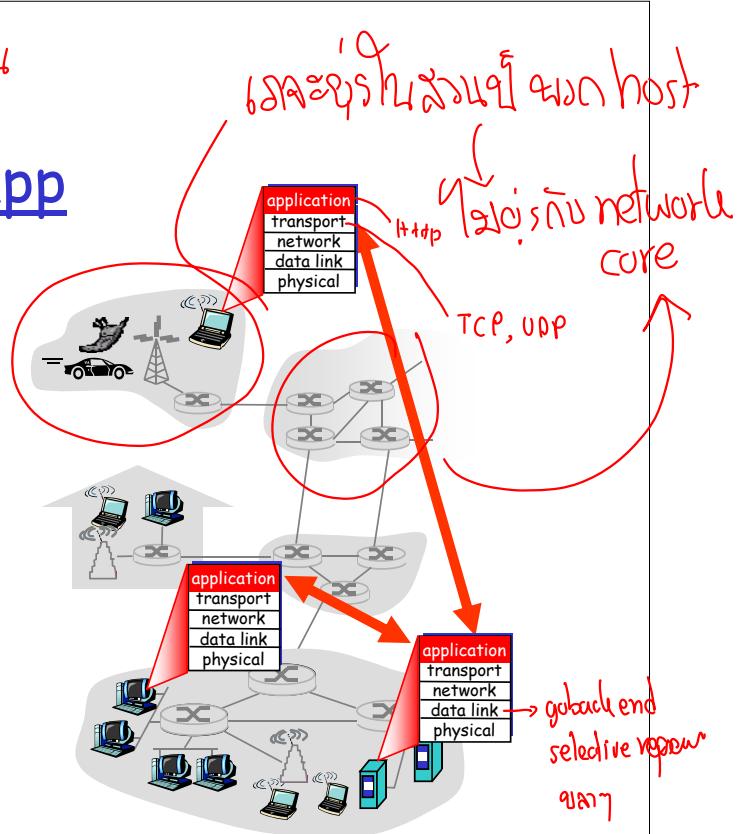
Creating a network app

write programs that

- ❖ run on (different) end systems
- ❖ communicate over network
- ❖ e.g., web server software communicates with browser software

No need to write software for network-core devices

- ❖ Network-core devices do not run user applications
- ❖ applications on end systems allows for rapid app development, propagation



2: Application Layer 5

Chapter 2: Application layer

- 2.1 Principles of network applications
- 2.2 Web and HTTP
- 2.3 FTP
- 2.4 Electronic Mail
 - SMTP, POP3, IMAP
- 2.5 DNS
- 2.6 P2P applications
- 2.7 Socket programming with TCP
- 2.8 Socket programming with UDP

2: Application Layer 6

Application architectures

- Client-server
- Peer-to-peer (P2P)
- Hybrid of client-server and P2P

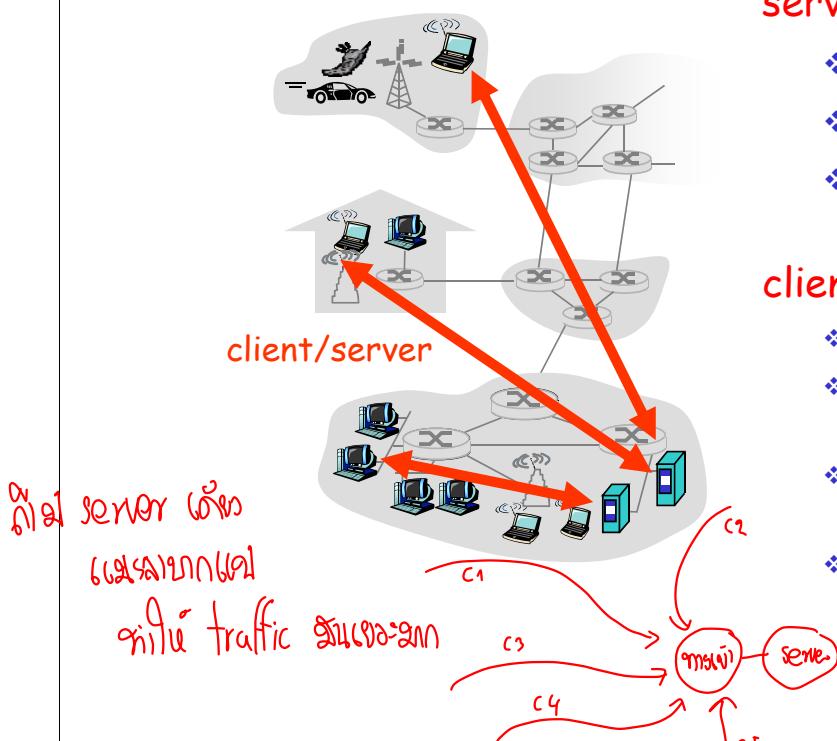
օճառչություն և լինգ

զորոշ եղանակներ

application layer → զանազան հայտ և դաշտավայրական պրոցես

2: Application Layer 8

Client-server architecture



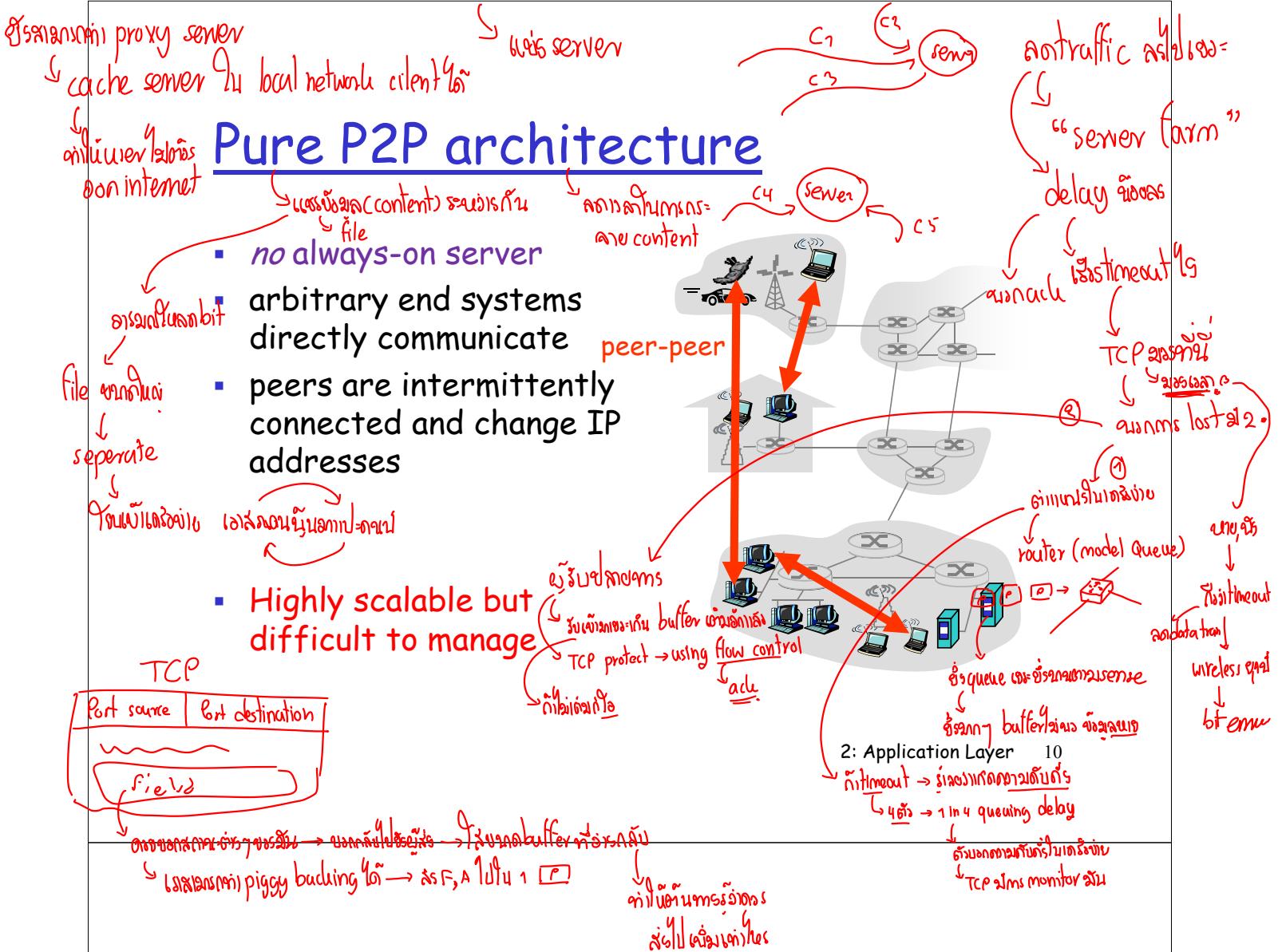
server:

- ❖ always-on host
- ❖ permanent IP address, port
- ❖ **server farms** for scaling

clients:

- ❖ communicate with server
- ❖ may be intermittently connected
- ❖ may have dynamic IP addresses
- ❖ do not communicate directly with each other

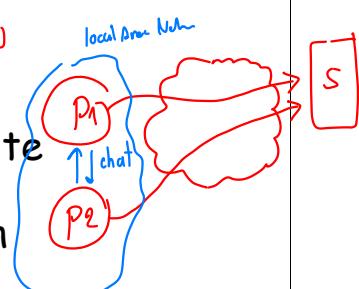
2: Application Layer 9



Hybrid of client-server and P2P

Skype

QoS model uses cache server



- voice-over-IP P2P application
- centralized server: finding address of remote party
- client-client connection: direct (not through server)

Instant messaging

- chatting between two users is P2P
- centralized service: client presence detection/location
 - user registers its IP address with central server when it comes online
 - user contacts central server to find IP addresses of buddies

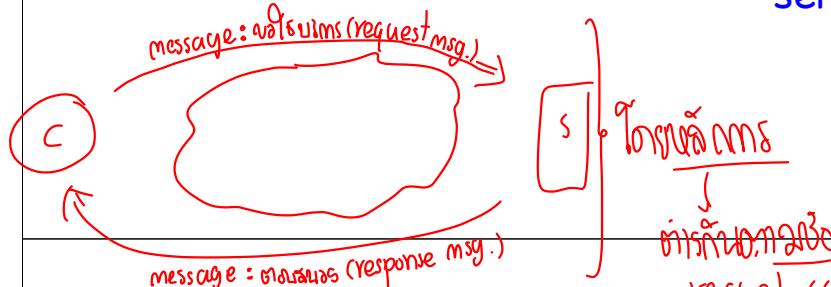
Application Layer

ពិសេសរាជការណ៍នៃអ្នកបើងបីជាសម្រាប់

Processes communicating

Process: program running within a host.

- within same host, two processes communicate using **inter-process communication** (defined by OS).
- processes in different hosts communicate by exchanging messages



Client process: process that initiates communication

Server process: process that waits to be contacted

- Note: applications with P2P architectures have client processes & server processes

លទ្ធផលនៃការងារ 2 រដ្ឋបាល

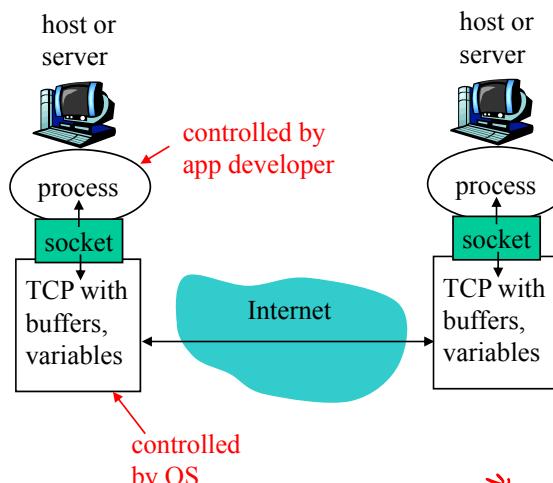
2: Application Layer 12

ទូរសព្ទ, ឯកសារ ការងារ

request, command
query, response

Sockets (ក្រសួងទំនាក់ទំនង layer តាមលេខ (transport))

- process sends/receives messages to/from its socket
- socket analogous to door
 - sending process shoves message out door
 - sending process relies on transport infrastructure on other side of door which brings message to socket at receiving process



- API: (1) choice of transport protocol; (2) ability to fix a few parameters (lots more on this later)

Application Programming Interface

2: Application Layer 13

ផល
គែង
ធ្វើ
និង

UDDI ឬ
WSDL ឬ
axis ឬ
JMS ឬ

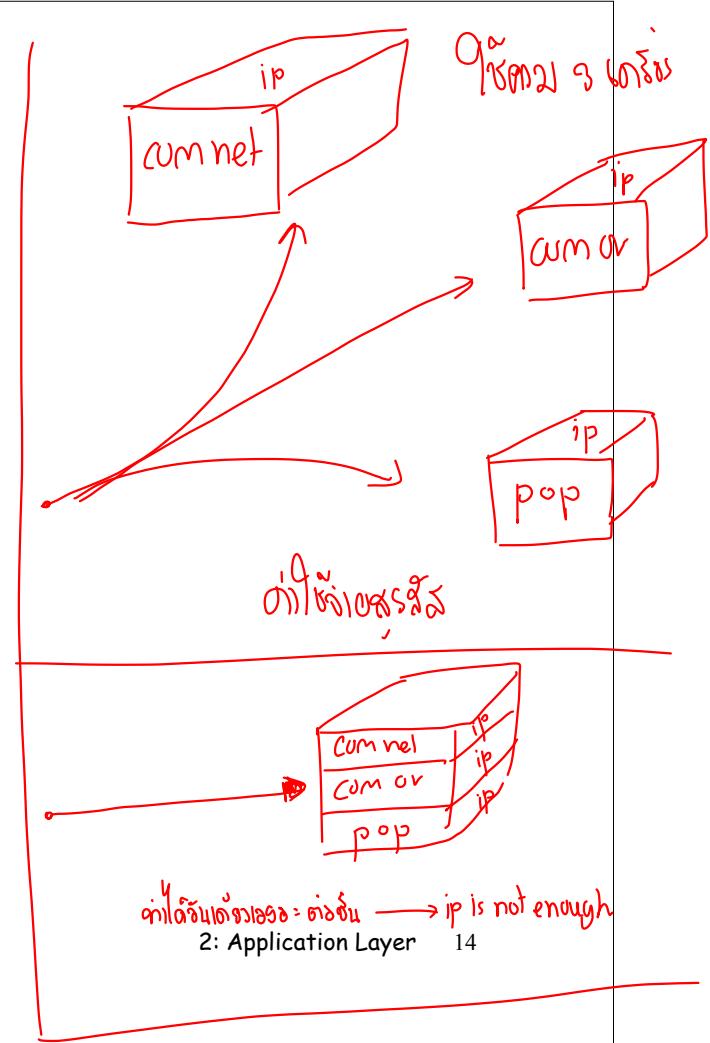
សារការថ្លែងការពី
រាយការណ៍នៃអ្នកបើងបីជាសម្រាប់
រាយការណ៍នៃអ្នកបើងបីជាសម្រាប់
រាយការណ៍នៃអ្នកបើងបីជាសម្រាប់

Addressing processes

- to receive messages, process must have **identifier**
- host device has unique 32-bit IP address
- **Q:** does IP address of host suffice for identifying the process?

↳ ip address has ~ port number
↓
server computer application process

~ 802.11 process ↳ is comnetwork
port



Addressing processes

- to receive messages, process must have **identifier**
- host device has unique 32-bit IP address
- **Q:** does IP address of host on which process runs suffice for identifying the process?
 - **A:** No, many processes can be running on same host
- **identifier** includes both **IP address** and **port numbers** associated with process on host.
- Example port numbers:
 - HTTP server: 80
 - Mail server: 25
- to send **HTTP message** to **gaia.cs.umass.edu** web server:
 - IP address: 128.119.245.12
 - Port number: 80
- more shortly...

message → request
→ reply } → protocol → application

App-layer protocol defines

- 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 processes send & respond to messages

Public-domain protocols: စာတမ်း

- defined in RFCs

- allows for interoperability

- e.g., HTTP, SMTP

Proprietary protocols: မြန်မာစွဲလုပ်ငန်း

- e.g., Skype

ပေးနှံမှု

method	Sp	URL	Sp	Version	Cr	If
Header field name	:		value	Cr	If	
	:					
Header field name	:		value	Cr	If	
Cr	If					
Entity Body						

16

ပုံစံတွေ၏ ဖျော့ချုပ်အကြောင်းရှင်းကို တော်းသွေး မြန်မာစွဲလုပ်ငန်း မြန်မာစွဲလုပ်ငန်း

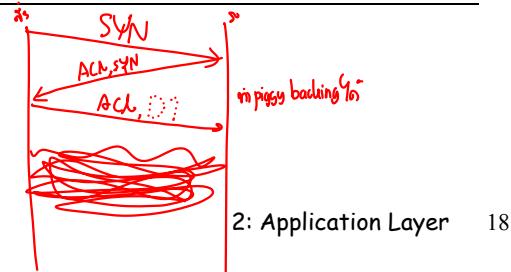
What transport service does an app need?

- ① Data loss → error and lost
 - some apps (e.g., audio) can tolerate some loss
 - other apps (e.g., file transfer, telnet) require 100% reliable data transfer
- ② Timing → ပုံစံတွေ၏ ပို့ဆောင်ရည်၊ ပို့ဆောင်ရည်
 - some apps (e.g., Internet telephony, interactive games) require low delay to be "effective"
- ③ Throughput → ပုံစံတွေ၏ ပို့ဆောင်ရည်
 - some apps (e.g., multimedia) require minimum amount of throughput to be "effective"
 - other apps ("elastic apps") make use of whatever throughput they get
- Security
 - Encryption, data integrity, ...

လုပ်ငန်းမြန်မာစွဲလုပ်ငန်း

Transport service requirements of common apps

Application	Data loss	Throughput	Time Sensitive
file transfer	no loss	elastic	no
e-mail	no loss	elastic	no
Web documents	no loss	elastic	no
real-time audio/video	loss-tolerant	audio: 5kbps-1Mbps video: 10kbps-5Mbps	yes, 100's msec
stored audio/video	loss-tolerant	same as above	yes, few secs
interactive games	loss-tolerant	few kbps up	yes, 100's msec
instant messaging	no loss	elastic	yes and no



Internet transport protocols services

TCP service:

- connection-oriented: setup required between client and server processes
- reliable transport between sending and receiving process
- flow control: sender won't overwhelm receiver
- congestion control: throttle sender when network overloaded
- does not provide: timing, minimum throughput guarantees, security

UDP service:

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

Q: why bother? Why is there a UDP?

Internet apps: application, transport protocols

App 服务

Application	Application layer protocol	Underlying transport protocol
e-mail	SMTP [RFC 2821]	TCP
remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
file transfer	FTP [RFC 959]	TCP
streaming multimedia	HTTP (eg Youtube), RTP [RFC 1889]	TCP or UDP
Internet telephony	SIP, RTP, proprietary (e.g., Skype)	typically UDP

RTP (Real-Time Application Protocol) ถูกแนะนำโดย RFC 3550

SIP (Session Initiation Protocol) RFC 3261

RTP (Real-time Transport Protocol) RFC 1889

2: Application Layer 20

Chapter 2: Application layer

- 2.1 Principles of network applications
 - app architectures
 - app requirements
- 2.2 Web and HTTP
- 2.4 Electronic Mail
 - SMTP, POP3, IMAP
- 2.5 DNS
- 2.6 P2P applications
- 2.7 Socket programming with TCP
- 2.8 Socket programming with UDP

Web and HTTP

First some jargon

- Web page consists of objects
- Object can be HTML file, JPEG image, Java applet, audio file,...
- Web page consists of base HTML-file which includes several referenced objects
- Each object is addressable by a URL (Uniform Resource Locator)
- Example URL:

www.someschool.edu/someDept/pic.gif

host name

path name

ຕົວຢ່າງໄປຕົ້ນໄດ້ມານັກສອນ

2: Application Layer 22



ຖຸນ ຫິນ
ຖຸນ ທີ່
ຖຸນ ສາ

Firefox - Basic HTML Sample Page

sheldongbrown.com/webs_sample.html

Most Visited Getting Started Suggested Sites Web Slice Gallery

COLNAGO SCIENCE AND PASSION COMBINED

A Simple Sample Web Page

By Sheldon Brown

Demonstrating a few HTML features

HTML is really a very simple language. It consists of ordinary text, with commands that are enclosed by "<" and ">" characters, or between an "&x" and a "&".

You don't really need to know much HTML to create a page, because you can copy bits of HTML from other pages that do what you want, then change the text!

This page shows on the left as it appears in your browser, and the corresponding HTML code appears on the right. The HTML commands are linked to explanations of what they do.

Line Breaks

HTML doesn't normally use line breaks for ordinary text. A white space of any size is treated as a single space. This is because the author of the page has no way of knowing the size of the reader's screen, or what size type they will have their browser set for.

If you want to put a line break at a particular place, you can use the "
" command, or, for a paragraph break, the "<p>" command, which will insert a blank line. The heading command ("<h1></h1>") puts a blank line above and below the heading text.

Starting and Stopping Commands

Most HTML commands come in pairs: for example, "<h1>" marks the beginning of a size 1 heading, and "</h1>" marks the end of it. The closing command is always the same as the opening command, except for the addition of the "&".

Modifiers are sometimes included along with the basic command, inside the opening command's < >. The modifier does not need to be repeated in the closing command.

EN 603 26/6/2556

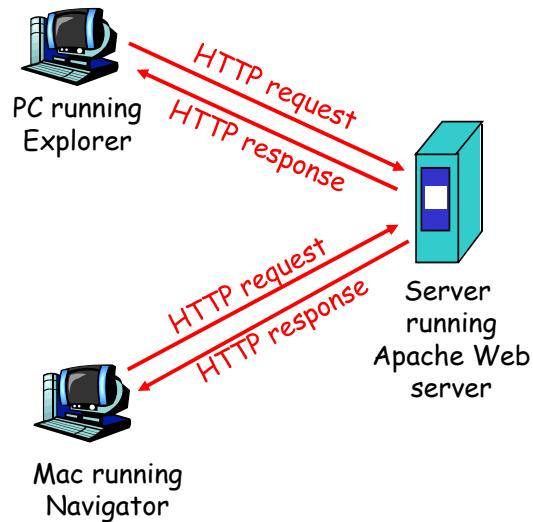
```
<html>
<head>
<title>Document Title</title>
</head>
<body>
.....
</body>
</html>
```

2: Application Layer 23

HTTP overview

HTTP: Hypertext Transfer Protocol

- Web's application layer protocol
- Client/Server Model
 - ❖ **Client:** browser that requests, receives, "displays" Web objects
 - ❖ **Server:** Web server sends objects in response to requests



Web browser : Internet Explorer, FireFox, Google Chrome

Web Server : Apache, Internet Information Service (IIS)

2: Application Layer 24

HTTP protocol ဆဲ ပေါ်၍ work မရအပ်ပါဘူး → နောက်လိုအပ်မယ့် သိမှတ်မယ့်

HTTP overview (continued)

Uses TCP:

- ① ▪ client initiates TCP connection (creates socket)

common port to server, port 80

server accepts TCP

register port 'connection from client'

- HTTP messages (application-layer protocol messages) exchanged between browser (HTTP client) and Web server (HTTP server)

- TCP connection closed

HTTP is "stateless"

- server maintains no information about past client requests

cheh?

HTTP မျှတ်၏

လိုအပ်

aside -

Protocols that maintain "state" are complex!

- past history (state) must be maintained
- if server/client crashes, their views of "state" may be inconsistent, must be reconciled

2: Application Layer 25

HTTP connections

၃၁၂ ချုပ်မှတ် စောင့်ရှုပုံ TCP → ၃၂၂၅၂

Nonpersistent HTTP

- At most one object is sent over a TCP connection.



1 obj ois 1 tcp connect

Persistent HTTP

- Multiple objects can be sent over single TCP connection between client and server.



1 connet အတွက် ရှုပုံများ
တိုက်ပါ

၂၀၁၇
၂၀၁၈
၂၀၁၉

2: Application Layer 26

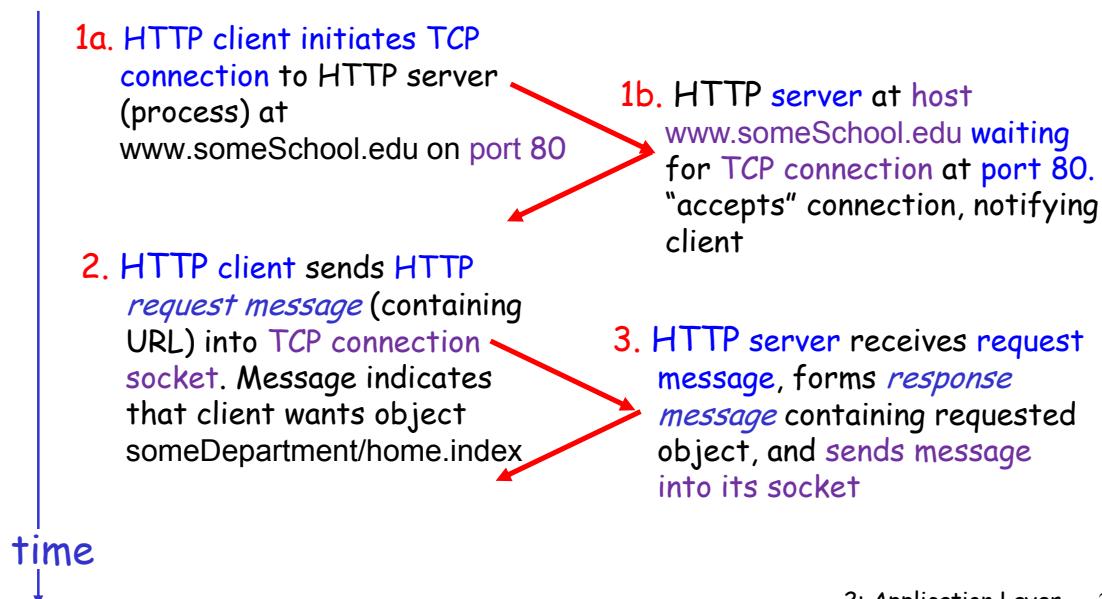
Nonpersistent HTTP

Suppose user enters URL

www.someSchool.edu/someDepartment/home.index

(contains text,
references to 10
jpeg images)

ဆောင်ရွက်မည့် | ⇒ ရှုပုံများ



2: Application Layer 27

Nonpersistent HTTP (cont.)

- time ↓
4. HTTP server closes TCP connection.
 5. HTTP client receives response message containing html file, displays html. Parsing html file, finds 10 referenced jpeg objects
 6. Steps 1-5 repeated for each of 10 jpeg objects
- A red arrow points from step 4 to step 5.

2: Application Layer 28

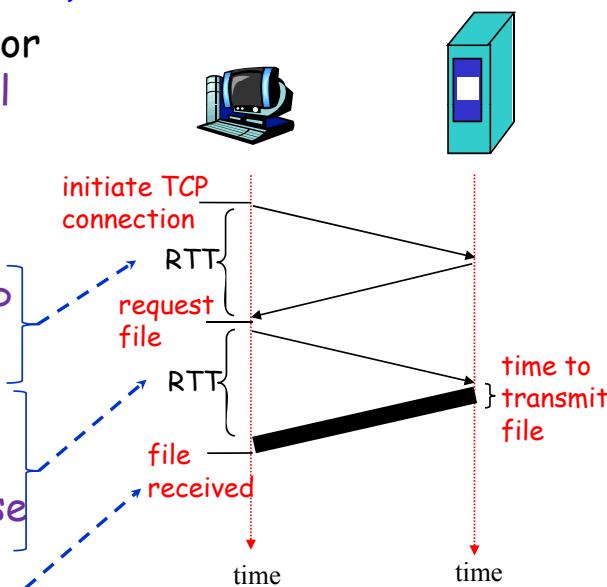
Non-Persistent HTTP: Response time

(RTT : Round Trip Time)

Definition of RTT: time for a small packet to travel from client to server and back.

Response time:

- one RTT to initiate TCP connection
 - one RTT for HTTP request and first few bytes of HTTP response to return
 - file transmission time
- total = 2RTT+transmit time**



2: Application Layer 29

Persistent HTTP

Nonpersistent HTTP issues:

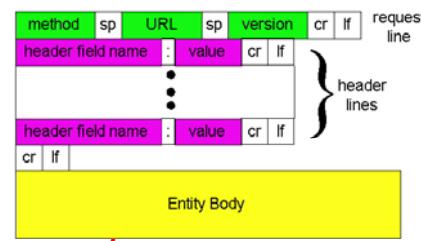
- requires 2 RTTs per object
- OS overhead for each TCP connection
- browsers often open parallel TCP connections to fetch referenced objects

Persistent HTTP

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

2: Application Layer 30

HTTP Request Message



- two types of HTTP messages: *request, response*
- **HTTP request message:**
 - ❖ ASCII (human-readable format)

request line
(GET, POST,
HEAD commands)

GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
User-agent: Mozilla/4.0
Connection: close
Accept-language: fr

status line

టిప్పణించుట లో ఇది కషాయి

header lines

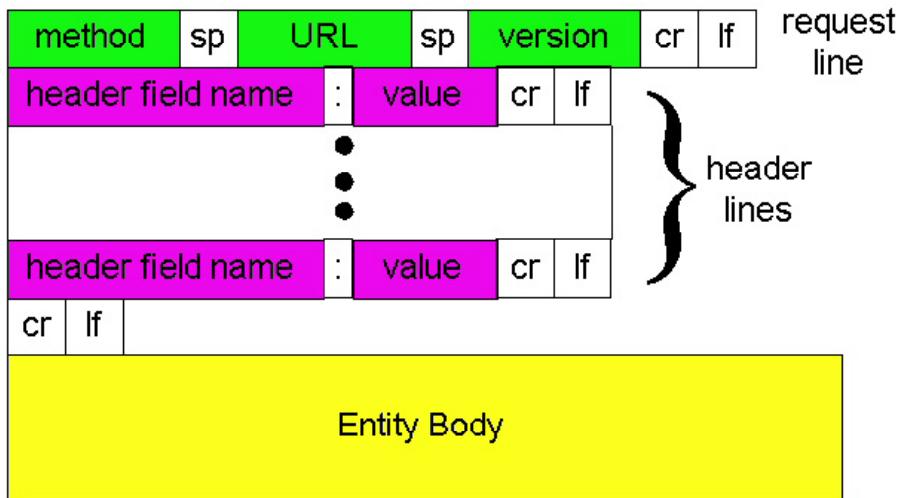
Carriage return
line feed
indicates end
of message

(extra carriage return, line feed)

2: Application Layer 31

HTTP Request Message: general format

WDPN }



2: Application Layer 32

Uploading form input

Post method:

- Web page often includes **form input**
- **Input is uploaded to server in entity body**

URL method:

- Uses **GET** method
- **Input is uploaded in URL field of request line:**

www.somesite.com/animalsearch?monkeys&banana

GET www.somesite.com/animalsearch?monkeys&banana HTTP/1.1

2: Application Layer 33

Method types

HTTP/1.0 (RFC 1945)

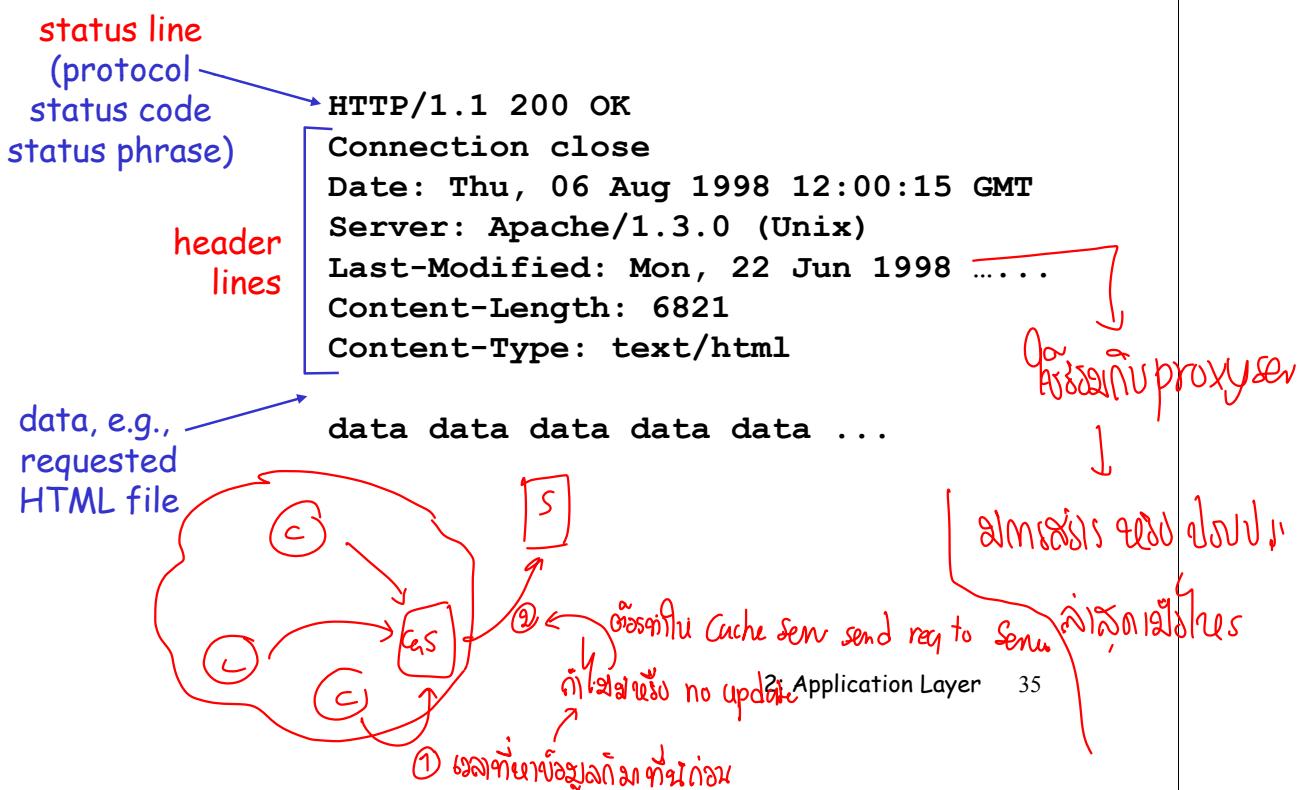
- GET
- POST
- HEAD
 - ❖ asks server to leave requested object out of response

HTTP/1.1 (RFC 2616)

- GET, POST, HEAD
- PUT
 - ❖ uploads file in entity body to path specified in URL field
- DELETE
 - ❖ deletes file specified in the URL field

2: Application Layer 34

HTTP response message



2: Application Layer 35

HTTP response status codes

In first line in server->client response message.

A few sample codes:

200 OK

- ❖ request succeeded, requested object later in this message

301 Moved Permanently

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

400 Bad Request

- ❖ request message not understood by server

404 Not Found

- ❖ requested document not found on this server

505 HTTP Version Not Supported

2: Application Layer 36

Trying out HTTP (client side) for yourself

1. Telnet to your favorite Web server:

telnet cis.poly.edu 80

Opens TCP connection to port 80 (default HTTP server port) at cis.poly.edu. Anything typed in sent to port 80 at cis.poly.edu

2. Type in a GET HTTP request:

GET /~ross/ HTTP/1.1
Host: cis.poly.edu

By typing this in (hit carriage return twice), you send this minimal (but complete) GET request to HTTP server

3. Look at response message sent by HTTP server!

working both but secure

2: Application Layer 37

Telnet unix.kmitl.ac.th

```

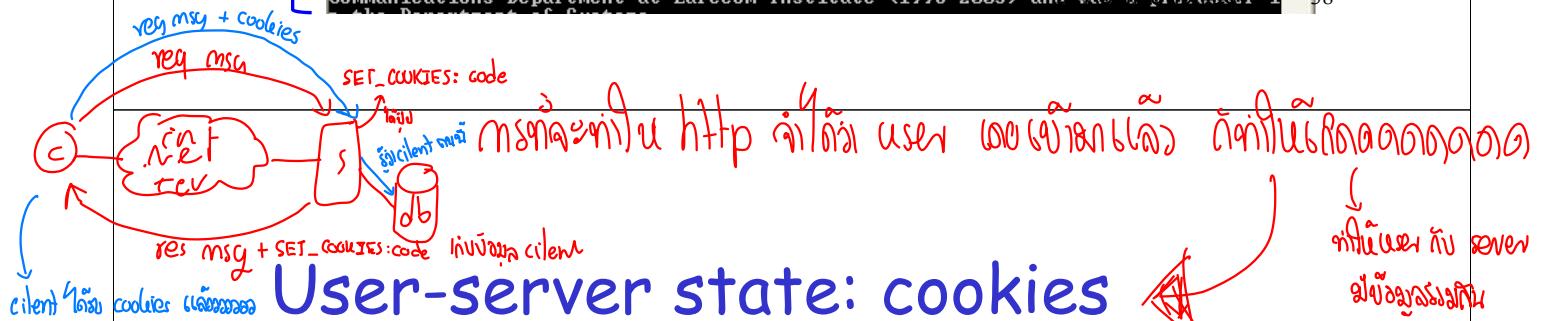
$ telnet cis.poly.edu 80
Trying 128.238.32.126...
Connected to cis.poly.edu.
Escape character is '^J'.
GET /~ross/ HTTP/1.1
Host: cis.poly.edu

HTTP/1.0 200 OK
Date: Thu, 07 Jul 2005 17:50:13 GMT
Server: Apache/1.2.5
Last-Modified: Thu, 20 Jan 2005 19:19:17 GMT
ETag: "15834-2748-41f00435"
Content-Length: 10056
Accept-Ranges: bytes
Content-Type: text/html
X-Cache: MISS from proxy.net.kmitl.ac.th
Connection: close

<html>
<head>
<title>Keith Ross's Homepage</title>
</head>
<body>

<br>&nbsp;
<h2>
Keith W. Ross</h2>
<h3> Leonard J. Shustek Professor of Computer Science</h3>
<p>
Department of Computer and Information Science
<br>Polytechnic University
<br>8ix MetroTech Center
<br>Brooklyn, NY 11201
<br>phone: 718-260-3859
<br>fax: 718-260-3609
<p>e-mail:&nbsp; <a href="mailto:ross@poly.edu">ross@poly.edu</a>
<h3>
Biosketch</h3>
Professor Ross is the <a href="http://www.poly.edu/news/Keith_Ross_news.cfm">Leonard
J. Shustek</a> Chair Professor in Computer Science</a> at Polytechnic
University. Before joining
Polytechnic University, he was a professor in the Multimedia
Communications Department at Eurecom Institute (1998-2003) and was a professor i

```



Many major Web sites
use cookies

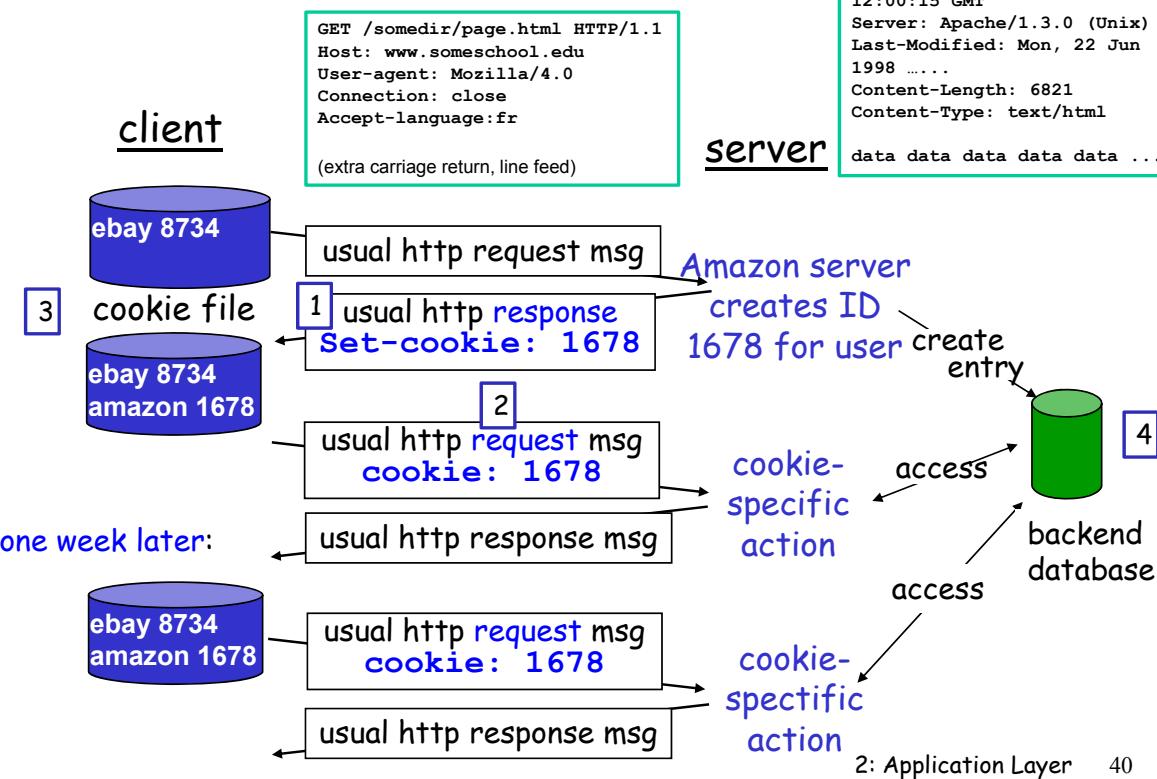
Four components:

- 1) cookie header line of
HTTP response message
- 2) cookie header line in
HTTP request message
- 3) cookie file kept on
user's host, managed by
user's browser
- 4) back-end database at
Web site

Example:

- Susan always access Internet always from PC
- visits specific e-commerce site for first time
- when initial HTTP requests arrives at site, site creates:
 - ❖ unique ID
 - ❖ entry in backend database for ID

Cookies: keeping "state" (cont.)



Cookies (continued)

What cookies can bring:

- authorization
- shopping carts
- recommendations
- user session state (Web e-mail)

Cookies and privacy:

- cookies permit sites to learn a lot about you
- you may supply name and e-mail to sites

Privacy *
1 sensitive data

How to keep "state":

- protocol endpoints: maintain state at sender/receiver over multiple transactions
- cookies: http messages carry state

ex. បិទសាខាអំពីចំណែកអាជីវកម្មនៃ super market នៅក្នុង

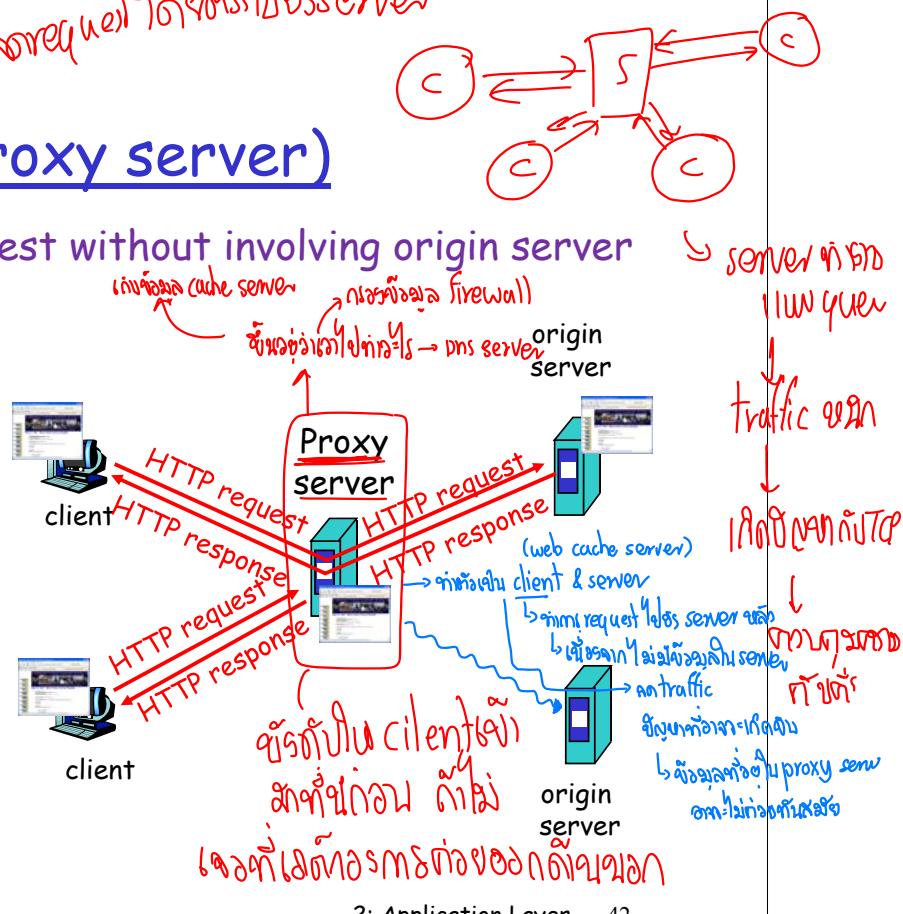
ជីវកម្ម

រួចរាល់ពីសេវាបណ្តុះបណ្តាល

Web caches (proxy server)

Goal: satisfy client request without involving origin server

- user sets browser: Web accesses via cache
- browser sends all HTTP requests to cache
 - ❖ object in cache: cache returns object
 - ❖ else cache requests object from origin server, then returns object to client



2: Application Layer 42

More about Web caching

- cache acts as both client and server
- typically cache is installed by ISP (university, company, residential ISP)

Why Web caching?

- reduce response time for client request
- reduce traffic on an institution's access link.
- Internet dense with caches: enables "poor" content providers to effectively deliver content (but so does P2P file sharing)

2: Application Layer 43

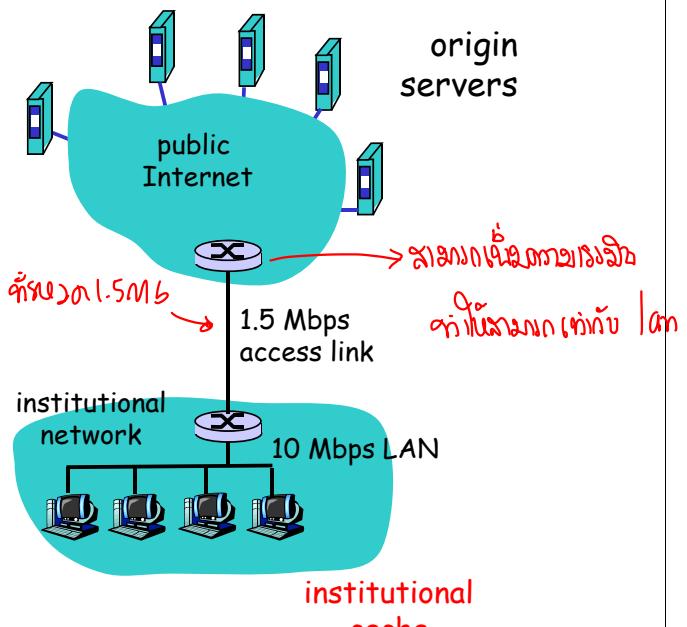
Caching example

Assumptions

- average object size = 100,000 bits 0.1 Mb
- average request rate from institution's browsers to origin servers = 15 requests/sec $\rightarrow 0.15 \text{ Mb/s}$
- delay from institutional router to any origin server and back to router = 2 sec

Consequences

- ~~utilization~~ ~~bandwidth~~
- utilization on LAN = 15%
 - utilization on access link = 100%
 - total delay = Internet delay + access delay + LAN delay
 $= 2 \text{ sec} + \text{minutes} + \text{milliseconds}$



2: Application Layer 44

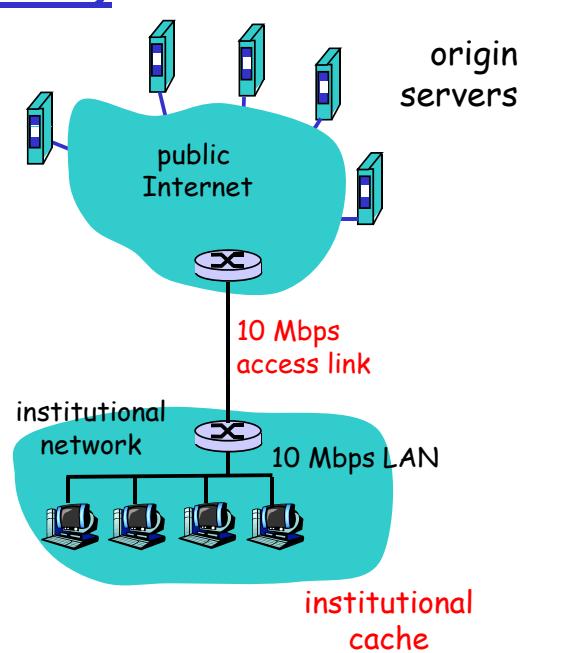
Caching example (cont)

possible solution

- increase bandwidth of access link to, say, 10 Mbps

consequence

- utilization on LAN = 15%
- utilization on access link = 15%
- Total delay = Internet delay + access delay + LAN delay
 $= 2 \text{ sec} + \text{msecs} + \text{msecs}$
- often a costly upgrade



2: Application Layer 45

Caching example (cont)

possible solution: install cache

- suppose hit rate is 0.4

consequence

- 40% requests will be satisfied almost immediately

- 60% requests satisfied by origin server

- utilization of access link reduced to 60%, resulting in negligible delays (say 10 msec)

hit rate \Rightarrow

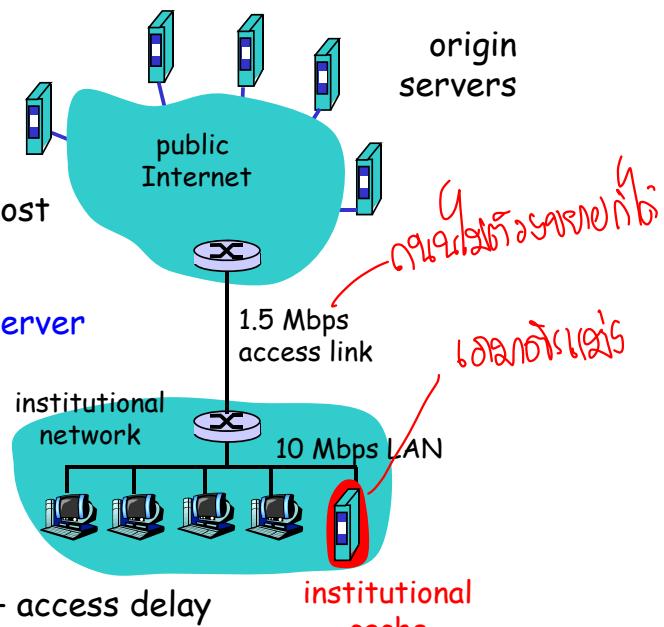
origin servers
public Internet

institutional web cache server

- total avg delay = Internet delay + access delay + LAN delay

$$\hookrightarrow \text{hit rate is } 50\% = 0.6 * (2.01) \text{ secs} + 0.4 * \text{milliseconds} < 1.4 \text{ secs}$$

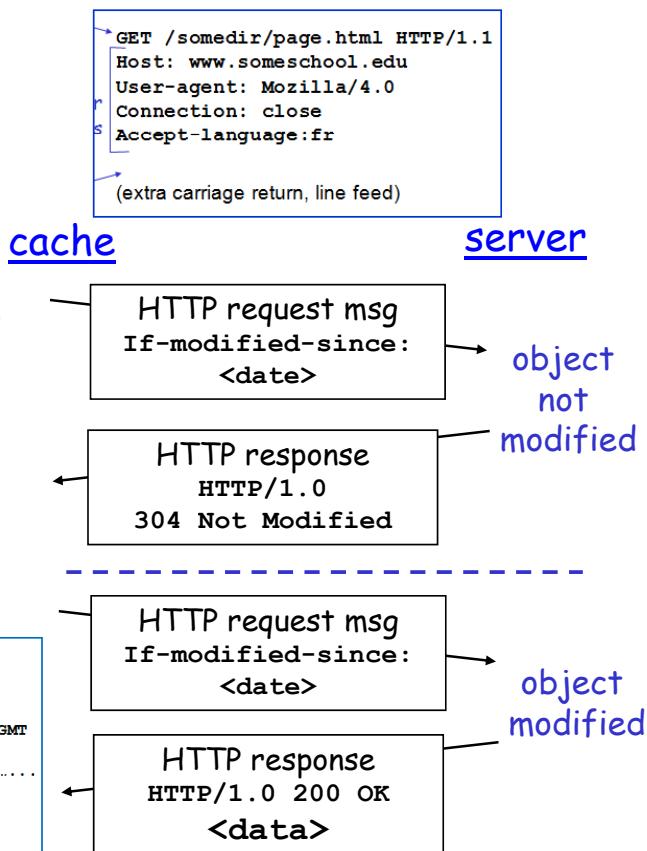
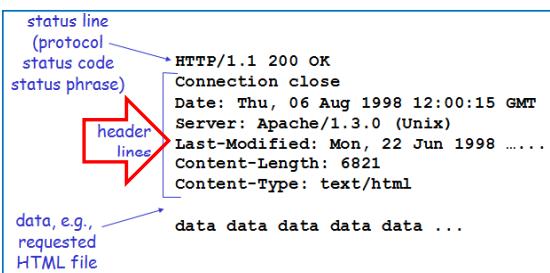
origin servers



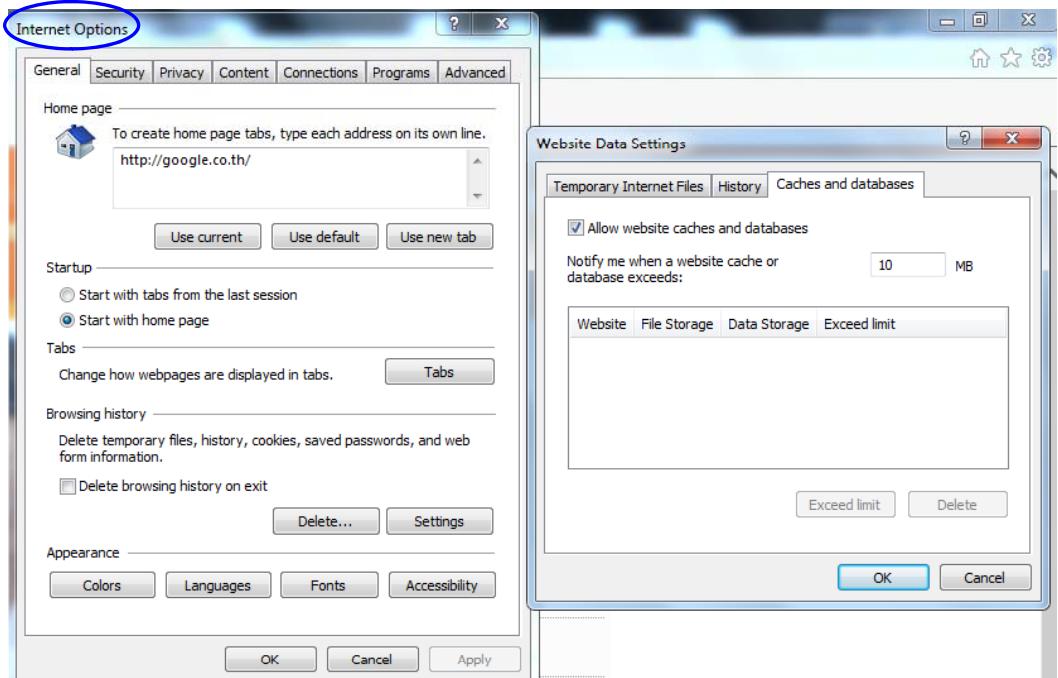
2: Application Layer 46

Conditional GET

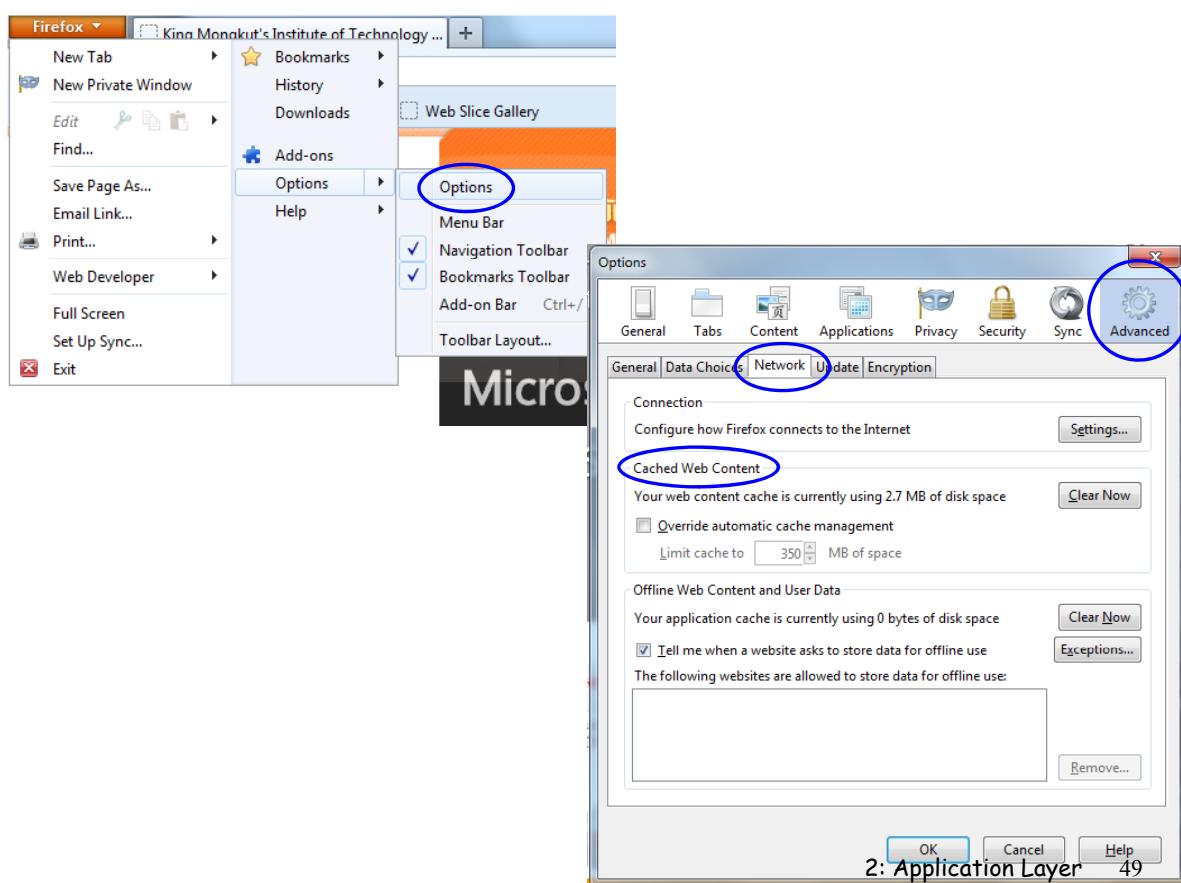
- Goal: don't send object if cache has up-to-date cached version
- cache: specify date of cached copy in HTTP request
 - If-modified-since: <date>
- server: response contains no object if cached copy is up-to-date:
 - HTTP/1.0 304 Not Modified



2: Application Layer 47



2: Application Layer 48

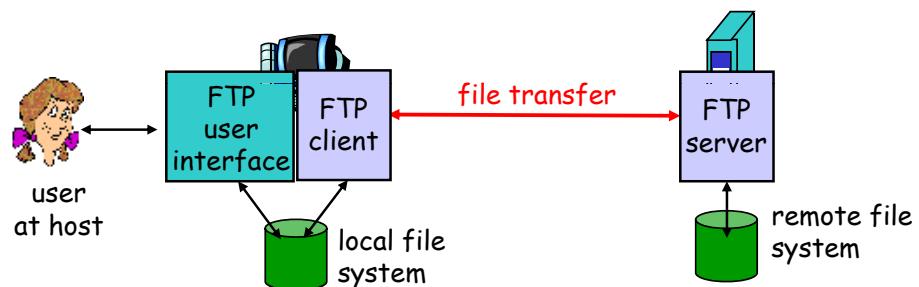


Chapter 2: Application layer

- 2.1 Principles of network applications
- 2.2 Web and HTTP
- 2.3 FTP
- 2.4 Electronic Mail
 - SMTP, POP3, IMAP
- 2.5 DNS
- 2.6 P2P applications
- 2.7 Socket programming with TCP
- 2.8 Socket programming with UDP
- 2.9 Building a Web server

2: Application Layer 50

FTP: the File Transfer Protocol

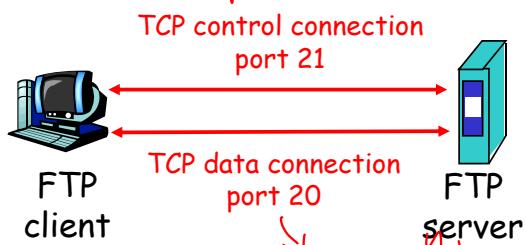


- transfer file to/from remote host
- client/server model
 - ❖ *client*: side that initiates transfer (either to/from remote)
 - ❖ *server*: remote host
- ftp: RFC 959
- ftp server: port 21

2: Application Layer 51

FTP: separate control, data connections

- ❑ FTP client contacts FTP server at port 21, TCP is transport protocol
- ❑ client authorized over control connection
- ❑ client browses remote directory by sending commands over control connection.
- ❑ when server receives file transfer command, server opens 2nd TCP connection (for file) to client
- ❑ after transferring one file, server closes data connection.



- ❑ server opens another TCP data connection to transfer another file.
- ❑ control connection: "out of band"
- ❑ FTP server maintains "state": current directory, earlier authentication

server maintains state
close control
(2/2)

2: Application Layer 52

FTP commands, responses

Sample commands:

- sent as ASCII text over control channel
- **USER username**
- **PASS password**
- **LIST** return list of files in current directory
- **RETR filename** retrieves (gets) file
- **STOR filename** stores (puts) file onto remote host

Sample return codes

- status code and phrase (as in HTTP)
- **331 Username OK, password required**
- **125 data connection already open; transfer starting**
- **425 Can't open data connection**
- **452 Error writing file**

2: Application Layer 53

เข้าสู่ระบบผ่านทาง password

เข้าสู่ระบบผ่านทาง anonymous ftp

Username: anonymous
password: enter

(ก) remote ที่ตั้ง

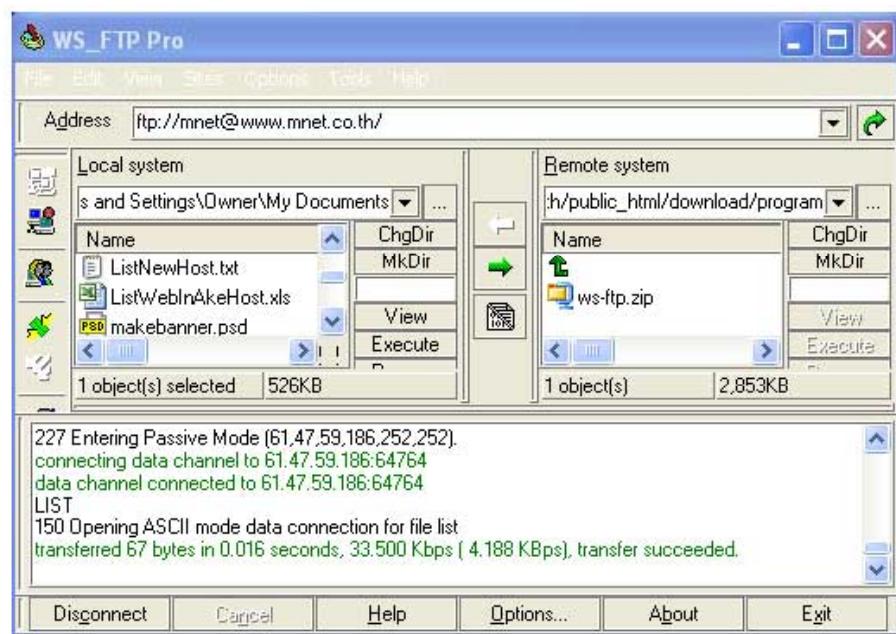
```

C:\Windows\system32\cmd.exe
C:\Users\Skyline>ftp
ftp> o unix.kmitl.ac.th
Connected to neutrino.crsc.kmitl.ac.th.
220-Welcome to KMITL Unix and FTP Services..
220
User <neutrino.crsc.kmitl.ac.th:<none>> ktsakcha
331 Please specify the password.
Password:
230 Login successful.
ftp> ?
Commands may be abbreviated. Commands are:
!
?           delete      literal      prompt      send
append      debug       ls           put         status
ascii       disconnect   mdelete     pwd         trace
bell        get         mdir        quit        type
binary      glob        mget        quote       user
bye         hash        mkdir      recv        verbose
cd          help        mput        rename
close       lcd         open        rmdir
ftp>
ftp> bye
221 Goodbye.

C:\Users\Skyline>

```

2: Application Layer 54



ดาวน์โหลดโปรแกรม WS_FTP Pro

2: Application Layer 55

C:\WINDOWS\system32\command.com

```

C:\DOCUMENTUME\AGNUN>ftp ftp.internic.net
Connected to ftp.internic.net.
220 Internic FTP Server
User <ftp.internic.net:(none)>: anonymous
331 Anonymous login ok, send your complete email address as your password
Password:
230 Anonymous access granted, restrictions apply
ftp> pwd
257 "/" is the current directory
ftp> ls -al
200 PORT command successful
150 Opening ASCII mode data connection for file list
...
domain
226 Transfer complete
ftp: 15 bytes received in 0.00Seconds 15000.00Kbytes/sec.
ftp> cd domain
250 CWD command successful
ftp> ls -al
200 PORT command successful
150 Opening ASCII mode data connection for file list
db.cache.md5
in-addr.arpa.gz
..
arpa.zone.gz.sig
ip6.arpa.gz.md5
root.zone.gz.sig
in-addr.arpa.gz.md5
.listing
named.root.md5
db.cache
db.cache.sig
edu.zone.gz.sig
named.cache.md5
ip6.arpa.md5
named.cache
edu.zone.gz.md5
arpa.zone.gz
arpa.zone.gz.md5
named.cache.sig
root.zone.gz
named.root.sig
edu.zone.gz
ip6.arpa.gz
INTERNIC_ROOT_ZONE.signatures.asc
root.zone.gz.md5
named.root
INTERNIC_ROOT_ZONE.signatures
226 Transfer complete
ftp: 441 bytes received in 0.00Seconds 441000.00Kbytes/sec.
ftp> bye
221 Goodbye.

C:\DOCUMENTUME\AGNUN>

```

2: Application Layer 56

Example Anonymous FTP Servers

C:\WINDOWS\system32\command.com

```

C:\DOCUMENTUME\AGNUN>ftp ftp.nectec.or.th
Connected to ftp.nectec.or.th.
220 Welcome to NECTEC FTP service.
User <ftp.nectec.or.th:(none)>: anonymous
331 Please specify the password.
Password:
230 Login successful.
ftp>
ftp> pwd
257 "/"
ftp> ls -al
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
drwxr-xr-x 15 0 0 4096 Oct 30 2008 .
drwxr-xr-x 15 0 0 4096 Oct 30 2008 ..
drwxr-xr-x 6 0 0 4096 Oct 28 2007 .
drwxr-xr-x 13 0 0 4096 Aug 07 2008 .
drwxr-xr-x 2 0 0 4096 Dec 07 2007 .proc
drwxr-xr-x 3 0 0 4096 Aug 07 2004 .rsync
lrwxrwxrwx 1 0 0 11 Mar 13 2007 NECTEC -> .a/1/NECTEC
d--x--x--x 2 0 0 4096 Mar 13 2007 bin
drwxr-xr-x 2 0 0 4096 Mar 23 2007 bsd-distributions
drwxr-xr-x 2 0 0 4096 Mar 13 2007 etc
lrwxrwxrwx 1 0 0 13 Oct 28 2007 index.html -> .a/index.html
tnl
lrwxrwxrwx 1 0 0 9 Mar 13 2007 info -> .a/1/info
drwxr-xr-x 2 0 0 4096 Mar 13 2007 lib
drwxr-xr-x 2 0 0 4096 Oct 28 2007 linux-distributions
drwxr-xr-x 3 0 0 4096 Oct 28 2007 linux-documents
drwxr-xr-x 2 0 0 4096 Oct 28 2007 linux-softwares
drwxr-xr-x 2 0 0 4096 Aug 22 2008 pub
drwxr-xr-x 2 0 0 4096 Oct 30 2007 sites
226 Directory send OK.
ftp: 1210 bytes received in 3.44Seconds 0.35Kbytes/sec.
ftp> quit
221 Goodbye.

C:\DOCUMENTUME\AGNUN>

```

ftp.nectec.or.th

ftp.kmit.ac.th

C:\WINDOWS\system32\command.com

```

C:\DOCUMENTUME\AGNUN>ftp ftp.kmit.ac.th
Connected to green20.crc.kmit.ac.th.
220 Welcome to our FTP service.
User <green20.crc.kmit.ac.th:(none)>: Anonymous
331 Please specify the password.
Password:
230 Login successful.
ftp>
ftp> pwd
257 "/"
ftp>
ftp> ls -al
200 PORT command successful. Consider using PASV.
150 Here comes the directory listing.
drwxr-xr-x 2 0 0 4096 Aug 29 2007 .
drwxr-xr-x 2 0 0 4096 Aug 29 2007 ..
226 Directory send OK.
ftp: 119 bytes received in 0.00Seconds 119000.00Kbytes/sec.
ftp>
ftp> bye
221 Goodbye.

C:\DOCUMENTUME\AGNUN>

```

Chapter 2: Application layer

- 2.1 Principles of network applications
- 2.2 Web and HTTP
- 2.3 FTP
- 2.4 Electronic Mail
 - SMTP, POP3, IMAP
- 2.5 DNS
- 2.6 P2P applications
- 2.7 Socket programming with TCP
- 2.8 Socket programming with UDP

2: Application Layer 59

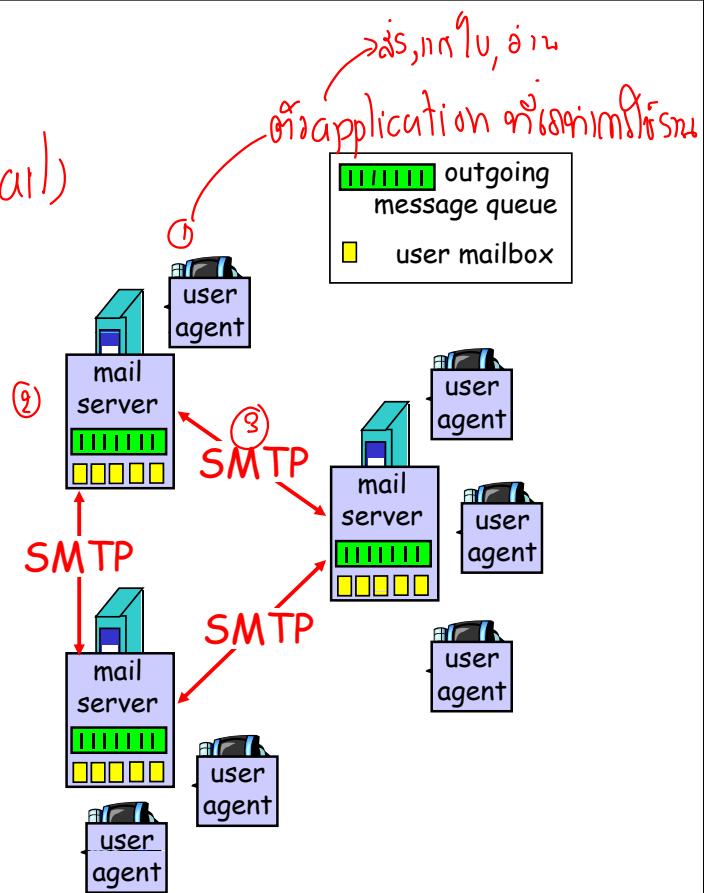
Electronic Mail (email)

Three major components:

- ① ○ User Agents
- ② ○ Mail Servers
- ③ ○ Simple Mail Transfer Protocol: SMTP (RFC 2821)

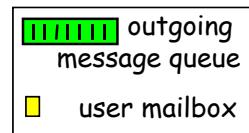
User Agent

- a.k.a. "Mail Reader"
- Composing, Editing, Reading Mail Messages
- e.g., Eudora, Outlook, elm, Mozilla Thunderbird
- outgoing, incoming messages stored on server



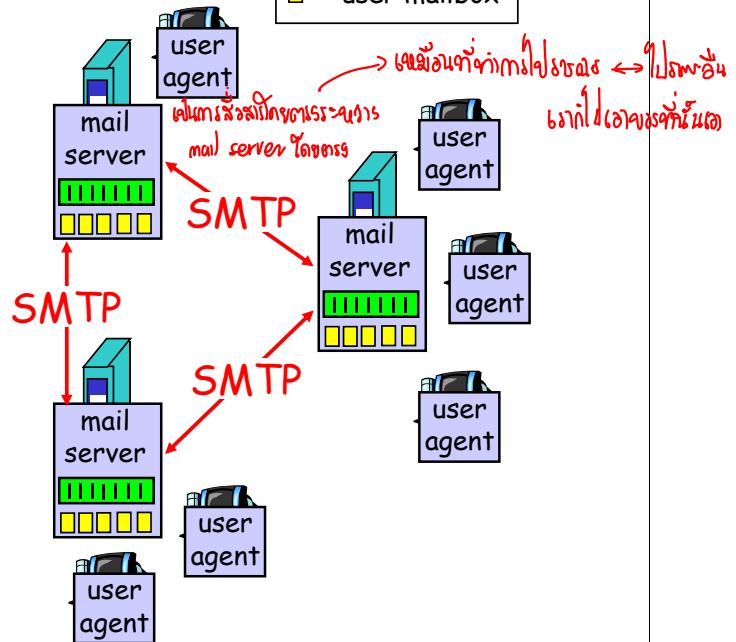
2: Application Layer 60

Electronic Mail: mail servers



Mail Servers

- mailbox contains incoming messages for user
- message queue of outgoing (to be sent) mail messages
- SMTP protocol between mail servers to send email messages
 - "Client": sending mail server
 - "Server": receiving mail server



2: Application Layer 61

Electronic Mail: SMTP [RFC 2821]

- ① ต่อ connection ② ต่อ msg ③ ปิด ; ต่อ connection
- uses TCP to reliably transfer email message from client to server, port 25
 - direct transfer: sending server to receiving server
 - three phases of transfer
 - handshaking (greeting)
 - transfer of messages
 - closure
 - command/response interaction
 - commands: ASCII text แบบเดียว
 - response: status code and phrase
 - messages must be in 7-bit ASCII

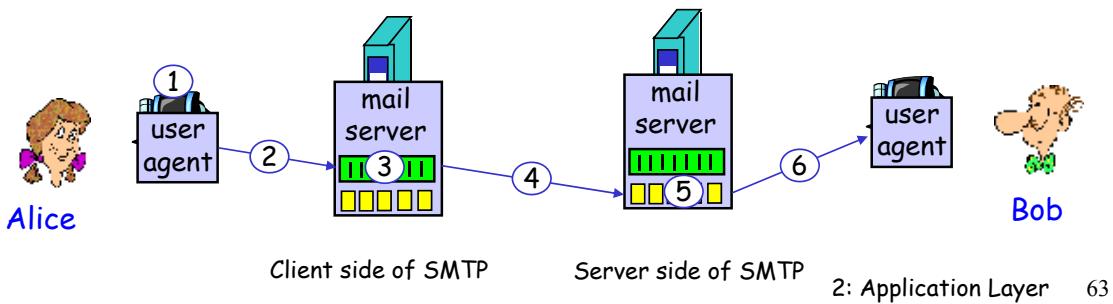
ถ้าต้องการ

2: Application Layer 62

→ ସଂସାଧନ

Scenario: Alice sends message to Bob

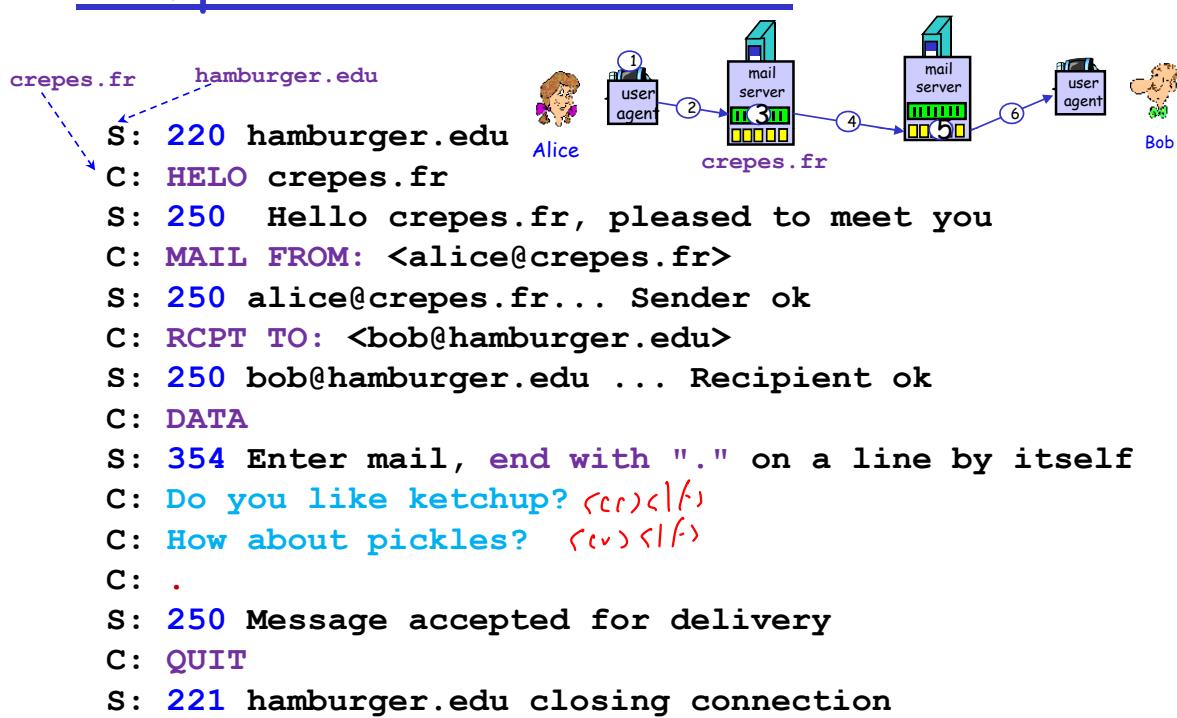
- 1) Alice uses user agent (UA) to compose message and "to" bob@someschool.edu
- 2) Alice's UA sends message to her mail server; message placed in message queue
- 3) Client side of SMTP opens TCP connection with Bob's mail server
- 4) SMTP client sends Alice's message over the TCP connection
- 5) Bob's mail server places the message in Bob's mailbox
- 6) Bob invokes his user agent to read message



2: Application Layer 63

→ କେବେ ମିଳିବାରେ ଏବଂ କେବେ ନିର୍ଦ୍ଦିତ ହୁଏଗଲା ?

Sample SMTP interaction



2: Application Layer 64

Try SMTP interaction for yourself:

- **telnet servername 25**
- see 220 reply from server
- enter HELO, MAIL FROM, RCPT TO, DATA, QUIT commands

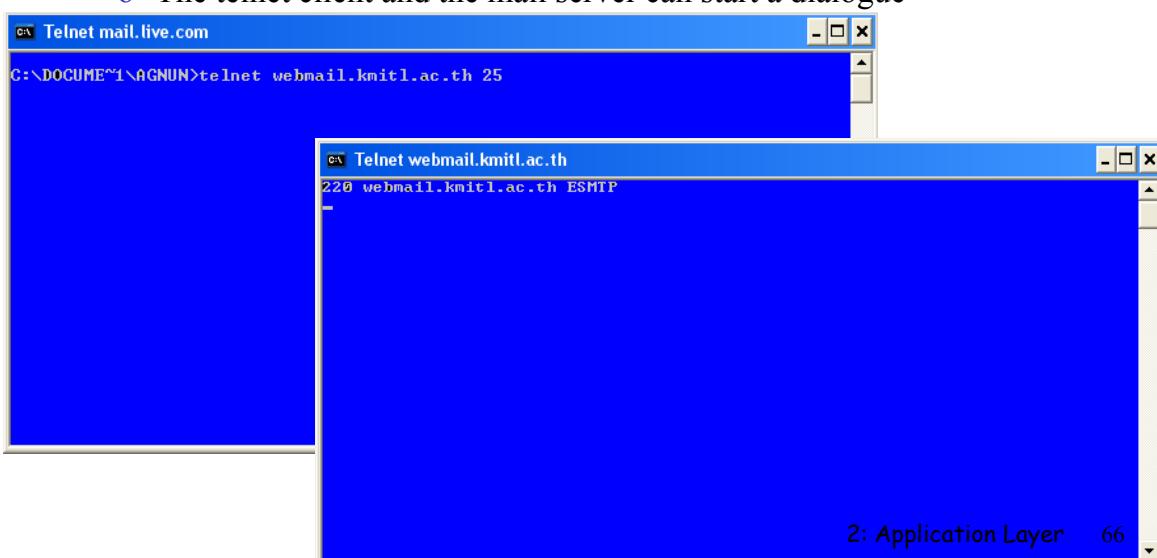
above lets you send email without using email client
(reader)

ຕາມນີ້ telnet ຍັງເປັດໄວ້

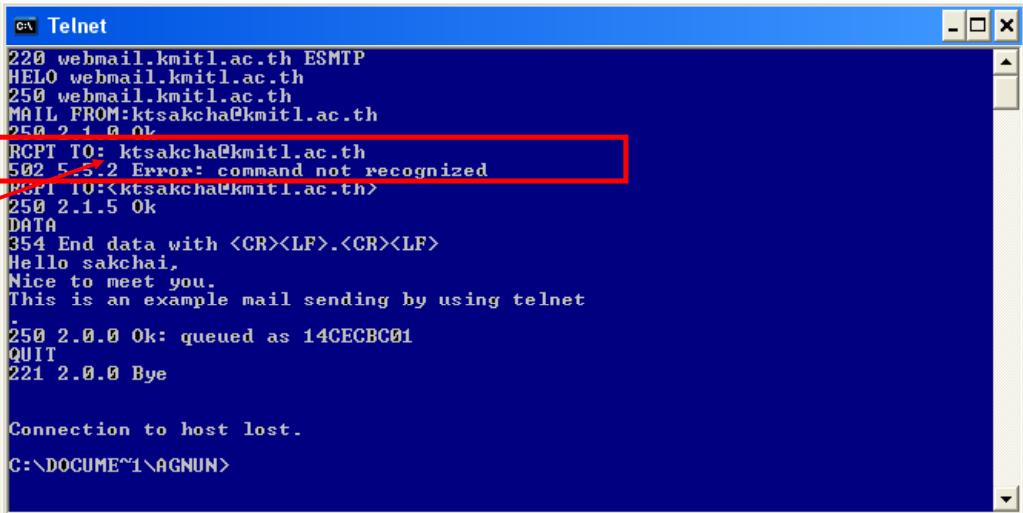
2: Application Layer 65

An Example SMTP Session

- How to connect to an SMTP server?
 - telnet servername 25
 - A TCP connection gets established over port number 25
 - The telnet client and the mail server can start a dialogue



An Example SMTP Session (cont.)



The screenshot shows a Telnet session window titled "Telnet". The session content is as follows:

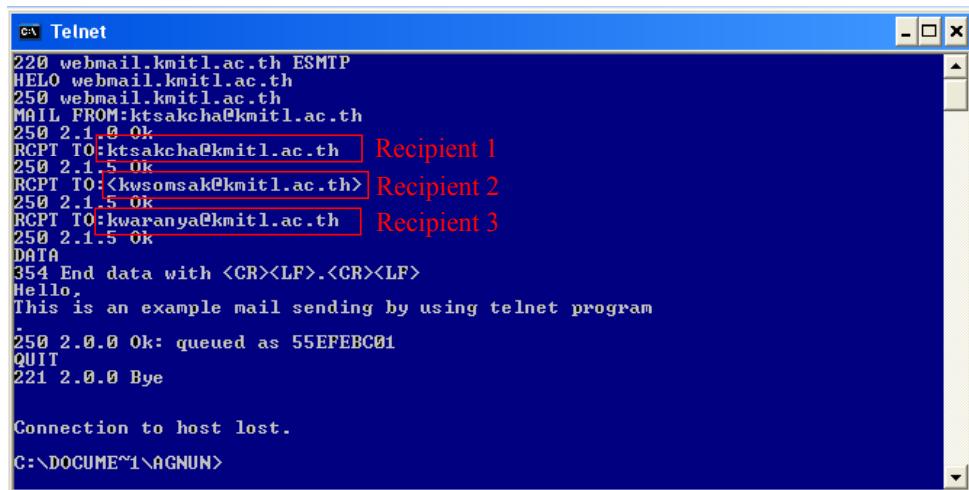
```
220 webmail.kmitl.ac.th ESMTP
HELO webmail.kmitl.ac.th
250 webmail.kmitl.ac.th
MAIL FROM:ktsakcha@kmitl.ac.th
250 2.1.0 Ok
RCPT TO: ktsakcha@kmitl.ac.th
502 5.5.2 Error: command not recognized
RCPT TO:<ktsakcha@kmitl.ac.th>
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
Hello sakchai,
Nice to meet you.
This is an example mail sending by using telnet
.
250 2.0.0 Ok: queued as 14CECB01
QUIT
221 2.0.0 Bye

Connection to host lost.
C:\DOCUMENTS\AGNUN>
```

A red box highlights the line "RCPT TO: ktsakcha@kmitl.ac.th" and the error message "502 5.5.2 Error: command not recognized". A red arrow points from the word "error" to this highlighted area.

2: Application Layer 67

An Example SMTP Session (cont.)



The screenshot shows a Telnet session window titled "Telnet". The session content is as follows:

```
220 webmail.kmitl.ac.th ESMTP
HELO webmail.kmitl.ac.th
250 webmail.kmitl.ac.th
MAIL FROM:ktsakcha@kmitl.ac.th
250 2.1.0 Ok
RCPT TO:ktsakcha@kmitl.ac.th Recipient 1
250 2.1.5 Ok
RCPT TO:<kusomsak@kmitl.ac.th> Recipient 2
250 2.1.5 Ok
RCPT TO:<kwaranya@kmitl.ac.th> Recipient 3
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
Hello,
This is an example mail sending by using telnet program
.
250 2.0.0 Ok: queued as 55EFEB01
QUIT
221 2.0.0 Bye

Connection to host lost.
C:\DOCUMENTS\AGNUN>
```

Three recipient addresses are listed in the RCPT TO command: "Recipient 1", "Recipient 2", and "Recipient 3". Each recipient address is enclosed in a red box.

Sending email to several recipients

2: Application Layer 68

SMTP: final words

- 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 message
- SMTP: multiple objects sent in multipart message

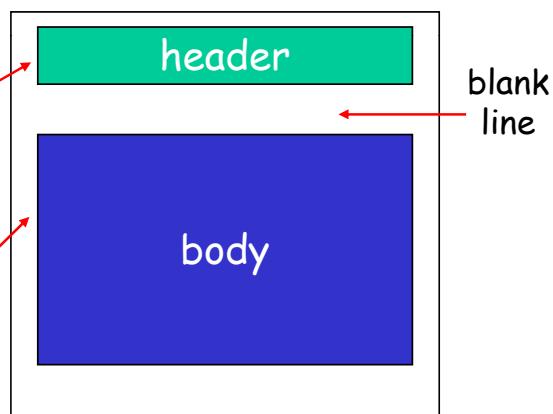
2: Application Layer 69

Mail Message Format

SMTP (RFC 2821) : protocol for exchanging email messages

RFC 822: standard for text message format:

- header lines, e.g.,
 - ❖ To:
 - ❖ From:
 - ❖ Subject:*different from SMTP commands!*
- body
 - ❖ the "message", ASCII characters only

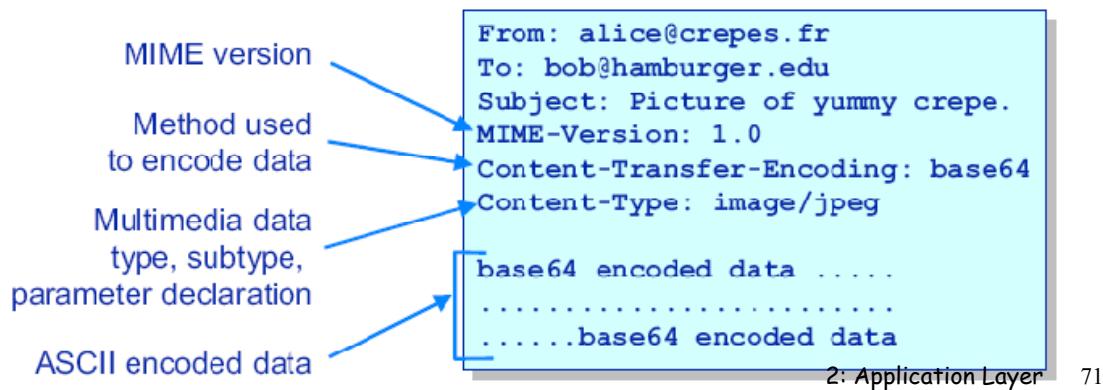


Mail Message Format

MIME – Multipurpose Internet Mail Extensions

(RFC 2045, 2056)

- o Multimedia mail extensions
- o SMTP requires all data to be *7-bit ASCII characters*
 - o All non-ASCII data must be encoded as ASCII strings
- o Additional lines in the message header declare MIME content type



71

MIME types and subtypes

Type	Subtype	Description
Text	Plain html	Unformatted Text Html format
Image	Gif Jpeg bmp	Still picture in GIF format Still picture in JPEG format Still picture in BMP format
Audio	Basic X-wav	Audible sound (au) Wave file
Video	Mpeg Quicktime	Movie in MPEG format Mov.
Multipart	Mixed	Independent parts in the specified order

MIME Types : Multipart Types

MIME version

From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
Content-Type: multipart/mixed; boundary=98766789

Header/Body separators

--98766789
Content-Transfer-Encoding: quoted-printable
Content-Type: text/plain

Dear Bob,
Please find a picture of a crepe.
--98766789
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

base64 encoded data

.....
.....base64 encoded data
--98766789--

2: Application Layer 75

Received Message

- Receiving Server ที่ได้รับ message ด้วย RFC 822 และ MIME header line จะทำการเพิ่ม
- Received : header line ในบรรทัดบนสุดของ message

Received: from crepes.fr by hamburger.edu; 12 Oct 98 15:27:39 GMT

From: alice@crepes.fr
To: bob@hamburger.edu
Subject: Picture of yummy crepe.
MIME-Version: 1.0
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

2: Application Layer 76

Received Message

- บางครั้ง บรรทัด **Received:** มีหลายบรรทัด เช่น

```
Received: from hamburger.edu by sushi.jp; 3 Jul 01 15:30:01 GMT
Received: from crepes.fr by hamburger.edu; 3 Jul 01 15:17:39 GMT
```

- เกิดจาก user มีการ forward mail ไปยัง SMTP Server
อื่นๆ

2: Application Layer 77

Base64 Encoding Table

Base64 Encoding Table

Value	Binary	Char									
0	000000	A	16	010000	Q	32	100000	g	48	110000	w
1	000001	B	17	010001	R	33	100001	h	49	110001	x
2	000010	C	18	010010	S	34	100010	i	50	110010	y
3	000011	D	19	010011	T	35	100011	j	51	110011	z
4	000100	E	20	010100	U	36	100100	k	52	110100	0
5	000101	F	21	010101	V	37	100101	l	53	110101	1
6	000110	G	22	010110	W	38	100110	m	54	110110	2
7	000111	H	23	010111	X	39	100111	n	55	110111	3
8	001000	I	24	011000	Y	40	101000	o	56	111000	4
9	001001	J	25	011001	Z	41	101001	p	57	111001	5
0	001010	K	26	011010	a	42	101010	q	58	111010	6
11	001011	L	27	011011	b	43	101011	r	59	111011	7
12	001100	M	28	011100	c	44	101100	s	60	111100	8
13	001101	N	29	011101	d	45	101101	t	61	111101	9
14	001110	O	30	011110	e	46	101110	u	62	111110	+
15	001111	P	31	011111	f	47	101111	v	63	111111	/

2: Application Layer 78

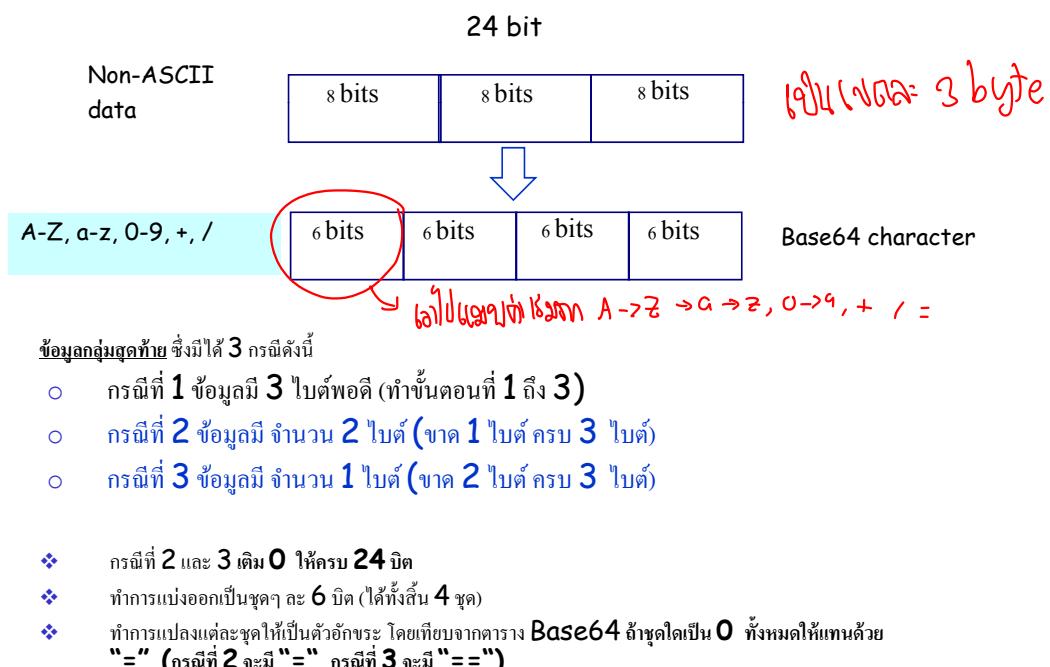
Base64 Encoding

1. ทำการเข้ารหัสครั้งละ 3 ตัวอักษรระ โดยทำการแปลงแต่ละตัวอักษรเป็นเลขฐานสองขนาด 8 บิต โดยเทียบจากตาราง ASCII (ให้เติม 0 ตรงตำแหน่งบิตซ้ายสุด ให้เป็น 8 บิต)
2. นำเลขฐานสองของแต่ละตัวอักษรมาเขียนเรียงกัน (24 บิต)
3. ทำการแบ่งออกเป็นชุดๆ ละ 6 บิต (ได้ทั้งสิ้น 4 ชุด)
4. ทำการแปลงแต่ละชุดให้เป็นตัวอักษร โดยเทียบจากตาราง Base64
5. กลับไปทำขั้นตอนที่ 1 ใหม่ จนกว่าจะเหลือ ข้อมูลอุ่มสุดท้าย ซึ่งมีໄດ້ 3 กรณีดังนี้
 - ❖ กรณีที่ 1 ข้อมูลมี 3 ตัวอักษรพอดี (ทำขั้นตอนที่ 1 ถึง 4)
 - ❖ กรณีที่ 2 ข้อมูลมี จำนวน 2 ตัวอักษร
 - ❖ กรณีที่ 3 ข้อมูลมี จำนวน 1 ตัวอักษร
6. ทำการแปลงตัวอักษรที่เหลือ (2 ตัวหรือ 1 ตัว) ให้เป็นเลขฐานสอง
7. เติม 0 ให้ครบ 24 บิต
8. ทำการแบ่งออกเป็นชุดๆ ละ 6 บิต (ได้ทั้งสิ้น 4 ชุด)
9. ทำการแปลงแต่ละชุดให้เป็นตัวอักษร โดยเทียบจากตาราง Base64 ถ้าชุดใดเป็น 0 ทั้งหมดให้แทนด้วย “=”

ASCII ສົມທະນະ ຈຶກ ລະບົບໂຄງການ

ເລືອດູບ set ຂະ 8 ໂຕໂດຍ

Base64 Encoding



กรณีที่ 1 ครบ 3 ไบต์ หรือ 24 บิต

- จงแปลง ABC ให้อยู่ในรูปของ Base64
- $A=41_{16}$, $B=42_{16}$, $C=43_{16}$
- 0100 0001 0100 0010 0100 0011
- 0100 0001 0100 0010 0100 0011
- Q U J D

- $ABC \rightarrow QUJD$

2: Application Layer 86

กรณี 2 เหลือ 2 ไบต์

- จงแปลง $ABCDE$ ให้อยู่ในรูปของ Base64
 - $A=41_{16}$, $B=42_{16}$, $C=43_{16}$, $D=44_{16}$, $E=45_{16}$
 - 0100 0001 0100 0010 0100 0011 01000100 01000101
 - 0100 0001 0100 0010 0100 0011 01000100 01000101 00000000
 - Q U J D R E U =
 - " "
 - $ABCDE \rightarrow QUJDREU=$
- มีเพียง 1 byte

2: Application Layer 87

กรณี 3 เหลือ 1 ไบต์

- จงแปลง ABCD ให้อยู่ในรูปของ Base64

ผลลัพธ์ = ค่าเบนดา

$$A=41_{16}, B=42_{16}, C=43_{16}, D=44_{16}$$

4 byte 0 ที่เสีย

0100 0001 0100 0010 0100 0011 01000100

จะหายไปหมด

0100 0001 0100 0010 0100 0011 01000100

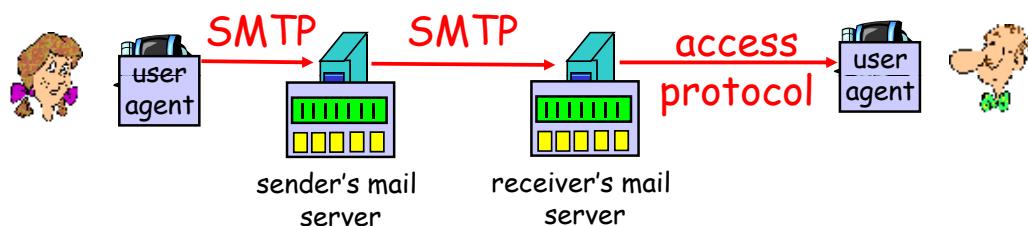
Q U J D R A = =

- ABCD → QUJDRA==

กรณีนี้จะเสียไป 4 byte ที่ไม่ได้ใช้กัน

2: Application Layer 88

Mail access protocols



- SMTP: delivery/storage to receiver's server

- Mail access protocol: retrieval from server

- ☞ POP: Post Office Protocol [RFC 1939] Port No. = 110
- authorization (agent <-->server) and download
- ☞ IMAP: Internet Mail Access Protocol [RFC 1730] Port No. = 143
- more features (more complex)
 - manipulation of stored messages on server
- ☞ HTTP: gmail, Hotmail, Yahoo! Mail, etc.

สามารถเข้าถึงได้จากทุกที่ในโลก
ทุกที่ทุกเวลา

2: Application Layer 89

POP3 protocol

authorization phase

- o client commands:
 - ❖ user: declare username
 - ❖ pass: password
- o server responses
 - ❖ +OK
 - ❖ -ERR

transaction phase, client:

- o list: list message numbers
- o retr: retrieve message by number
- o dele: delete
- o quit

```

S: +OK POP3 server ready
C: user bob
S: +OK
C: pass hungry
S: +OK user successfully logged on

```

```

C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: <message 1 contents>
S: .
C: dele 1
C: retr 2
S: <message 1 contents>
S: .
C: dele 2
C: quit → TCP connection
S: +OK POP3 server signing off

```

2: Application Layer 90

POP3 (more) and IMAP

More about POP3

- o Previous example uses "download and delete" mode.
- o Bob **cannot** re-read e-mail if he changes client
- o "Download-and-keep": copies of messages on different clients
- o POP3 is stateless across sessions

IMAP → email in server session

- o Keep all messages in one place: the **server**
- o Allows user to organize messages in **folders**
- o IMAP keeps user state across sessions:
 - o names of **folders** and mappings between message IDs and folder name

2: Application Layer 91

Chapter 2: Application layer

- 2.1 Principles of network applications
- 2.2 Web and HTTP
- 2.3 FTP
- 2.4 Electronic Mail
 - SMTP, POP3, IMAP
- 2.5 DNS
- 2.6 P2P applications
- 2.7 Socket programming with TCP
- 2.8 Socket programming with UDP

2: Application Layer 92

Application នៃទីតាំង 2 នេះ → ① User និង ② Application
② application នៃការប្រើប្រាស់
នៅក្នុងគ្រប់គ្រង

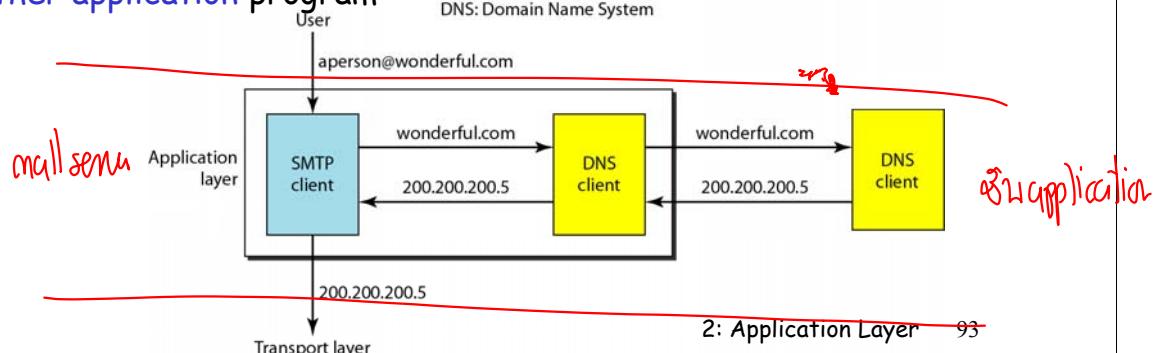
DNS: Domain Name System

- Client/server applications can be divided into two categories
 - Applications that can be directly used by user, such as e-mail
 - Applications that support other application program
- Domain Name System (DNS) is a supporting program that is used by other programs

នៅលើសម្រាប់អាមេរិក address

នៅក្នុងគ្រប់គ្រង

SMTP: Simple Mail Transfer Protocol (e-mail)
DNS: Domain Name System



2: Application Layer 93

ព័ត៌មាននៃ email address ត្រូវបានបញ្ជាក់ → ព័ត៌មាន convert to IP

DNS: Domain Name System

Social security Number (SSN)

People: many identifiers:

- ❖ SSN, name, passport #

Internet hosts, routers:

- ❖ IP address (32 bit) - used for addressing datagrams
- ❖ "name", e.g., www.yahoo.com - used by humans

Q: map between IP addresses and name ?

Domain Name System:

- *Distributed database* implemented in hierarchy of many *Name Servers*
- *Application-layer protocol* host, routers, name servers to communicate to *resolve* names (address/name translation)
 - ❖ note: core Internet function, implemented as application-layer protocol
 - ❖ complexity at network's "edge"

2: Application Layer 94

DNS

```
C:\Windows\system32\cmd.exe
C:\>nslookup www.yahoo.com
Server: diamond.ce.kmitl.ac.th
Address: 161.246.4.3

Non-authoritative answer:
Name: ds-sg-fp3.wg1.b.yahoo.com
Addresses: 2406:2000:ac:8::c:9102
           2406:2000:ac:8::c:9101
           106.10.139.246
Aliases: www.yahoo.com
          fd-fp3.wg1.b.yahoo.com
          ds-fp3.wg1.b.yahoo.com
          ds-sg-fp3-lfb.wg1.b.yahoo.com

C:\>
```

```
C:\Windows\system32\cmd.exe
C:\>nslookup www.ce.kmitl.ac.th
Server: diamond.ce.kmitl.ac.th
Address: 161.246.4.3

Name: jeweler19.ce.kmitl.ac.th
Address: 161.246.4.119
Aliases: www.ce.kmitl.ac.th

C:\>_
```

```
C:\Windows\system32\cmd.exe
C:\>nslookup webmail.kmitl.ac.th
Server: diamond.ce.kmitl.ac.th
Address: 161.246.4.3

Non-authoritative answer:
Name: blue18.crsc.kmitl.ac.th
Address: 161.246.34.53
Aliases: webmail.kmitl.ac.th

C:\>
```

DNS services

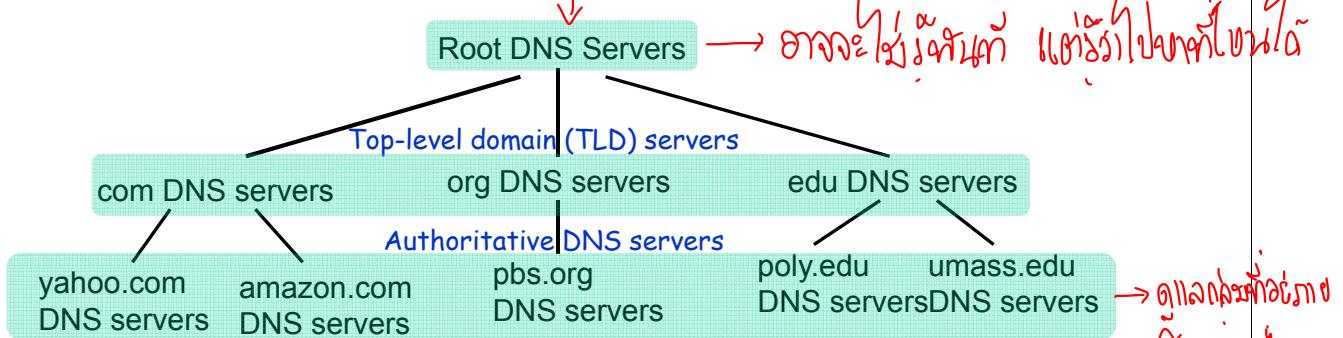
- hostname to IP address translation
- host aliasing
 - Canonical, alias names
- mail server aliasing
- load distribution
 - replicated Web servers: set of IP addresses for one canonical name

Why not centralize DNS?

- single point of failure
- traffic volume
- distant centralized database
- maintenance

doesn't scale! 2: Application Layer 95

Distributed, Hierarchical Database



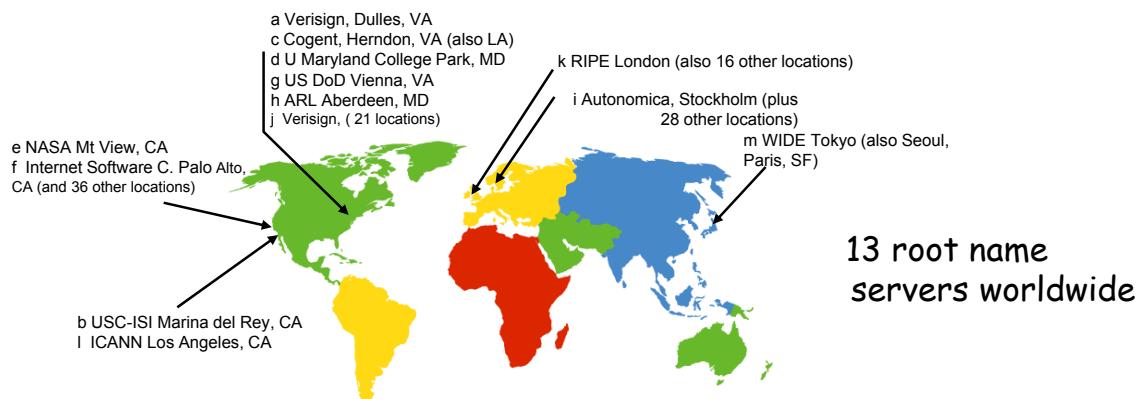
Client wants IP for www.amazon.com; 1st approx:

- client queries a **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**

2: Application Layer 96

DNS: Root name servers

- contacted by **Local Name Server** that can not resolve name
- **root name server**:
 - ❖ contacts **authoritative name server** if name mapping is known
 - ❖ **gets mapping**
 - ❖ **returns mapping to local name server**



2: Application Layer 97

TLD and Authoritative Servers

- Top-level domain (TLD) servers:
 - ❖ responsible for com, org, net, edu, etc, and all top-level country domains uk, fr, ca, jp. → ຈຳກັບຄານມາດິລິຕົວ.
 - ❖ Network Solutions maintains servers for com TLD
 - ❖ Educause for edu TLD
- Authoritative DNS servers:
 - ❖ organization's DNS servers, providing authoritative hostname to IP mappings for organization's servers (e.g., Web, mail).
 - ❖ can be maintained by organization or service provider

Local 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
 - acts as proxy, forwards query into hierarchy

```
C:\>nslookup www.yahoo.com
Server:  diamond.ce.kmitl.ac.th
Address: 161.246.4.3

Non-authoritative answer:
Name:   ds-sg-fp3.wg1.b.yahoo.com
Addresses: 2406:2000:ac:8::c:9102
          2406:2000:ac:8::c:9101
          106.10.139.246
Aliases:  www.yahoo.com
          fd-fp3.wg1.b.yahoo.com
          ds-fp3.wg1.b.yahoo.com
          ds-sg-fp3-lfb.wg1.b.yahoo.com

C:\>
```

2: Application Layer 98

“default name server”

“ISP ອັດຕະໂນມໄລ້”
ISP ອັດຕະໂນມໄລ້

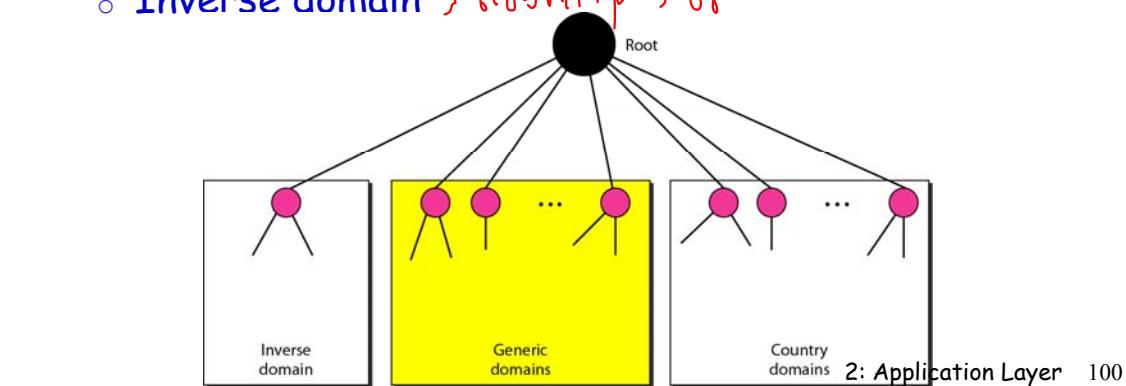
ເຊື້ອນຫຼຸດ
ເພື່ອສ່າງໃຫຍ່

2: Application Layer 99

DNS in Internet

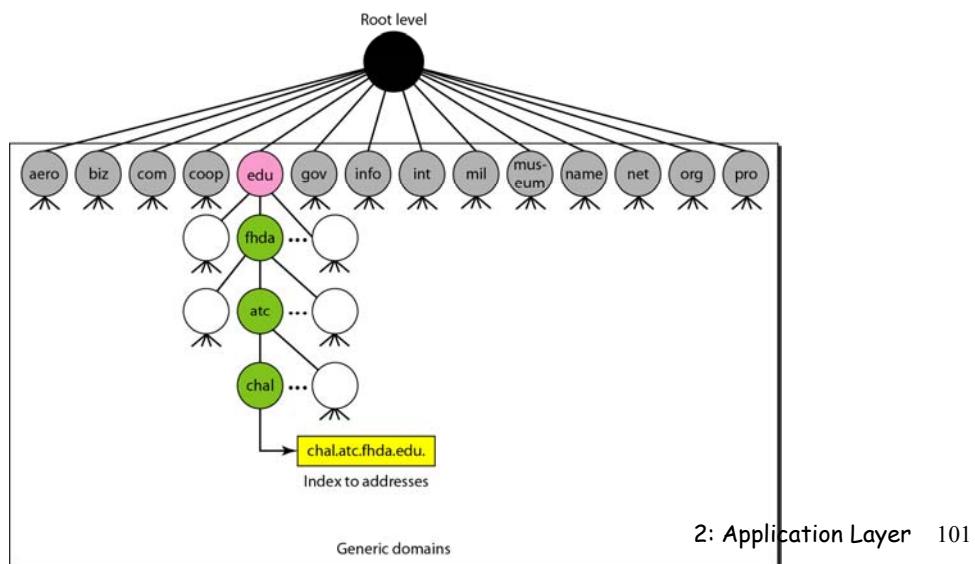
බාම්පිල්ස්ට්‍රුයෝජිත

- DNS is a protocol that can be used in different platforms
- In Internet, domain name space (tree) is divided into three different sections:
 - Generic domain
 - Country domains
 - Inverse domain



Generic Domains

- Generic domains define registered hosts according to their generic behavior
- Each node in tree defines a domain, which is an index to domain name space database



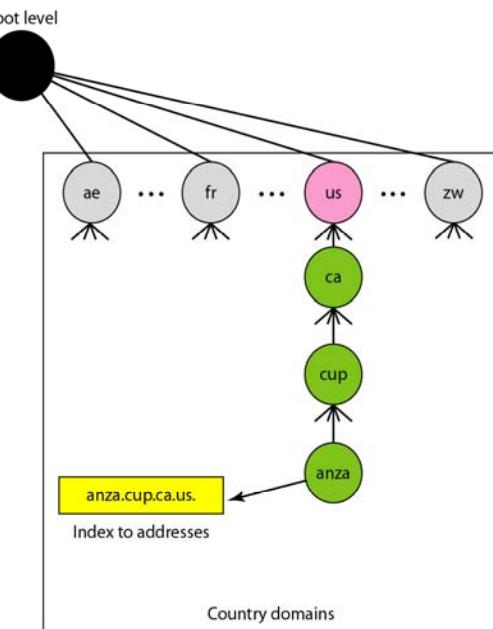
Generic domain labels

<i>Label</i>	<i>Description</i>
aero	Airlines and aerospace companies
biz	Businesses or firms (similar to “com”)
com	Commercial organizations
coop	Cooperative business organizations
edu	Educational institutions
gov	Government institutions
info	Information service providers
int	International organizations
mil	Military groups
museum	Museums and other nonprofit organizations
name	Personal names (individuals)
net	Network support centers
org	Nonprofit organizations
pro	Professional individual organizations

2: Application Layer 102

Country Domains

- Country domains section uses **two-character country abbreviations**
- Second labels can be organization, or they can be more specific, national designations
- Ex. Anza.cup.ca.us can be translated to De Anza College in Cupertino, California, in the United State



.ae - United Arab Emirates

.fr - France

.us - United States

.zw - Zimbabwe

.ch - Switzerland

2: Application Layer 103

Inverse Domain

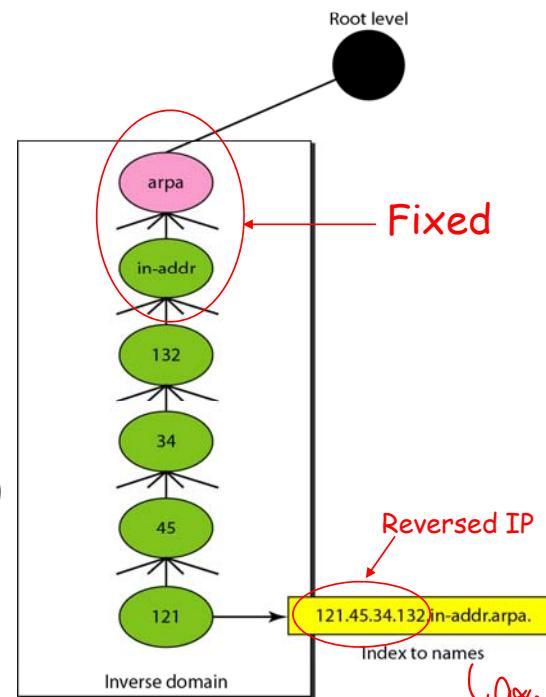
- Inverse domain is used to **map IP Address to Domain Name**
- For example, when server has received a request from client to do a task
- Although server has file that contains a list of authorized clients, only IP address of client (extracted from received IP packet) is listed
- Server asks its resolver to send a query to DNS server to **map address to name** to determine if client is on authorized list
- This type of query is called an **inverse or pointer (PTR) query**

2: Application Layer 104

នៅក្នុងរបាយការណ៍ពី Inverse domain

Inverse Domain (cont.)

- To handle a pointer query, inverse domain is added to domain name space with the **first-level node** called **arpa** (for historical reasons)
- The **second level** is also one single node named **in-addr** (for inverse address)
- The rest of the domain defines **IP addresses**



Reversed IP.in-addr.arpa

IP address = 132.34.45.121

2: Application Layer

Query goes this way
តាមលទ្ធផលដែលបានស្វែងរក

105

DNS name resolution example

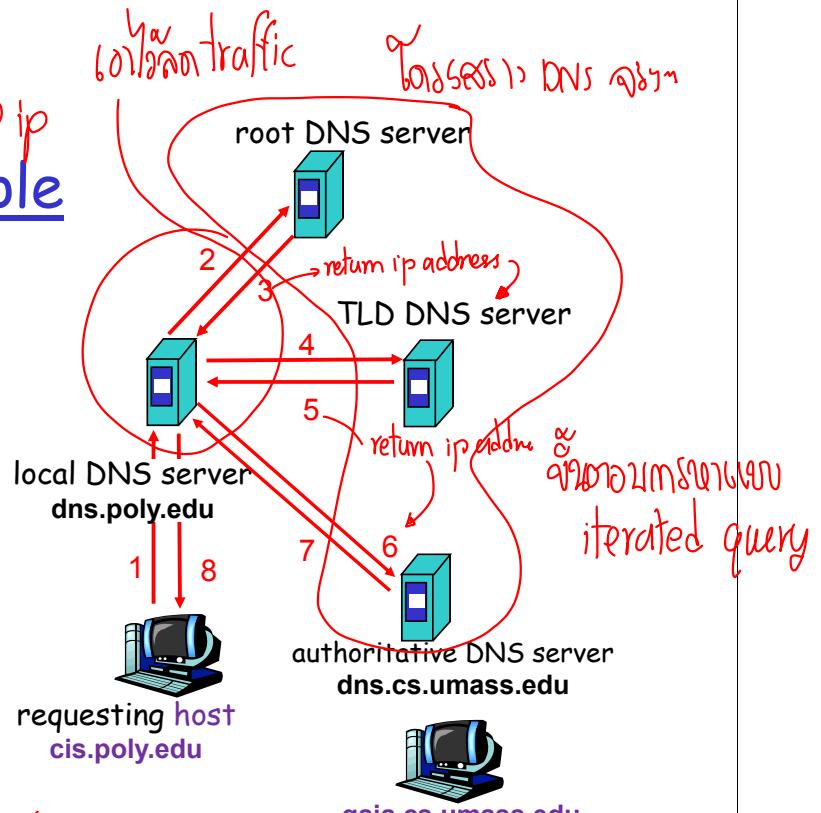
- Host at **cis.poly.edu** wants IP address for **gaia.cs.umass.edu**

↳

Iterated Query:

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

↳ (for loop iteration)

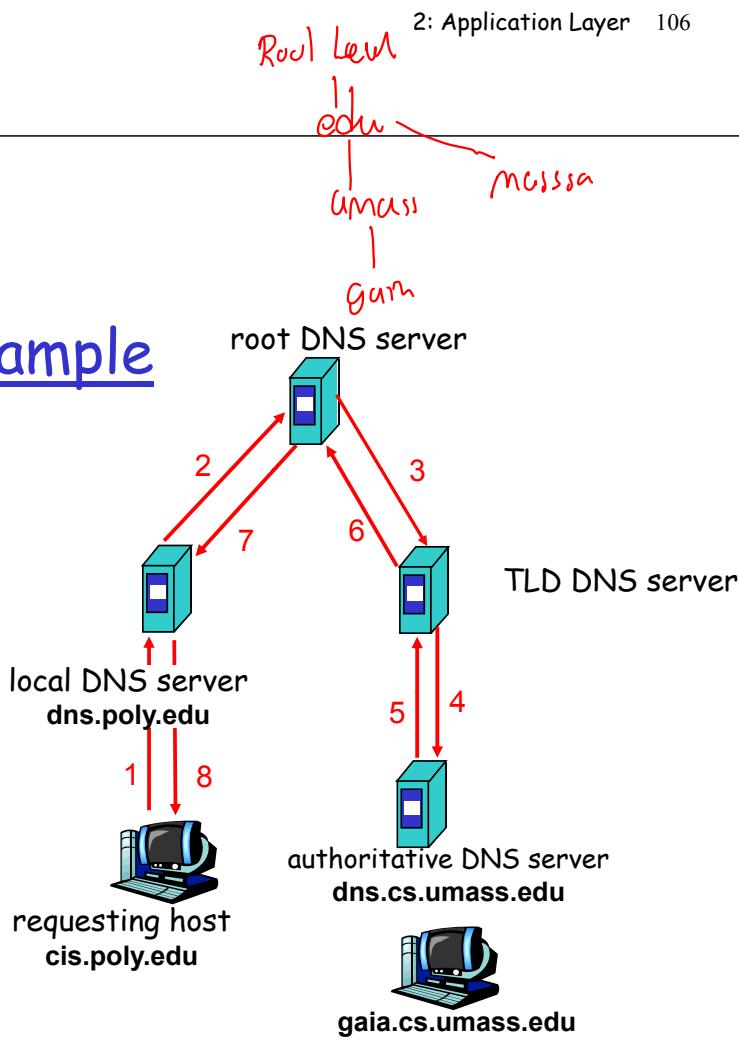


DNS name resolution example

Recursive Query:

- puts burden of name resolution on contacted name server

↳ (put burden on)



TTL : time to live of RR;

It is determines when a resource should be removed from cache

DNS records

DNS: distributed db storing Resource Records (RR)

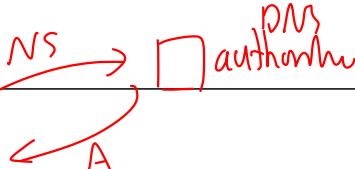
RR format: (name, value, type, TTL)

- Type=A
 - ❖ name is hostname
 - ❖ value is IP address

(relay1.bar.foo.com, 145.37.93.126, A)
- Type=NS
 - ❖ name is domain (e.g. foo.com)
 - ❖ value is hostname of authoritative name server for this domain

(foo.com, dns.bar.foo.com, NS)

name server
server ដែលផ្តល់
ទិន្នន័យនៃ DNS



- Type=CNAME
 - ❖ name is alias name for some "canonical" (the real) name
 - ❖ value is canonical name

www.ibm.com is really servereast.backup2.ibm.com
- Type=MX (mail)
 - ❖ value is canonical name of mail server associated with name

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

2: Application Layer 109

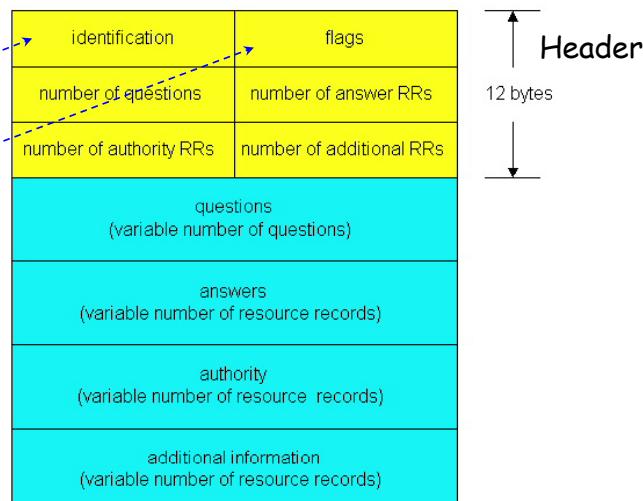
មួយគម្រោងសម្រាប់ mail server

DNS protocol, messages

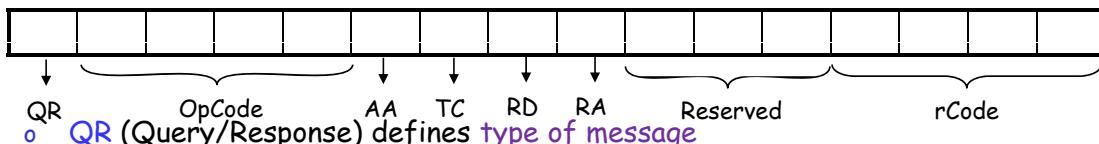
DNS protocol : **query** and **reply** messages, both with same message format

message header

- identification: 16 bits # for query, reply to query uses same #
- flags: (16 bits)
 - ❖ query or reply
 - ❖ recursion desired
 - ❖ recursion available
 - ❖ reply is authoritative



Flags Field



- o **QR** (Query/Response) defines **type of message**
 - o If it is **0**, the message is a **query**
 - o If it is **1**, the message is a **response**
- o **OpCode** : defines **type of query or response**
 - 0 if standard query, 1 if inverse query, 2 if a server status request
- o **AA** (authoritative answer)
 - o When it is set (value of 1) it means that **name server is an authoritative server**
 - o It is used only in **response message**
- o **TC** (truncated)
 - o When it is set (value of 1), it means that **response was more than 512 bytes** and truncated to 512
 - o It is used when DNS uses services of **UDP**
- o **RD** (recursion desired) → **U** **U** **U** **U** **U** **U**
 - o When it is **set (value of 1)**, it means client desires **recursive answer**
 - o It is **set in query message** and **repeated in response message**
- o **RA** (recursion available), this bit is valid in **response** and denotes whether **recursive query support is available**
- o **Reserved**
 - o 3-bit subfield is set to **000**
- o **rCode** shows status of error in **response**

2: Application Layer 111

u..

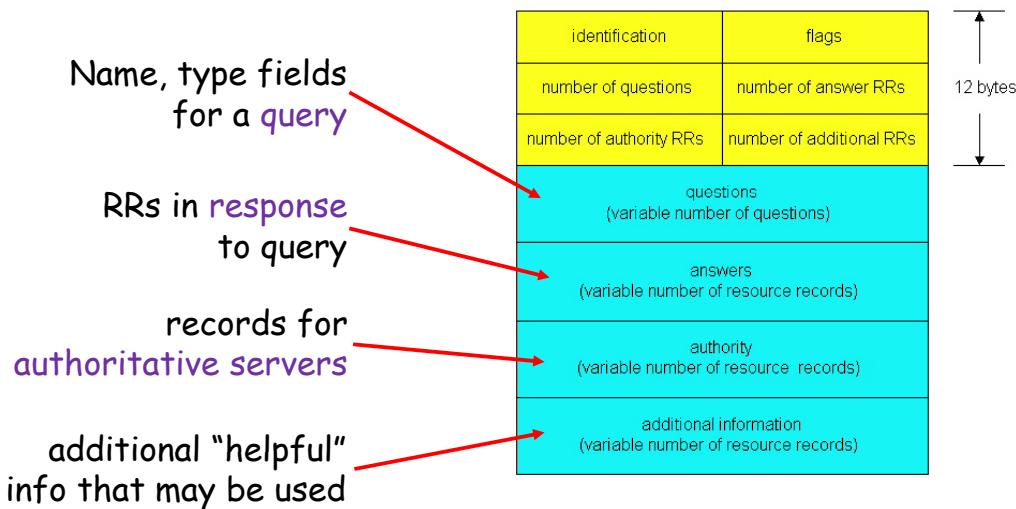
Flags Field

Value	Meaning
0	No error
1	Format error
2	Problem at name server
3	Domain reference problem
4	Query type not supported
5	Administratively prohibited
6-15	Reserved

Value of rCode

2: Application Layer 112

DNS protocol, messages



Ex.

- Answer field in reply to **MX query** contains a RR providing canonical name of mail server
- Additional section contains a **Type A** record providing IP address for canonical hostname of mail server

2: Application Layer 113

Inserting records into DNS

- example: new startup "**Network Utopia**"
- register name **networkutopia.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)** authn livev none com
hostname of authoritative name server
 - **(dns1.networkutopia.com, 212.212.212.1, A)** ; add num
; add ip
- create authoritative server
 - Type A record for **www.networkutopia.com**;
 - Type MX record for **networkutopia.com**
- How do people get IP address of your Web site?

TTL: time to live of RR:
It is determined when a resource should be removed from cache

DNS records

DNS: database storing Resource Records (RR)

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

- Type=A
 - ◆ **name** is hostname
 - ◆ **value** is IP address (relay1.bar.foo.com, 149.37.93.126, A)
- Type=CNAME
 - ◆ **name** is alias name for some "canonical" (the real) name. www.lim.com is really servereast.backup2.lim.com
 - ◆ **value** is canonical name (foo.com, relay1.bar.foo.com, CNAME)
- Type=NS
 - ◆ **name** is domain (e.g. foo.com)
 - ◆ **value** is hostname of authoritative name server for this domain (foo.com, dns.bar.foo.com, NS)
- Type=MX
 - ◆ **value** is canonical name of mail server associated with name (foo.com, mail.bar.foo.com, MX)

2: Application Layer 114

Encapsulation

- DNS can use either **UDP** or **TCP**.
- In both cases the well-known port used by the server is **port 53**.
- **UDP** is used when the size of response message is **less than 512 bytes** because most UDP packages have a **512-byte packet size limit**.
- If the size of **(response message)** is more than 512 bytes, a **TCP** connection is used.

2: Application Layer 115

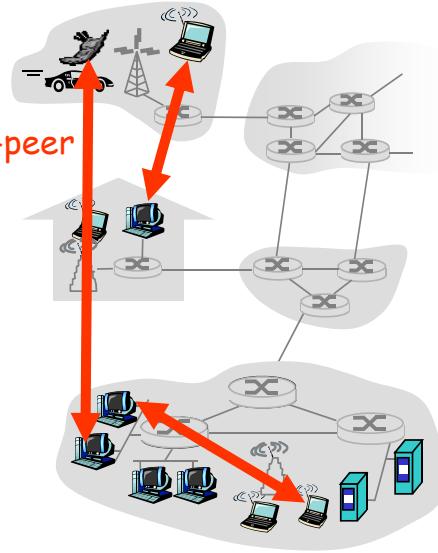
Chapter 2: Application layer

- 2.1 Principles of network applications
 - app architectures
 - app requirements
- 2.2 Web and HTTP
- 2.4 Electronic Mail
 - SMTP, POP3, IMAP
- 2.5 DNS
- 2.6 P2P applications
- 2.7 Socket programming with TCP
- 2.8 Socket programming with UDP

2: Application Layer 116

Pure P2P architecture

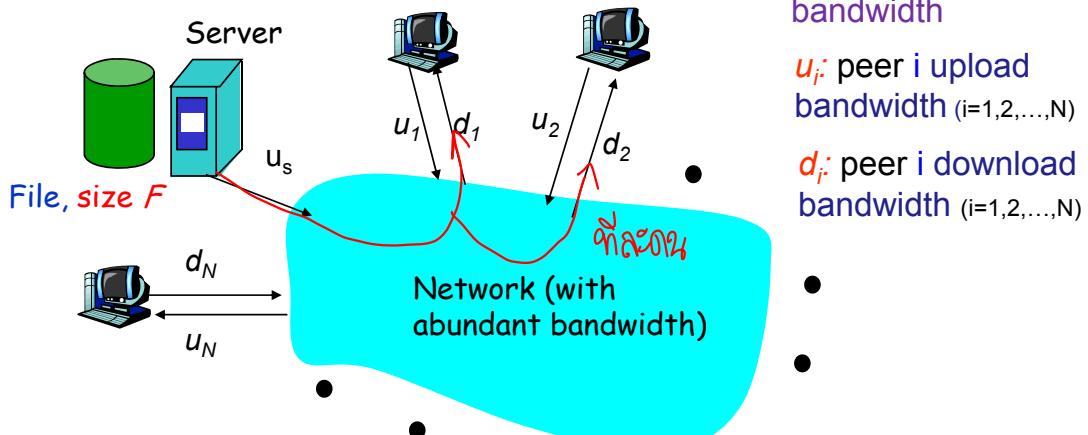
- no always-on server
- arbitrary end systems directly communicate
- peers are intermittently connected and change IP addresses
- Three topics:
 - File distribution
 - Searching for information (database distribution)
 - Case Study: Skype



2: Application Layer 117

File Distribution: Server-Client vs P2P

Question: How much time to distribute file from one server to N peers?

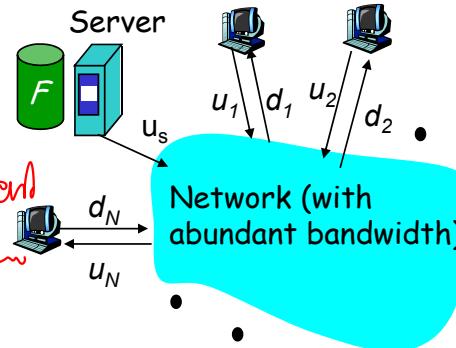


2: Application Layer 118

File distribution time: server-client

delay = download bandwidth \times server sequentially sends N copies:

- o NF/u_s time download client
- o client i takes F/d_i bandwidth time to download



Time to distribute F to N clients using client/server approach

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

increases linearly in N (for large N)

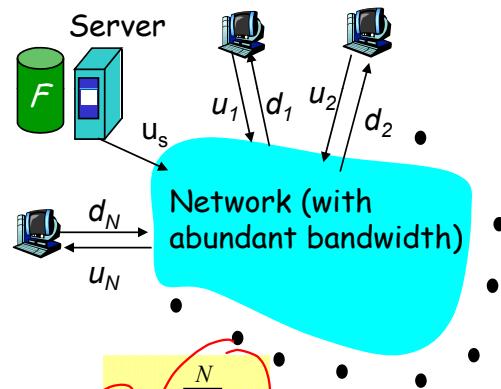
2: Application Layer 119

client has bandwidth
available

clients upload file at this rate

File distribution time: Peer to Peer (P2P)

- o server must send one copy: one peer
- o F/u_s time
- o client i takes F/d_i time to download
- o NF bits must be downloaded (aggregate)
- o fastest possible upload rate:



$$D_{P2P} = \max \left\{ \frac{F}{u_s}, \frac{F}{d_{min}}, \frac{NF}{u_s + \sum_{i=1}^N u_i} \right\}$$

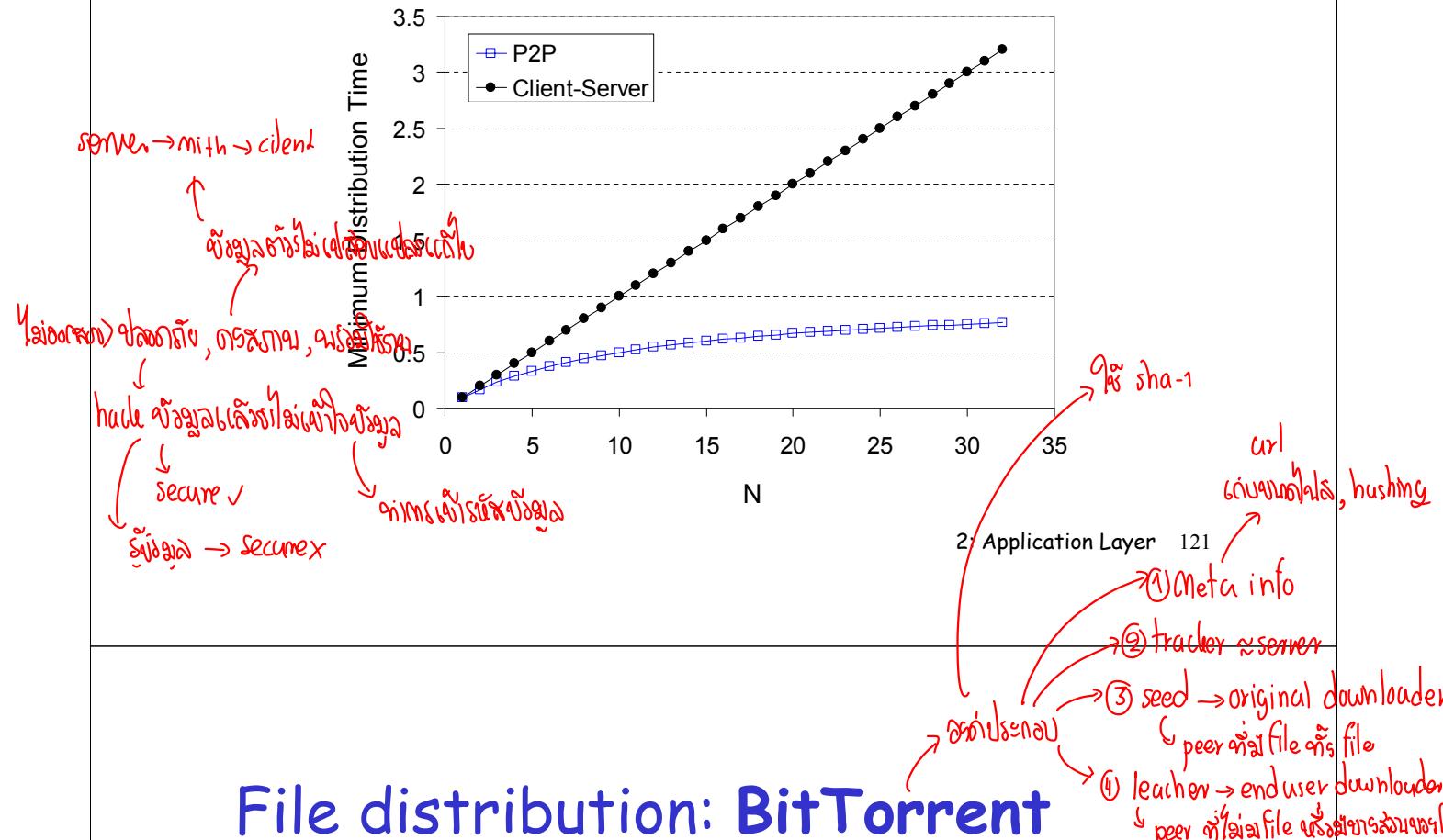
Total upload capacity of system as a whole = Upload rate of server plus upload rate of each of individual peer

2: Application Layer 120

max upload rate of all clients to server

Server-client vs. P2P example

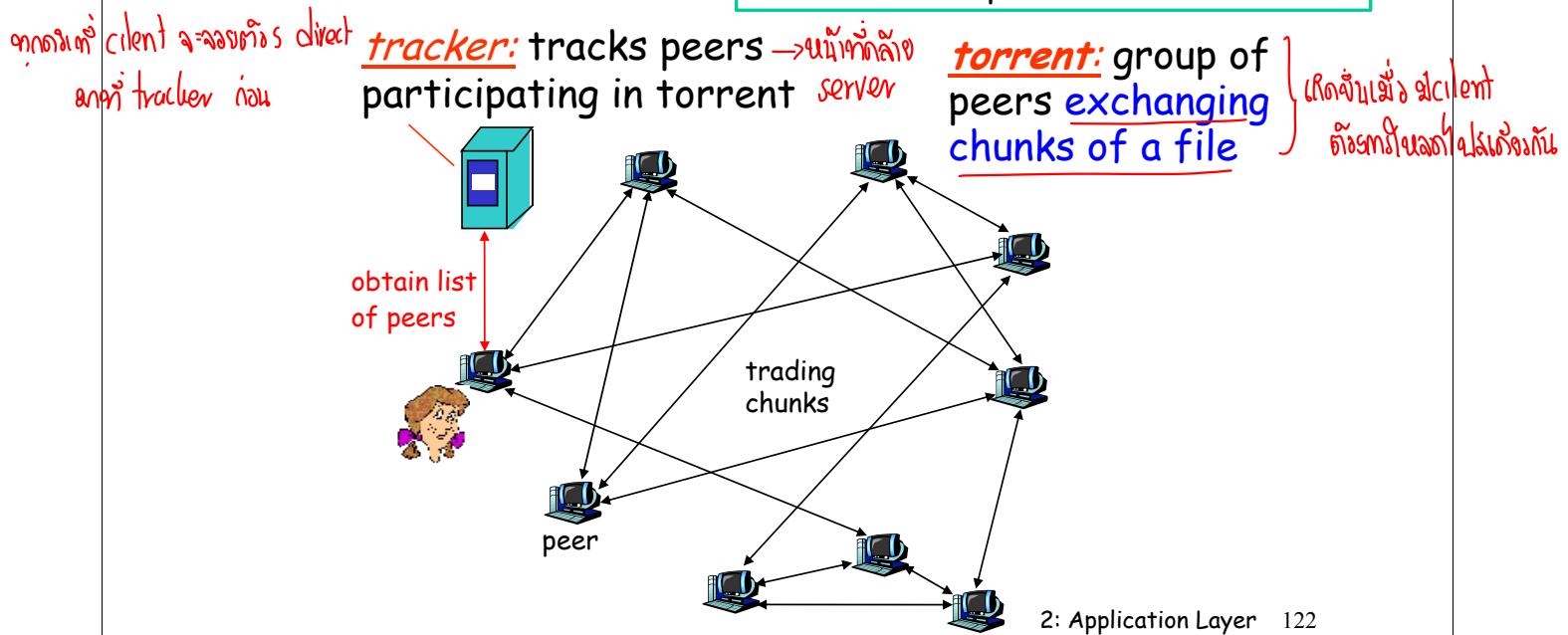
Client upload rate = u , $F/u = 1$ hour, $u_s = 10u$, $d_{\min} \geq u_s$



File distribution: BitTorrent

- P2P file distribution

Collection of all peers participating in distribution of particular file

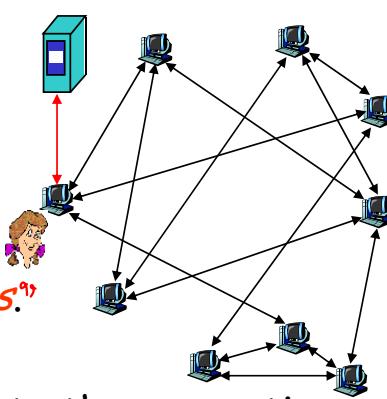




BitTorrent (1)

- file divided into 256KBytes "chunks".
- peer joining torrent: ตัวผู้ใช้งาน
 - has no chunks, but will accumulate them over time
 - registers with tracker to get list of peers,
 - connects to subset of peers ("neighbors") → peer ที่เพื่อน share ไฟล์นั้น
- while downloading, peer uploads chunks to other peers.
- peers may come and go → ภูมิทัศน์ของเพื่อนจะเปลี่ยนไป
- once peer has entire file, it may (selfishly) leave or (altruistically) remain

เห็นแก่ผู้อื่น/เดียบสละ



2: Application Layer 123

จังหวะ = พัฒนา

ผู้นำ peer ที่ต้องรู้สึกดี

ผลลัพธ์การซ่อนอยู่ในแบบ

ดูแลรักษาอย่างดี

ขอ chunk ของ → หาเพื่อนที่ดี → ได้รับจากเพื่อนที่ดี chunk

BitTorrent (2)

- Chunk ไหนควร request ก่อน
- Peer เพื่อนบ้านไหนที่ควรส่ง request chunk

① Pulling Chunks

- at any given time, different peers have different subsets of file chunks
- periodically, peer (Alice) asks each neighbor for list of chunks that they have
- Alice sends requests for her missing chunks
 - ❖ rarest first

Chunks that have fewest repeated copies among neighbors and then request those rarest chunks first

② Sending Chunks: tit-for-tat

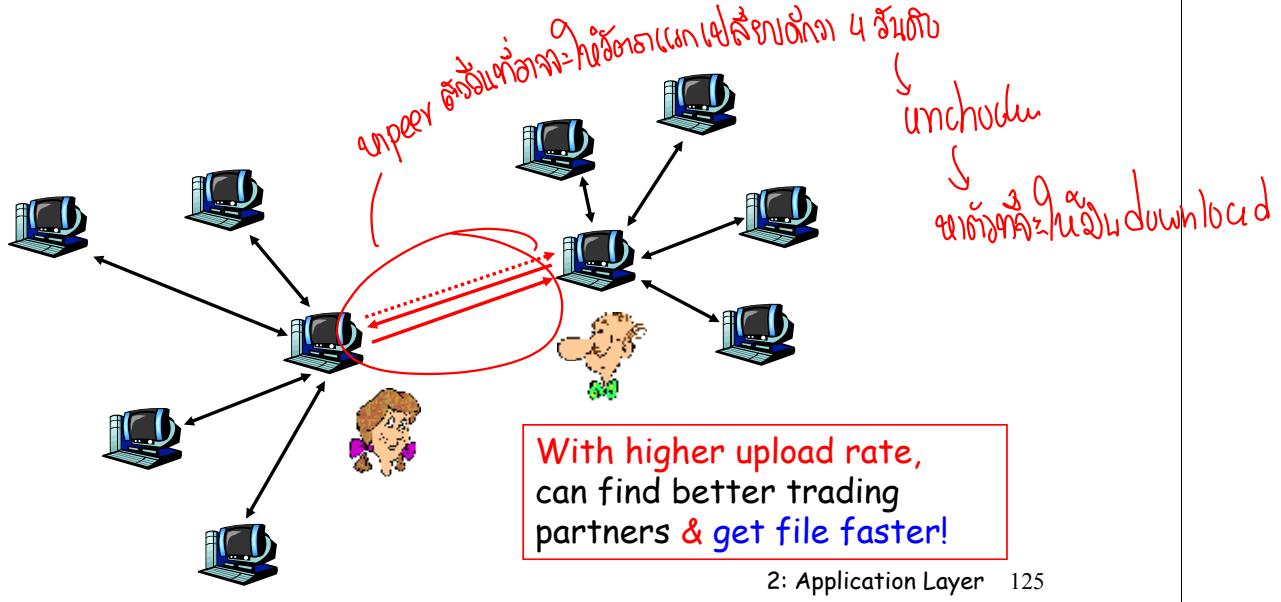
- Alice sends chunks to four neighbors currently sending her chunks at highest rate
 - ❖ re-evaluate top 4 every 10 secs
- every 30 secs: randomly select another peer, starts sending chunks
 - ❖ newly chosen peer may join top 4
 - ❖ "optimistically unchoke"

peer ที่ไม่ได้รับ download มาก choke

2: Application Layer 124

BitTorrent: Tit-for-tat (การตอบแทน)

- (1) Alice "optimistically unchoke" Bob การแลกเปลี่ยนกัน
- (2) Alice becomes one of Bob's top-four providers; Bob reciprocates
- (3) Bob becomes one of Alice's top-four providers



Distributed Hash Table (DHT)

(Indexing and Searching technique)

- ❑ DHT = distributed P2P database
- ❑ Database has (key, value) pairs;
 - ❖ key: social security (ss) number; value: human name
 - Ex. Key-value pair is (156-45-7081, Johnny Wu)
 - ❖ key: content type; value: IP address
 - Ex. Content type : Name of movies, albums, and software
 - Ex. Key-value pair is (Led Zeppelin IV, 203.17.123.38))
 - Ex. Key-value pair is (Doraemon, 200.15.100.22)
- ❑ Peers query Database with key
 - ❖ DB returns values that match the key
- ❑ Peers can also insert (key, value) pairs into our database

DHT Identifiers



An approach to designing P2P database

- Assign **integer** identifier to each peer in range $[0, 2^n - 1]$.
 - ❖ Each identifier can be represented by **n** bits
- Require each key to be an **integer** in **same range** using hash function.
- To get integer keys, hash original key.
 - ❖ eg, $\text{key} = h(\text{"Led Zeppelin IV"}) \rightarrow \text{key} \text{ (hash)} \rightarrow \text{mod } 2^n$
 - ❖ This is why they call it a **distributed "hash" table**

2: Application Layer 127

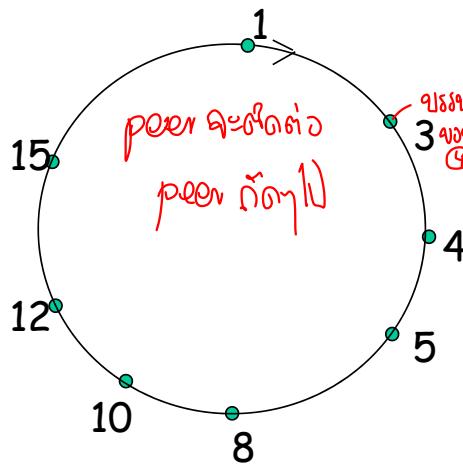
How to assign keys to peers?

- Central issue:
 - ❖ Assigning (key, value) pairs to peers.
- Rule: assign key to the peer that has the **closest** ID.
- Convention in lecture: defining **closest peer** as **immediate successor** of key.
- Ex: n=4; peers: 1, 3, 4, 5, 8, 10, 12, 14;
 - ❖ key = 13, then successor peer = 14
 - ❖ key = 15, then successor peer = 1

If key is larger than all peer identifiers, we use **modulo- 2^n** convention storing (key-value) pair in peer with smallest ID₂₈

Circular DHT (1)

key នៃវា ត្រូវបានដោយ peer ឯង



Building a database can be with client-server architecture

- all (key,value) pairs are stored in one central server such as Napster.

If peer keep track of all peers in system (peer IDs and IP address), peer could locally determine closest peer.

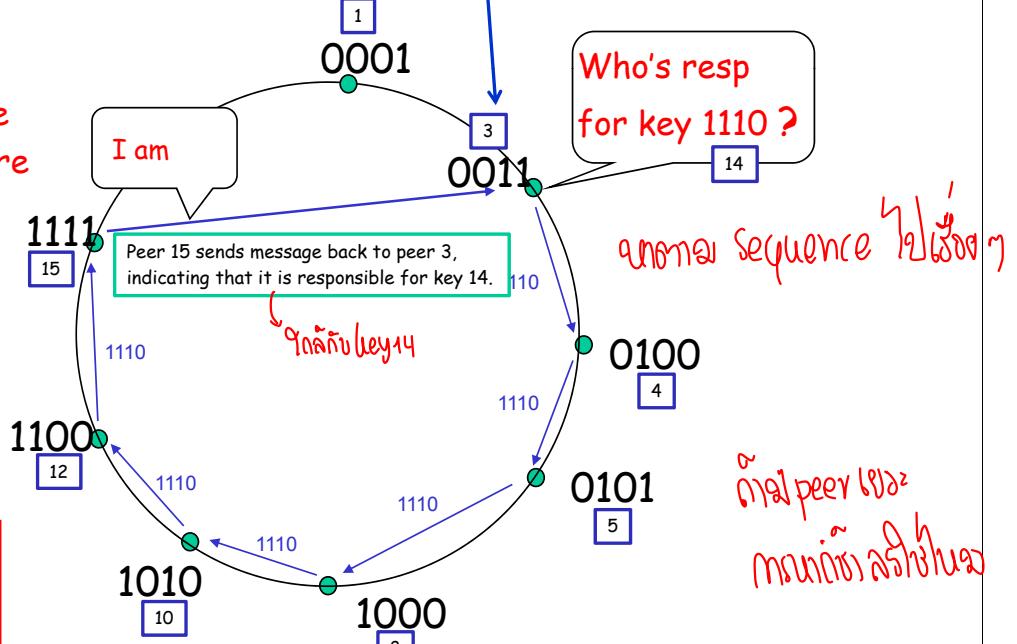
But such approach requires each peer to keep track of all other peers in DHT- which is completely impractical for large-scale system with millions of peers

- Each peer *only* aware of immediate successor and predecessor.
 - "Overlay network"
 - peers form abstract logical network
 - Links are not physical links, but are simply virtual liaisons
- 2: Application Layer 129

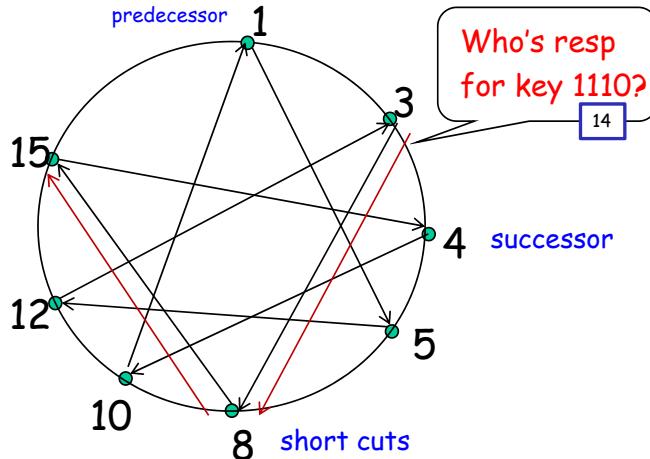
Circle DHT (2)

Peer 3 wants to determine which peer in DHT is responsible for key 14
-either for inserting or querying for (key-value) pair

O(N) messages on avg to resolve query, when there are N peers



Circular DHT with Shortcuts



When peer receives message that is querying for key, it forwards message to neighbor (successor neighbor or one of short-cut neighbors which is the closest to key)

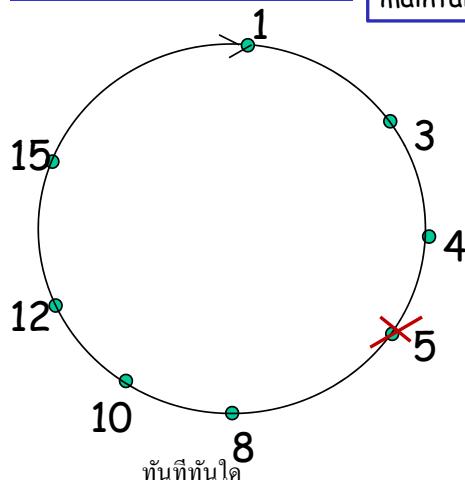
- ❑ Each peer keeps track of IP addresses of predecessor, successor, short cuts (used to expedite routing of query message).
- ❑ Reduced from 6 to 2 messages.
- ❑ Possible to design shortcuts so $O(\log N)$ neighbors, $O(\log N)$ messages in query

Short cuts can significantly reduce number of message used to process a query

กี่ peer ที่หายไป ร่องรอย กันตังสิบ ? อยู่ที่ไหนแล้ว
↳ ก็ไม่รู้ ip ของเพื่อนๆ กี่ 2 ใจความ → ก็ ping ดู

Peer Churn

In P2P system, peer can come or go without warning. When designing DHT, we must be concerned about maintaining DHT overlay in the presence of such peer churn



- To handle peer churn, require each peer to know IP address of its two successors.
- Each peer periodically pings its two successors to see if they are still alive.

- ❑ Peer 5 abruptly leaves
- ❑ Peer 4 detects; makes 8 its immediate successor; asks 8 who its immediate successor is; makes 8's immediate successor its second successor (peer 10).
- ❑ What if peer 13 wants to join?



This is incorrect. The torrent files contain no peer information.

> But how do you connect to that peer? How do you find one if there

> is no centralized location to meet up?

Important: Both DHT and a normal Tracker provide the IP address and ports for peers that are active for a Hash ID. There is very little difference between the two. Both are databases that store and associate, as a minimum, IP addresses and HashIDs. (Both can store and associate additional information, too.) After receiving an IP address from a Tracker or DHT, your client will attempt a connection to the IPs:Ports it provided.

You don't "meet up" on a tracker or on DHT. Instead, think of them as bulletin boards -- you go there and read information that others have left before, and you put your information there, and then you leave.

When you start a download on DHT, you basically send an announce message (like you would to the tracker) that is stored on the DHT network. The torrent's Hash ID and your IP address and port are stored. At the same time, you send a query to the DHT network, asking if any IP addresses are associated with the Hash ID. The DHT responds as a tracker would, providing any known IP addresses and ports.

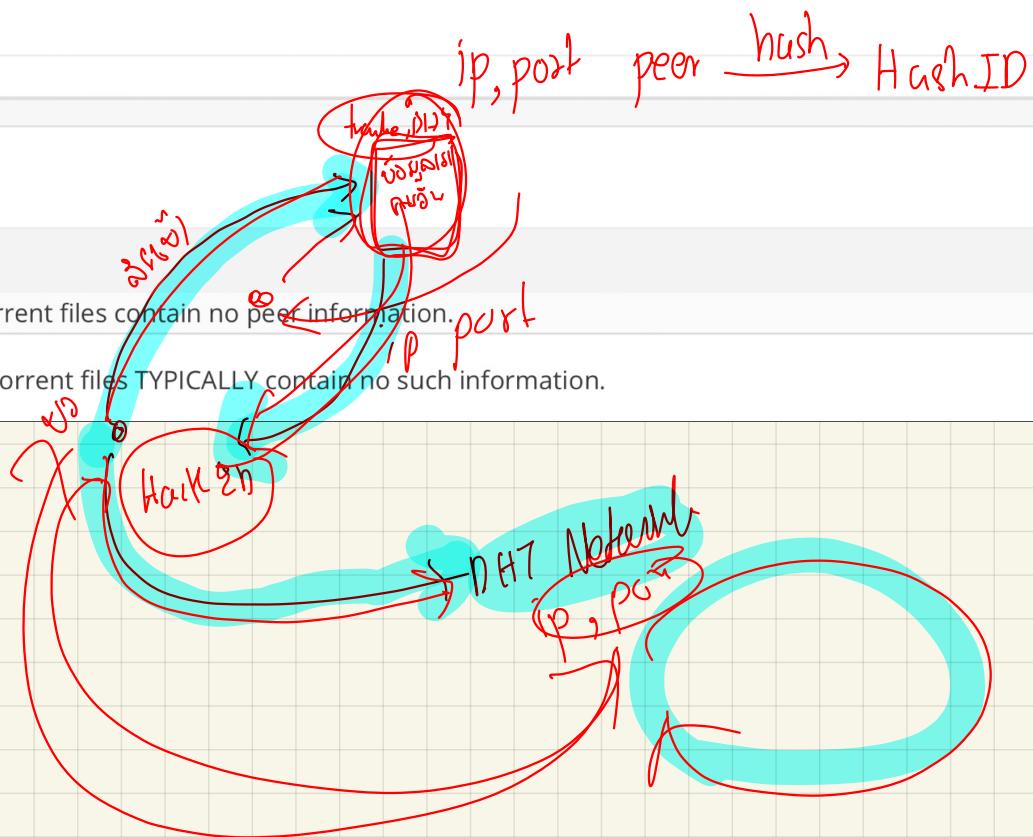
Posted July 8, 2007

Report post

Quote

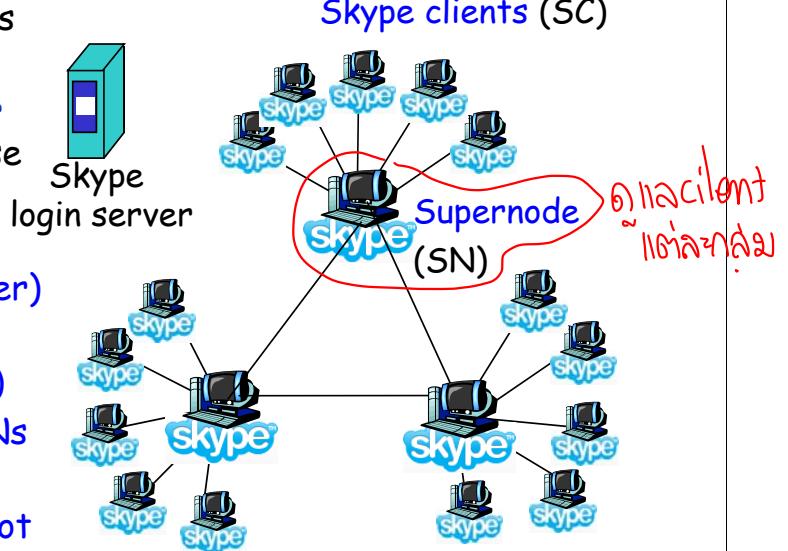
This is incorrect. The torrent files contain no peer information.

This is actually incorrect. Torrent files TYPICALLY contain no such information.



P2P Case study: Skype

- ❑ inherently P2P: pairs of users communicate.
- ❑ proprietary application-layer protocol (inferred via reverse engineering)
- ❑ hierarchical overlay with Supernodes (SNs) (super peer)
- ❑ Index maps usernames to IP addresses (and port number)
- ❑ Index is distributed over SNs
- ❑ Because Skype protocol is proprietary, it is currently not clear how index mappings are organized across supernodes

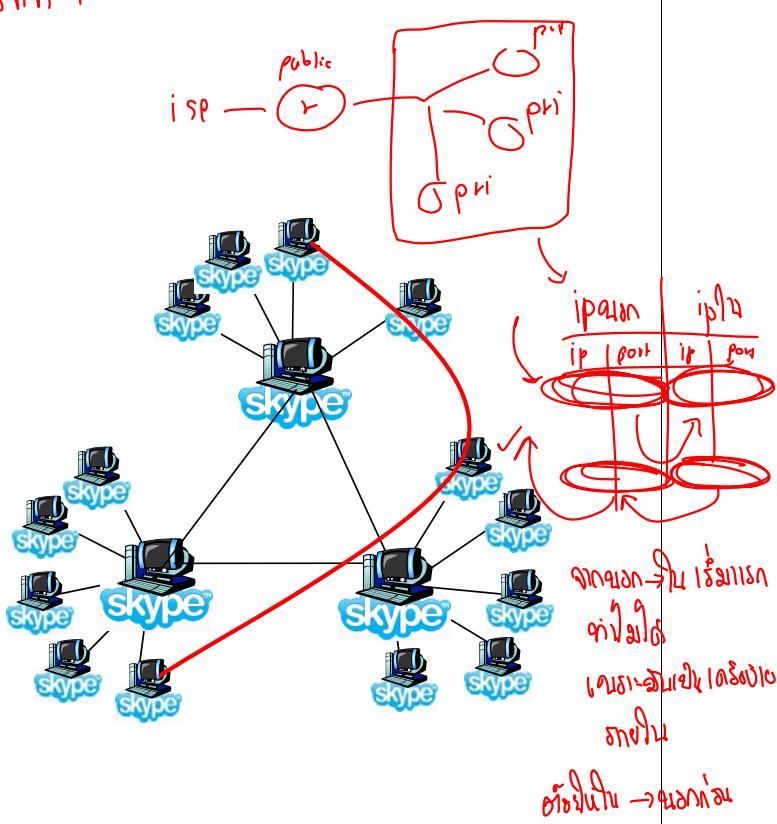


2: Application Layer 133

Peers as relays

Network Address Translator (NAT)

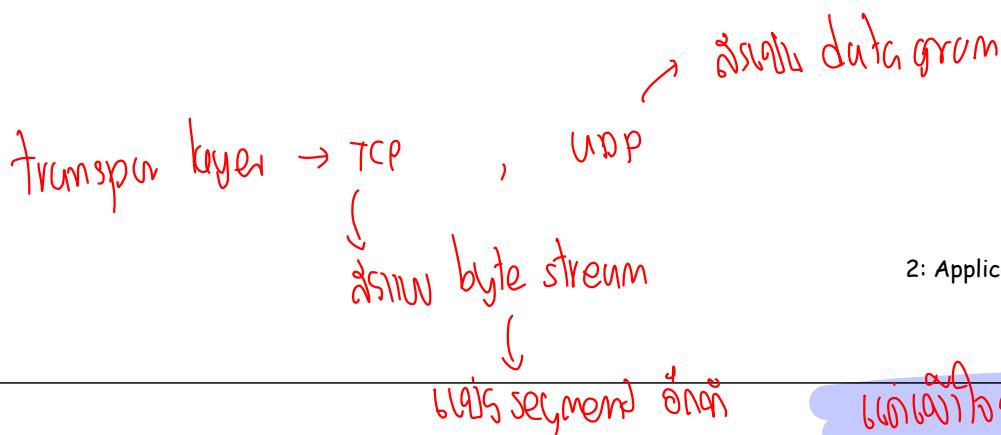
- ❑ Problem when both Alice and Bob are behind "NATs".
 - ❖ NAT prevents an outside peer from initiating a call to insider peer
- ❑ Solution:
 - ❖ Using Alice's and Bob's SNs, **Relay** is chosen
 - ❖ Each peer initiates session with relay.
 - ❖ Peers can now communicate through NATs via relay



2: Application Layer 134

Chapter 2: Application layer

- 2.1 Principles of network applications
- 2.2 Web and HTTP
- 2.3 FTP
- 2.4 Electronic Mail
 - SMTP, POP3, IMAP
- 2.5 DNS
- 2.6 P2P applications
- 2.7 Socket programming with TCP
- 2.8 Socket programming with UDP



Socket Programming → ต่อไปจะ是 Transport layer

Goal: learn how to build client/server application that communicate using sockets

↓
使う socket
使う API

Socket API (Application Programming Interface)

Berkeley Software Distribution

- introduced in BSD4.1 UNIX, 1981
- explicitly created, used, released by applications
- client/server paradigm
- two types of transport service via socket API:
 - ❖ unreliable datagram : User Datagram Protocol (UDP)
 - ❖ reliable, byte stream-oriented : Transmission Control Protocol (TCP)

socket

a *host-local, application-created, OS-controlled* interface (a "door") into which application process can both send and receive messages to/from another application process

Socket-programming using TCP

- Network Apps. consist of client program and server program.
 - When these two programs are executed, client process and server process are created.
 - Processes running on different machines communicate with each other by sending messages into socket.
 - Main task of developer's is to write code for both client and server program.
 - **Socket:** a door between application process and end-end-transport protocol (UDP or TCP)
 - byte and message → individual segment
 - **TCP service:** reliable transfer of bytes-stream from one process to another
 - controlled by Application developer
 - controlled by Operating system
 - host or server
-
- controlled by Application developer
controlled by operating system
host or server
- process
socket
TCP with buffers, variables
- Internet
- controlled by Application developer
controlled by Operating system
host or server
- process
socket
TCP with buffers, variables
- Ex. Max. buffer size, Max. segment size
- client originates request and service at server
application layer port
- 2: Application Layer 137

Socket programming with TCP

In order for server to be able to react to client's initial contact, server has to be ready.



Client must contact server

- server process must first be running
- server must have created socket (door) that welcomes client's contact

Client contacts server by:

- creating client-local TCP socket
- specifying IP address, port number of server process
- When client creates socket: client TCP establishes connection to server TCP

- When contacted by client, server TCP creates new socket for server process to communicate with client
 - ❖ allows server to talk with multiple clients
 - ❖ source port numbers used to distinguish clients (more in Chap 3)

application viewpoint

TCP provides reliable, in-order transfer of bytes ("pipe") between client and server

Client/server socket interaction: TCP

Server (running on `hostid`)

```
create socket,
port=x, for
incoming request:
welcomeSocket =
ServerSocket()
```

```
wait for incoming
connection request
connectionSocket =
welcomeSocket.accept()
```

```
read request from
connectionSocket
write reply to
connectionSocket
close
connectionSocket
```

When client creates socket: client TCP establishes connection to server TCP.
Three-way handshake takes place at transport layer which is completely transparent to client and server program

Client

TCP
connection setup

```
create socket,
connect to hostid, port=x
clientSocket =
Socket()
```

```
send request using
clientSocket
read reply from
clientSocket
close
clientSocket
```

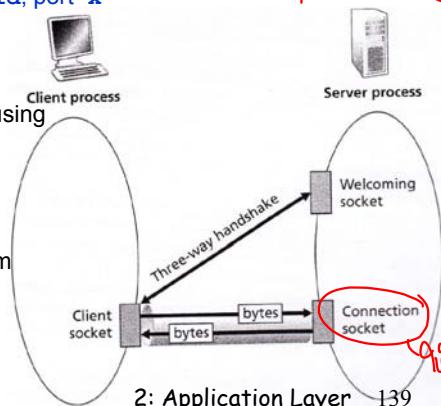
start
IP address of
server process
port number of
server process

ip:port → socket

influencing client

2: Application Layer 139

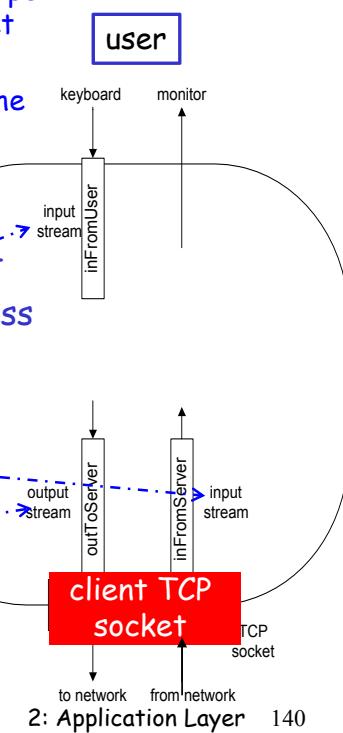
returning client



Stream jargon

- From app's perspective, TCP connection is direct **virtual pipe** between **client's socket** and **server's communication socket**
- Client process can send bytes into socket, and TCP guarantees that sever process will receive each byte in the order sent
- TCP thus provides **reliable byte-stream service** between client and server process

- A **stream** is a sequence of characters that flow into or out of a process.
- An **input stream** is attached to some input source for the process, e.g., keyboard or socket.
- An **output stream** is attached to an output source, e.g., monitor or socket.

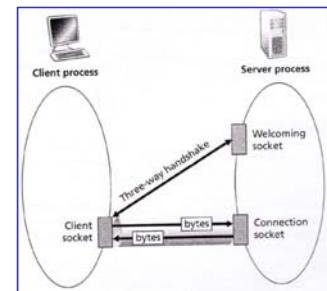
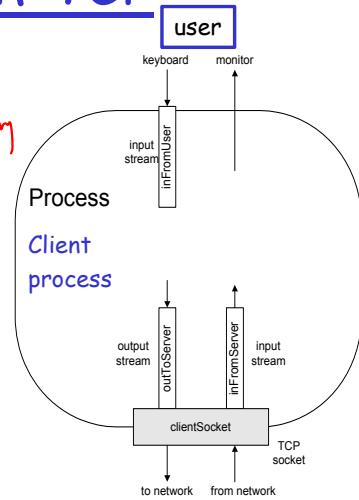


2: Application Layer 140

Socket programming with TCP

Example client-server app: ~~q1861n251gj~~

- 1) client reads line from standard input (`inFromUser stream`), sends to server via socket (`outToServer stream`)
- 2) server reads line from socket
- 3) server converts line to uppercase, sends back to client
- 4) client reads, prints modified line from socket (`inFromServer stream`)



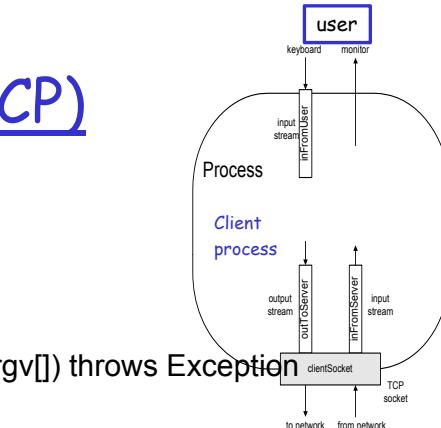
141

Example: Java client (TCP)

```
import java.io.*;
import java.net.*;
class TCPClient {

    public static void main(String argv[]) throws Exception {
        String sentence;
        String modifiedSentence;

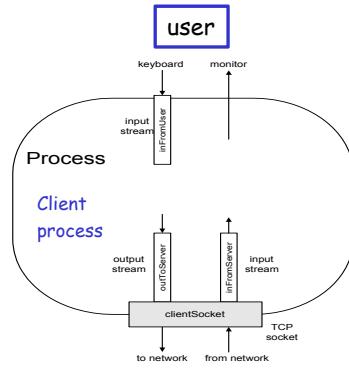
        Create input stream → BufferedReader inFromUser =
            new BufferedReader(new InputStreamReader(System.in));
        Create client socket, connect to server → Socket clientSocket = new Socket("hostname", 6789);
        Create output stream attached to socket → DataOutputStream outToServer =
            new DataOutputStream(clientSocket.getOutputStream());
    }
}
```



Example: Java client (TCP) cont.

```
Create  
input stream  
attached to socket ]→ BufferedReader inFromServer =  
new BufferedReader(new  
InputStreamReader(clientSocket.getInputStream()));  
  
sentence = inFromUser.readLine();  
Send line  
to server ]→ outToServer.writeBytes(sentence + '\n');  
  
Read line  
from server ]→ modifiedSentence = inFromServer.readLine();  
System.out.println("FROM SERVER: " + modifiedSentence);  
  
clientSocket.close();  
  
}  
}
```

2: Application Layer 143



Example: Java server (TCP)

```
import java.io.*;  
import java.net.*;  
  
class TCPServer {  
  
    public static void main(String argv[]) throws Exception  
    {  
        String clientSentence;  
        String capitalizedSentence;  
  
        Create  
        welcoming socket  
        at port 6789 ]→ ServerSocket welcomeSocket = new ServerSocket(6789);  
  
        Wait, on welcoming  
        socket for contact  
        by client ]→ while(true) {  
            Socket connectionSocket = welcomeSocket.accept();  
  
            Create input  
            stream, attached  
            to socket ]→ BufferedReader inFromClient =  
new BufferedReader(new  
InputStreamReader(connectionSocket.getInputStream()));
```

2: Application Layer 144

Example: Java server (TCP), cont

```
Create output stream, attached to socket → DataOutputStream outToClient =  
new DataOutputStream(connectionSocket.getOutputStream());  
  
Read in line from socket → clientSentence = inFromClient.readLine();  
  
capitalizedSentence = clientSentence.toUpperCase() + '\n';  
  
Write out line to socket → outToClient.writeBytes(capitalizedSentence);  
}  
}  
}  
} → End of while loop,  
loop back and wait for  
another client connection
```

2: Application Layer 145

Chapter 2: Application layer

- 2.1 Principles of network applications
- 2.2 Web and HTTP
- 2.3 FTP
- 2.4 Electronic Mail
 - ❖ SMTP, POP3, IMAP
- 2.5 DNS
- 2.6 P2P applications
- 2.7 Socket programming with TCP
- 2.8 **Socket programming with UDP**

2: Application Layer 146

Socket programming with UDP

UDP: no "connection" between client and server

- o no handshaking
- o sender explicitly attaches IP address and port of destination to each packet
- o server must extract IP address, port of sender from received packet

UDP: transmitted data may be received out of order, or lost

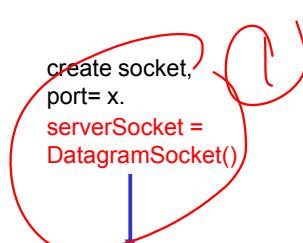
application viewpoint

UDP provides unreliable transfer of groups of bytes ("datagrams") between client and server

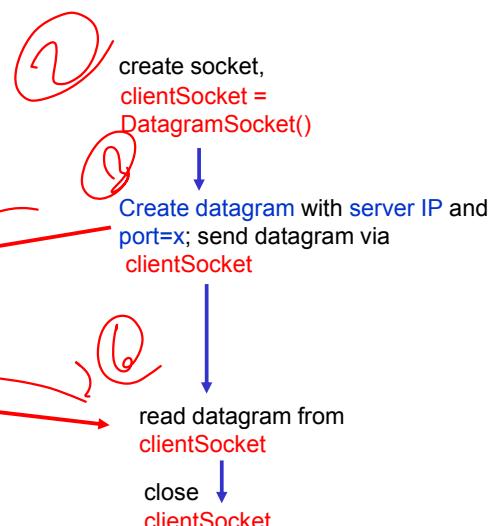
2: Application Layer 147

Client/server socket interaction: UDP

Server (running on hostid)

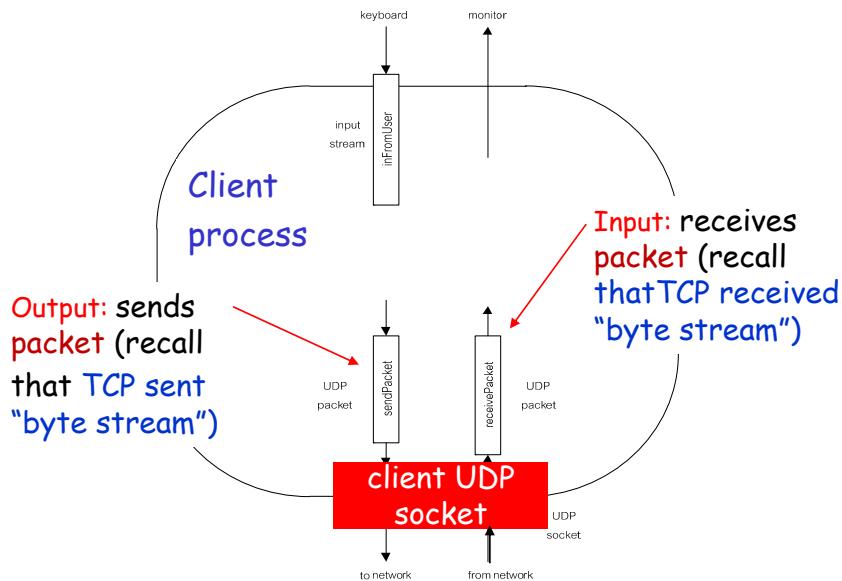


Client



2: Application Layer 148

Example: Java client (UDP)



2: Application Layer 149

Example: Java client (UDP)

```
import java.io.*;
import java.net.*; ประกอบด้วย DatagramSocket, InetAddress

class UDPClient {
    public static void main(String args[]) throws Exception
    {
        Input stream
        BufferedReader inFromUser =
            new BufferedReader(new InputStreamReader(System.in));
        Create client socket
        DatagramSocket clientSocket = new DatagramSocket();
        Translate hostname to IP address using DNS
        InetAddress IPAddress = InetAddress.getByName("hostname");
        byte[] sendData = new byte[1024];
        byte[] receiveData = new byte[1024];
        String sentence = inFromUser.readLine();
        sendData = sentence.getBytes();
    }
}
```

A handwritten note in red at the bottom right of the code snippet says "hostname → ip".

2: Application Layer 150

Example: Java client (UDP), cont.

```
Create datagram  
with data-to-send,  
length, IP addr, port] DatagramPacket sendPacket =  
new DatagramPacket(sendData, sendData.length, IPAddress, 9876);  
  
Send datagram  
to server] clientSocket.send(sendPacket);  
  
Read datagram  
from server] DatagramPacket receivePacket =  
new DatagramPacket(receiveData, receiveData.length);  
clientSocket.receive(receivePacket);  
  
String modifiedSentence =  
new String(receivePacket.getData());  
  
System.out.println("FROM SERVER:" + modifiedSentence);  
clientSocket.close();  
}  
}
```

2: Application Layer 151

Example: Java server (UDP)

```
import java.io.*;  
import java.net.*;  
  
class UDPServer {  
    public static void main(String args[]) throws Exception  
    {  
        Create  
        datagram socket  
        at port 9876] DatagramSocket serverSocket = new DatagramSocket(9876);  
  
        byte[] receiveData = new byte[1024];  
        byte[] sendData = new byte[1024];  
  
        while(true)  
        {  
            Create space for  
            received datagram] DatagramPacket receivePacket =  
            new DatagramPacket(receiveData, receiveData.length);  
            Receive  
            datagram] serverSocket.receive(receivePacket);
```

2: Application Layer 152

Example: Java server (UDP), cont

```
String sentence = new String(receivePacket.getData());
Get IP addr
port #, of
sender
    InetAddress IPAddress = receivePacket.getAddress();
    int port = receivePacket.getPort();

    String capitalizedSentence = sentence.toUpperCase();

    sendData = capitalizedSentence.getBytes();

Create datagram
to send to client
    DatagramPacket sendPacket =
        new DatagramPacket(sendData, sendData.length, IPAddress,
                           port);

Write out
datagram
to socket
    }
}

End of while loop,
loop back and wait for
another datagram
```