

Application Layer

In this lesson...

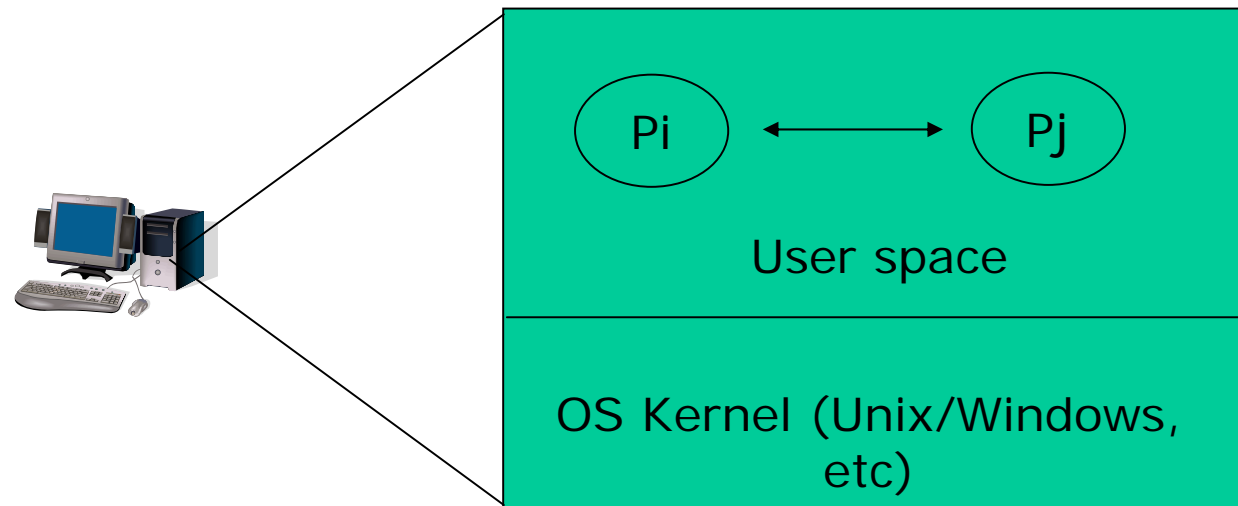
- Applications of networked computing and its challenges
 - ★ The Web(HTTP), FTP, DNS, Email
- Client/Server Programming
 - ★ Socket programming
- Programming Assignments



Kurose Text Book, Chapter 2.

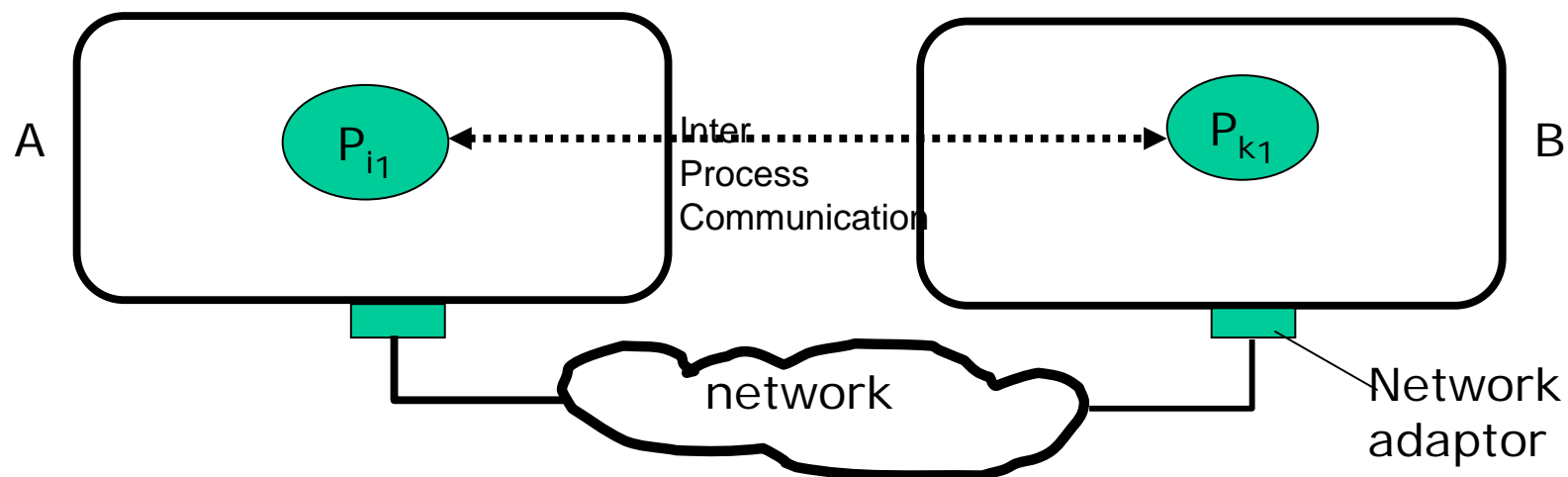
Basic Concepts: Inter-process communication

- What is a process?
 - ★ An instance of a program
- What is inter-process communication?
 - ★ Unique process ID
 - ★ OS's IPC mechanism



Network Communication

- Inter-process communication between different end systems.



- **Client Process:** process that initiates the communication
- **Server Process:** process that waits to be contacted



A network messaging application

1. Bob places a connect request to Alice@B via P_{i1}
 - ★ P_{i1} : locates Alice's B's **network address**, attaches it to a **Message** containing the connection request
 - ★ A sends the '**Message**' to B's address via the network
2. Network delivers the '**Message**' to B
3. B receives and accepts the **Message**, then **passes it to P_{k1}** where Alice is.
4. Alice *accepts* the request
5. P_{k1} returns the acceptance in a **Message** to P_{i1}
6. Network delivers the Message to A, which passes on the message to P_{i1}
7. P_{i1} invites Bob to key in his greeting, which he does.
8. The Greeting is then passed back to Alice@B in a **Message** via the network
9. B receives and passes on the **Message** to P_{k1} and then to Alice
10. Alice replies
11.



Some issues

- What is the **message**?
- There are some rules governing the communication between Alice and Bob, between P_{i1} and P_{k1} - **Protocol**
- How does P_{i1} locate the network **address of B**?
- How does the network deliver the message?
 - ★ How could it be done reliably?
- How does B know there is a message for him?
- How does B know there the **message is meant for P_{k1}** ?
- How does P_{k1} know the **meaning (semantic)** of the message?



Applications of Networked Computing

- Network Applications (Eg. Internet Applications)
 - ★ telnet and SSH
 - ★ E-mail
 - ★ ftp
 - ★ The web
 - ★ Instant messaging
 - ★ Video conferencing
 - ★ Streaming video
 - ★ Internet radio
 - ★ Internet telephony
 - ★ P2P file sharing
 - ★ Multi-player network games



- Reliable data transfer
 - ★ some apps (e.g., audio) can tolerate some loss
 - ★ other apps (e.g., file transfer, telnet) require 100% reliable data transfer
- Bandwidth
 - ★ some apps (e.g., multimedia, needs different encoding scheme for different bandwidth) require minimum amount of bandwidth to be “effective”
 - ★ other apps (“elastic apps”) make use of whatever bandwidth they get
- Timing
 - ★ some apps (e.g., Internet telephony, interactive games) require low delay to be “effective”
- QoS in packet switched network



Application Architecture

- Application architecture vs. network architecture
- Application developer's view: Network architecture is fixed and provides a specific set of services to applications
- One of the goal of Internet Architecture is to hide all the details of the physical layout and protocol complexities of the internet from the applications.
 - ★ Client/server architecture
 - Server farm -> powerful virtual server
 - Web, file transfer, remote login, e-mail
 - ★ P2P architecture (peers)
 - P2P file sharing, eg. Gnutella (www.gnutella.com)
 - Scalability
 - ★ Hybrid
 - Eg. Napster – P2P MP3 sharing, contact central server for querying about the availability of MP3. (www.napster.com)
 - Instant messaging
 - Skype internet telephony



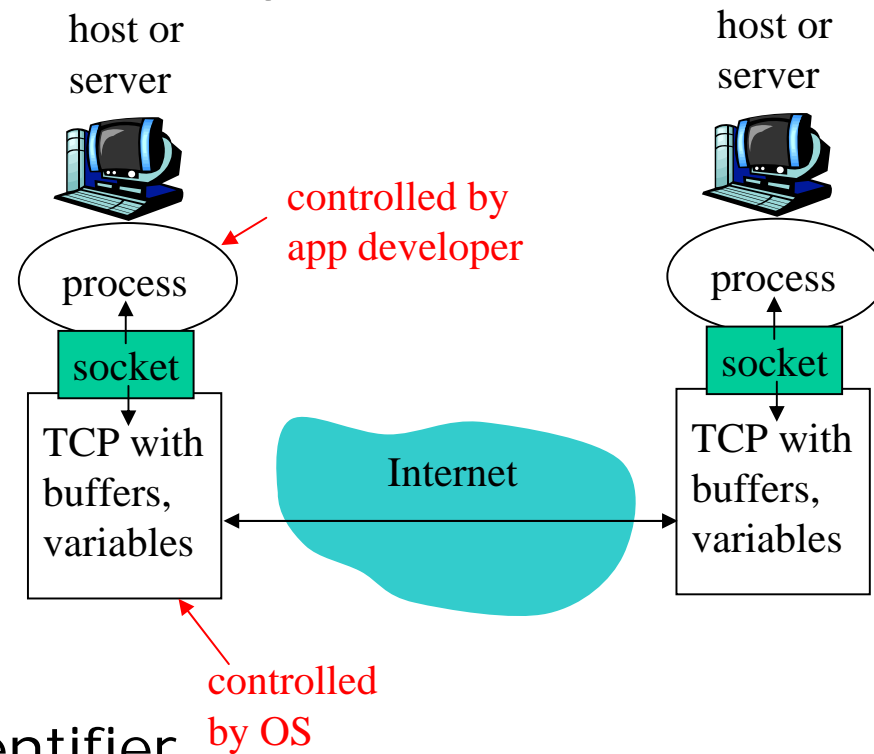
Processes Communicates

- Client Process:
 - ★ The process that initiates the communication in a communication session. (eg Browser)
- Server Process:
 - ★ The process that waits to be connected to begin the session. (eg WebServer)



- Application programming Interface – Socket
 - ★ Interface between the application layer and transport layer

Socket: A combination of IP address and port number.



- Addressing process
 - ★ Host address + process identifier
 - ★ Eg. IP address + port number
 - Eg HTTP – port 80, SMTP – port 25, ftp – port 21

Internet's Transport Layer Services

- Connection oriented service
 - ★ TCP (Applications: email, remote terminal access, web, file transfer)
 - ★ Transport layer control information is exchanged first (handshaking procedure) to **create a connection** before the application level messages begin to flow.
 - ★ Full-Duplex Connection
 - ★ Reliable Transport service
- Connection Less Service
 - ★ UDP (Applications: Internet Telephony, Streaming Audio/video, Remote file server)
 - ★ No connection, no handshaking, unreliable.

Each TCP connection is uniquely identified by a **socket pair**.



Application Layer Protocols: HTTP

- On top of TCP
- HTTP is stateless (Note: - TCP maintains connection state, application layer is not maintaining the application/user state)
 - ★ server maintains no information about past client requests
- Default HTTP port: 80

➤ RTT – Round Trip Time



Application Layer Protocols: HTTP

- Persistence vs non-persistence
 - ★ Non-persistence
 - Server terminates connection after transferring the file/object
 - HTTP 1.0 (RFC 1945)
 - New connection is established each time (new set of TCP variables and buffers)



Application Layer Protocols: HTTP

➤ Persistence vs non-persistence

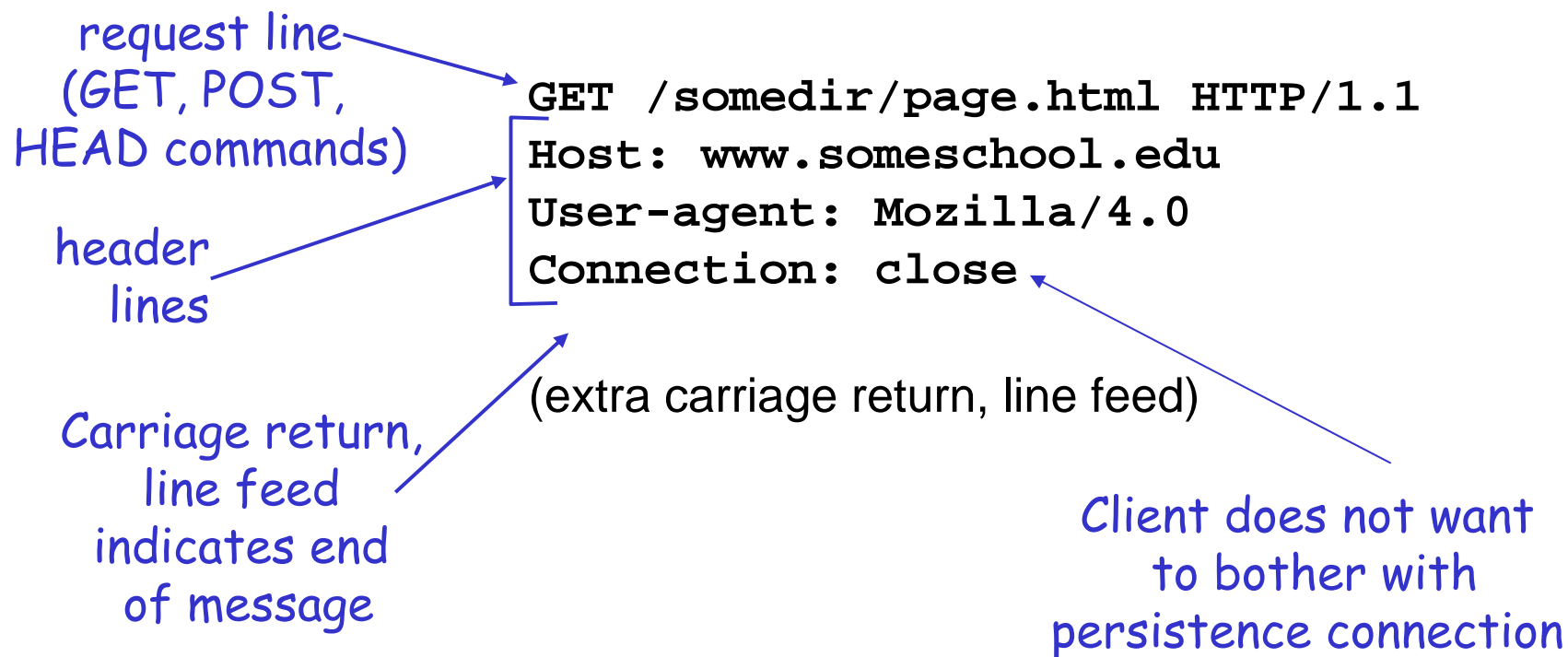
★ Persistence:

- Server leaves the TCP connection open after sending the response. Subsequent requests between the same client and server will use the same connection.
- **Without pipelining**
- **With pipelining**
- Default HTTP 1.1 (RFC 2616): persistence with pipelining

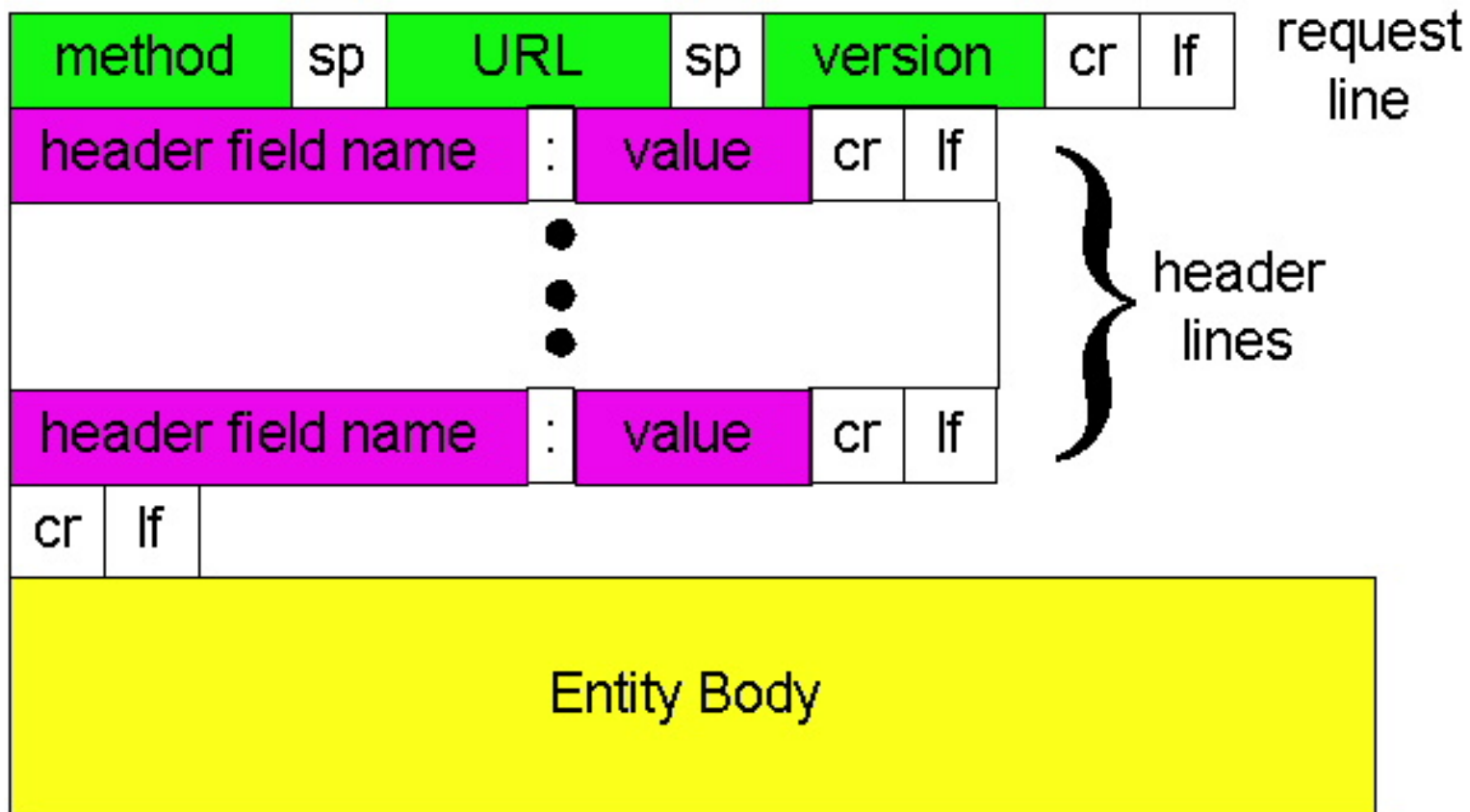


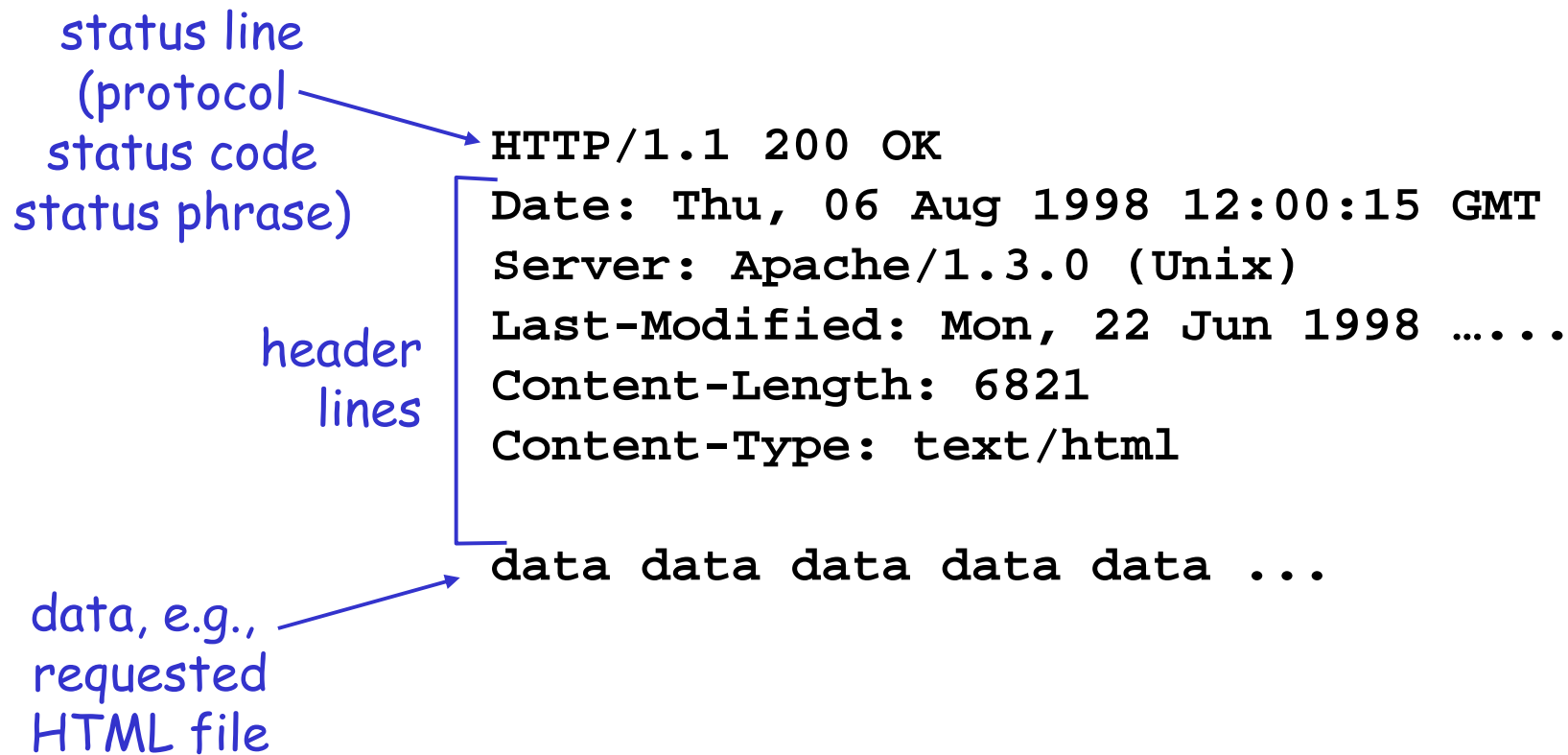


HTTP Request



HTTP Request





DEMO: Telnet, http, Curl



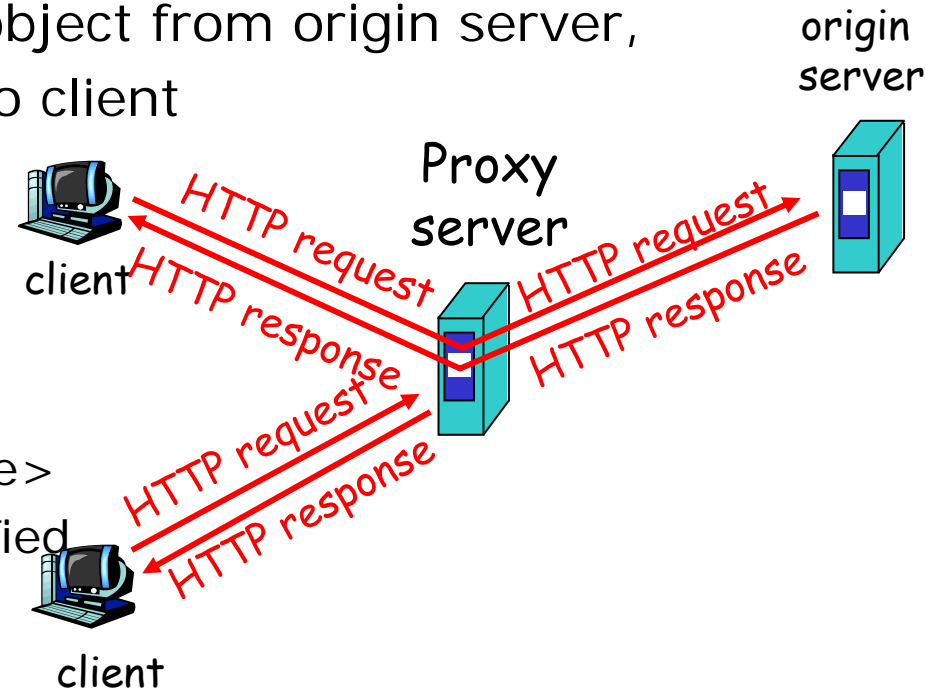
User State Management - Cookies

- HTTP server is stateless.
- HTTP uses cookies to manage user identity
- Cookies have four components.
 - ★ HTTP response header. (by web server upon initial request from client)
 - Set-cookie: 1232341
 - ★ HTTP request header. (every subsequent request by client)
 - Cookie: 1232341
 - ★ Cookie file is maintained by the browser (client)
 - ★ Back-end database at the web site (web server)



Web caches (proxy server)

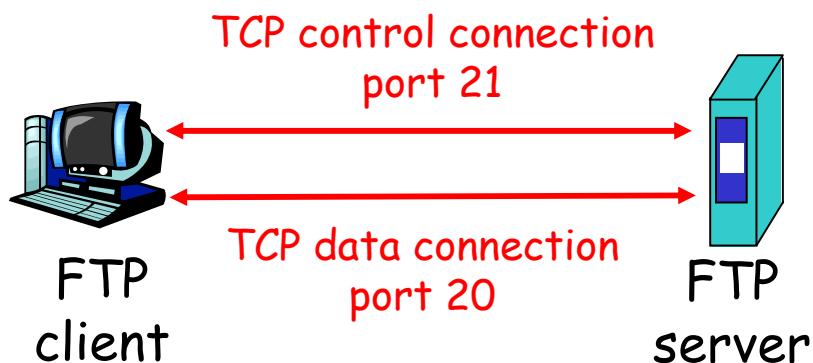
- **Goal:** satisfy client request without involving origin server.
Improve response time; reduce traffic.
 - ★ user sets browser: Web accesses via cache
 - ★ browser sends all HTTP requests to cache
 - object in cache: cache returns object
 - else cache requests object from origin server, then returns object to client



- **Conditional GET**
REQ: If-modified-since: <date>
RES: HTTP/1.0 304 Not Modified

Application Layer Protocols: FTP

- FTP client contacts FTP server at port 21, specifying TCP as transport protocol
- Client obtains authorization 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, data connection)** to client. (rfc: 2428)
- 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"** about the user throughout the **session**: current directory, earlier authentication



FTP: Demo

Ref: rfc2428?

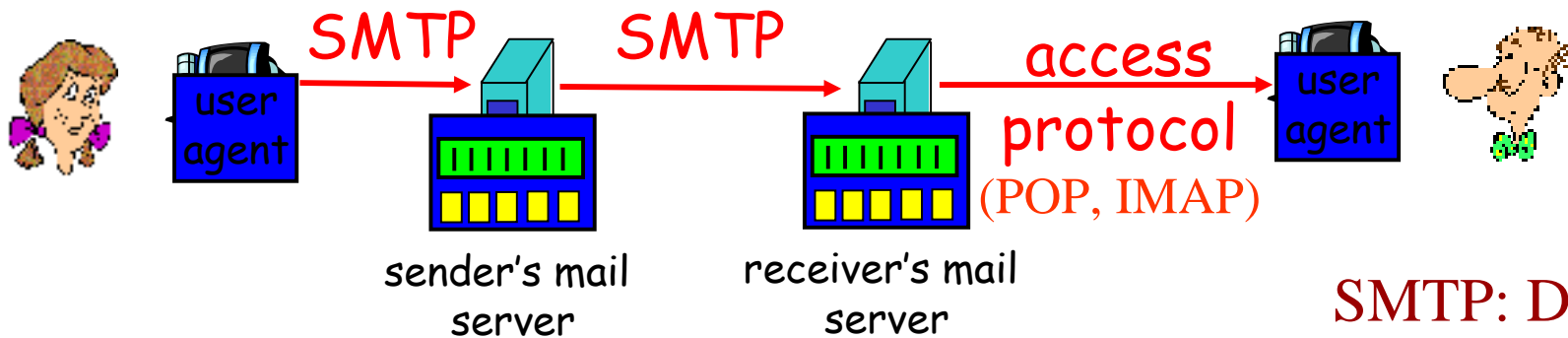
Application Layer Protocols for E-mail

- Three major components:
 - ★ user agents
 - ★ mail servers
 - ★ simple mail transfer protocol: SMTP (port 25)
- SMTP: delivery/storage to receiver's server
- Mail access protocol: retrieval from server
 - ★ POP: Post Office Protocol [RFC 1939]
 - authorization (agent <-->server) and download
 - ★ IMAP: Internet Mail Access Protocol [RFC 1730]
 - more features (more complex)
 - manipulation of stored msgs on server
 - ★ HTTP: Hotmail , Yahoo! Mail, etc.



Email - Scenario: Alice sends message to Bob

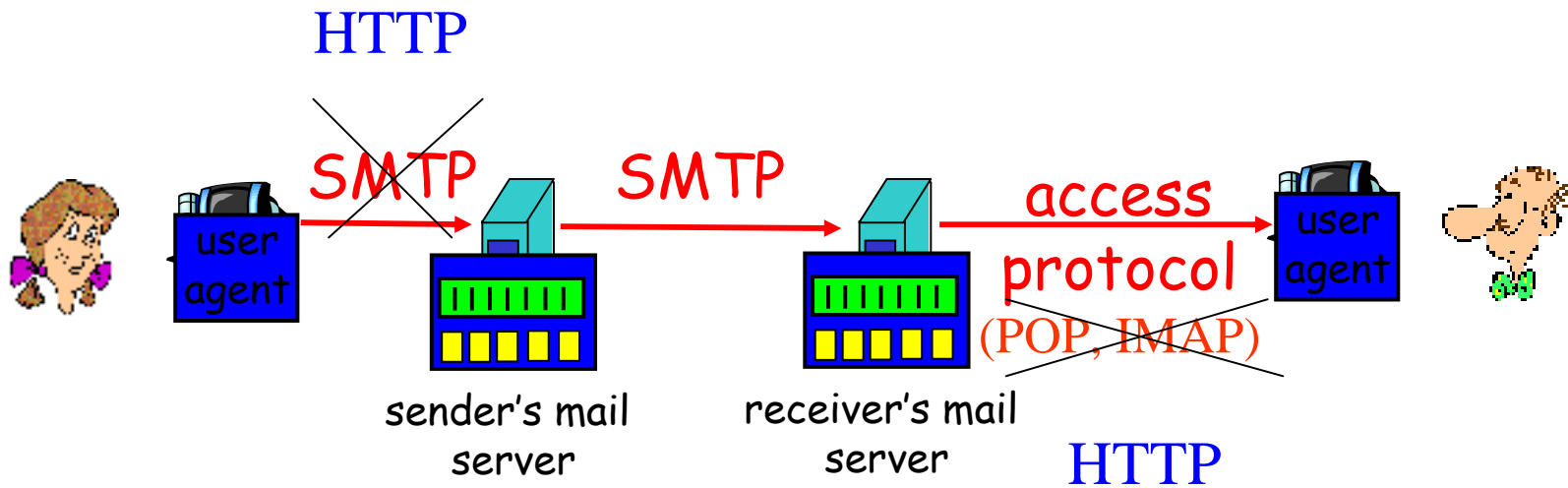
- Alice uses UA to compose message and "to" bob@someschool.edu
- Alice's UA sends message to her mail server; message placed in message queue
- Client side of SMTP opens TCP connection with Bob's mail server
- SMTP client sends Alice's message over the TCP connection
- Bob's mail server places the message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



SMTP: Demo

Image Source: Chapter 2 - Kurose& Ross

Web-based email



Push vs Pull protocol

➤ PULL

- ★ TCP connection is initiated by the machine that wants to receive a file

- ★ Q: Egs.?

➤ PUSH

- ★ TCP connection is initiated by the machine that wants to send a file

- ★ Q: Egs.?



Client-Server Applications Development

- Socket programming
 - ★ **Socket**: a door between application process and end-end-transport protocol (UDP or TCP)
 - ★ **TCP service**: reliable transfer of **bytes** from one process to another
 - **server process** must **first** be running
 - server must have created socket (door) that welcomes client's contact
 - creating client-local TCP socket
 - specifying IP address, port number of server process
 - When client creates socket: client TCP **establishes connection** to server TCP
 - ★ **UDP service**: unreliable transfer of **bytes** from one process to another
 - **no "connection"** between client and server
 - no handshaking
 - sender explicitly attaches IP address and port of destination to each packet
 - server must extract IP address, port of sender from received packet
 - transmitted data may be received out of order, or lost



- Java Socket API
 - ★ **Socket:**
<http://java.sun.com/j2se/1.5.0/docs/api/java/net/Socket.html>
 - ★ **ServerSocket:**
<http://java.sun.com/j2se/1.5.0/docs/api/java/net/ServerSocket.html>
 - ★ **MultiCastSocket:**
<http://java.sun.com/j2se/1.5.0/docs/api/java/net/MulticastSocket.html>
 - ★ **DatagramSocket:**
<http://java.sun.com/j2se/1.5.0/docs/api/java/net/DatagramSocket.html>
 - ★ **DatagramPacket:**
<http://java.sun.com/j2se/1.5.0/docs/api/java/net/DatagramPacket.html>

- **TCP Socket program demo**
- **UDP Socket program demo**
- **Simple HTTP 1.0 web Server demo**



➤ DNS Services

- ★ Hostname to IP address translation
- ★ Host aliasing
 - Canonical and alias names
- ★ Mail server aliasing
- ★ Load distribution
 - Replicated Web servers: set of IP addresses for one canonical name

Popular Implementation: **BIND**



➤ DNS Components

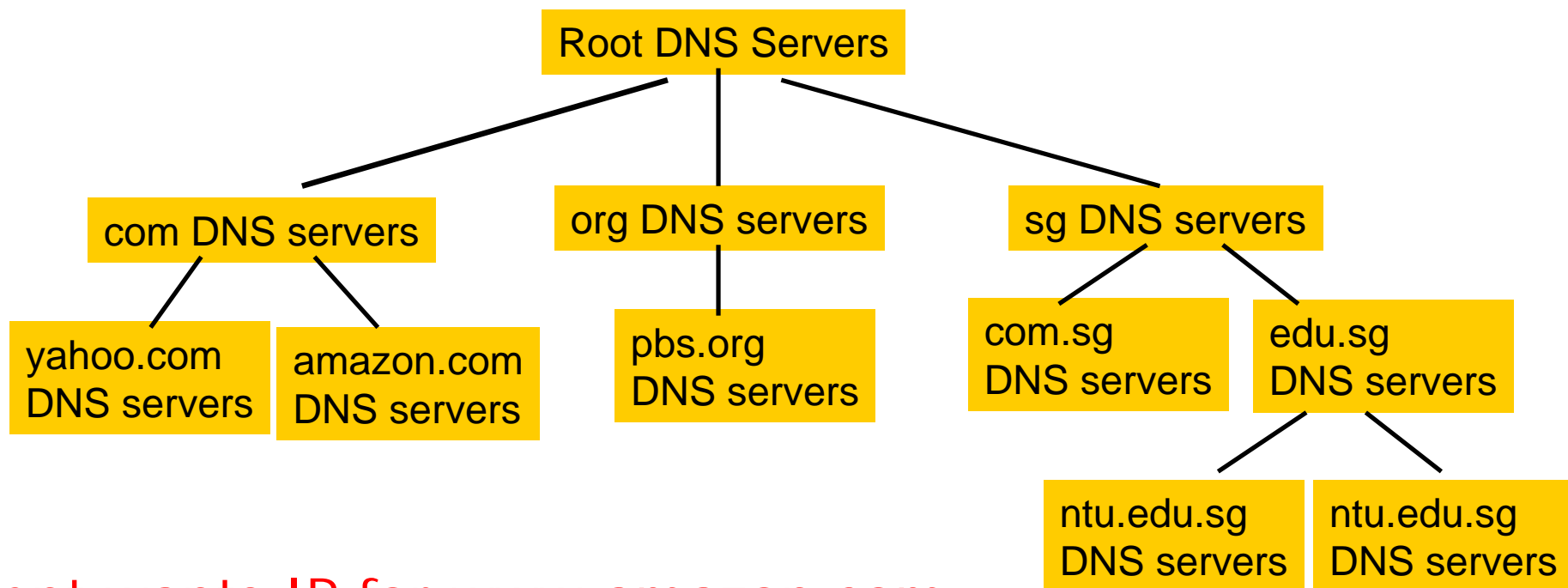
- ★ **distributed database** implemented in hierarchy of many name servers
- ★ **Application on top of UDP/TCP**: host, routers, name servers to communicate to **resolve** names (address/name translation)

➤ Why not centralize DNS?

- single point of failure
- traffic volume
- distant centralized database
- maintenance
- doesn't *scale!*



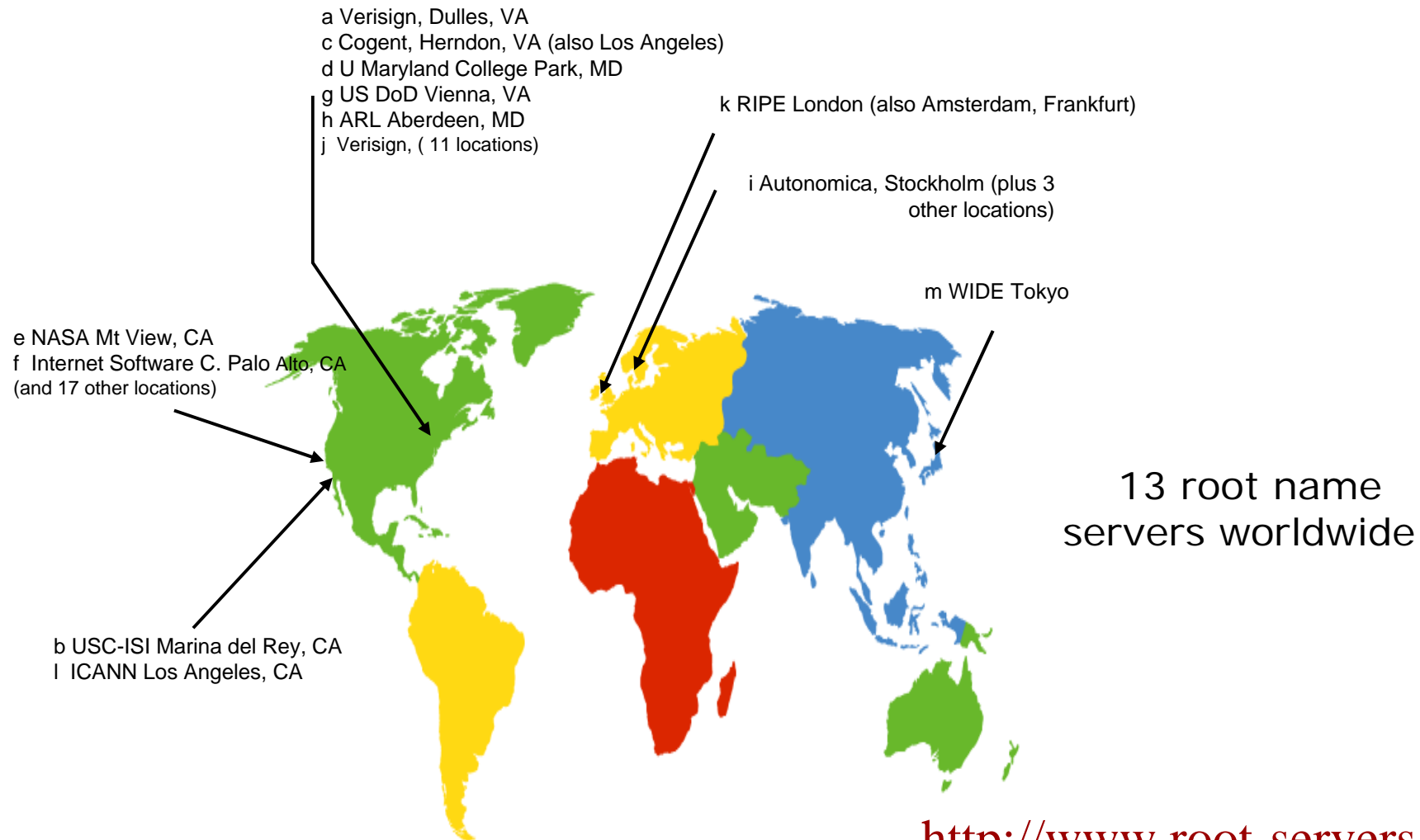
DNS: Distributed, Hierarchical Database



Client wants IP for www.amazon.com:

- 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





<http://www.root-servers.org/>

DNS: 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.
 - ★ 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 and mail).
 - ★ Can be maintained by organization or service provider



DNS: Local Name Servers

- Does not strictly belong to hierarchy
- Each ISP (residential ISP, company, university) has one.
 - ★ Also called “**default name server**”
- When a host makes a DNS query, query is sent to its local DNS server
 - ★ Acts as a proxy, forwards query into hierarchy.



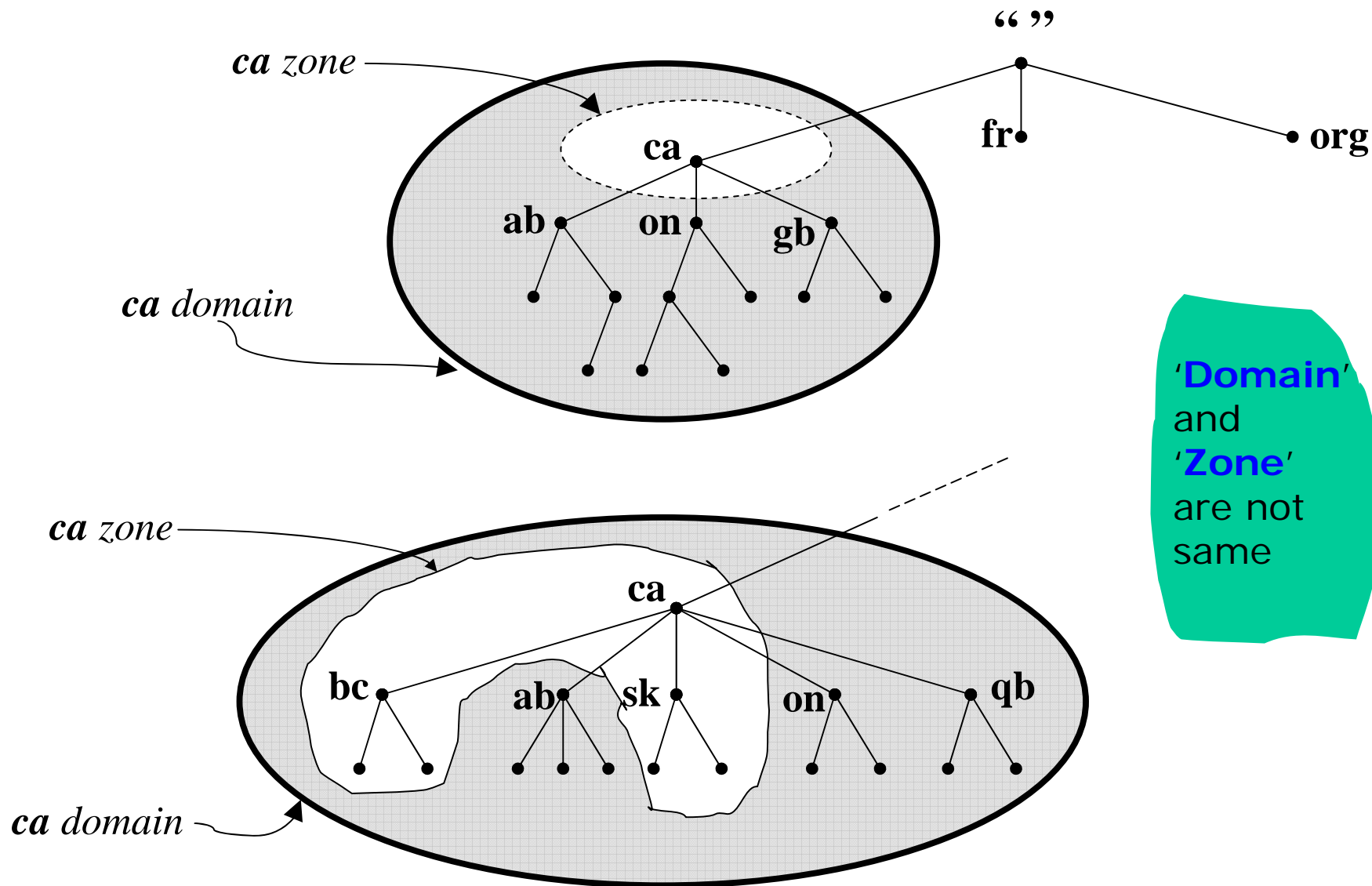
DNS: caching and updating records

- once (any) name server learns mapping, it **caches** mapping
 - ★ **cache entries timeout** (disappear) after some time
 - ★ TLD servers typically cached in local name servers
 - Thus root name servers not often visited
- update/notify mechanisms under design by IETF
 - ★ RFC 2136
 - ★ <http://www.ietf.org/html.charters/dnsind-charter.html>



DNS – Domains and Zones

[Source: Fig 2.8 & 2.9 of DNS and BIND by Paul Albitz & Cricket Liu]



In reality: name servers have data about a part of the Domain Name Space. (called 'Zone')

'Domain' and 'Zone' are not same



DNS records

DNS: distributed db storing **resource records** (RR)

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

➤ Type=A

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

➤ Type=NS

- ★ **name** is domain (e.g. foo.com)
- ★ **value** is hostname of authoritative name server for this domain (**primary & secondary**)

➤ 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

➤ Type=MX

- ★ **value** is name of mailserver associated with **name**

 Q: Use of secondary name server?

DNS Demo

DNS records

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

➤ Type=SOA

★ **name** is domain (e.g. foo.com)

★ **value** is hostname of authoritative name server for this domain and other info.

★ **ttl** – time-to-live when cached by others

➤ Type=PTR

★ **name** is IP address (in-addr.arpa. Domain)

★ **value** is host name

➤ Q: What is 'Zone' Transfer?



DNS Demo

PROGRAMMING ASSIGNMENTS

1. Modify the simple web server (given in text book) to process the HTTP request.
 - Call server-side scripts
 - Handle GET and POST requests
2. Develop a Web Cache Proxy
 - Send from cache 'if available and if it is new'
 - Otherwise, get new from the web site, cache it and send

