20MCA104
ADVANCED COMPUTER NETWORKS

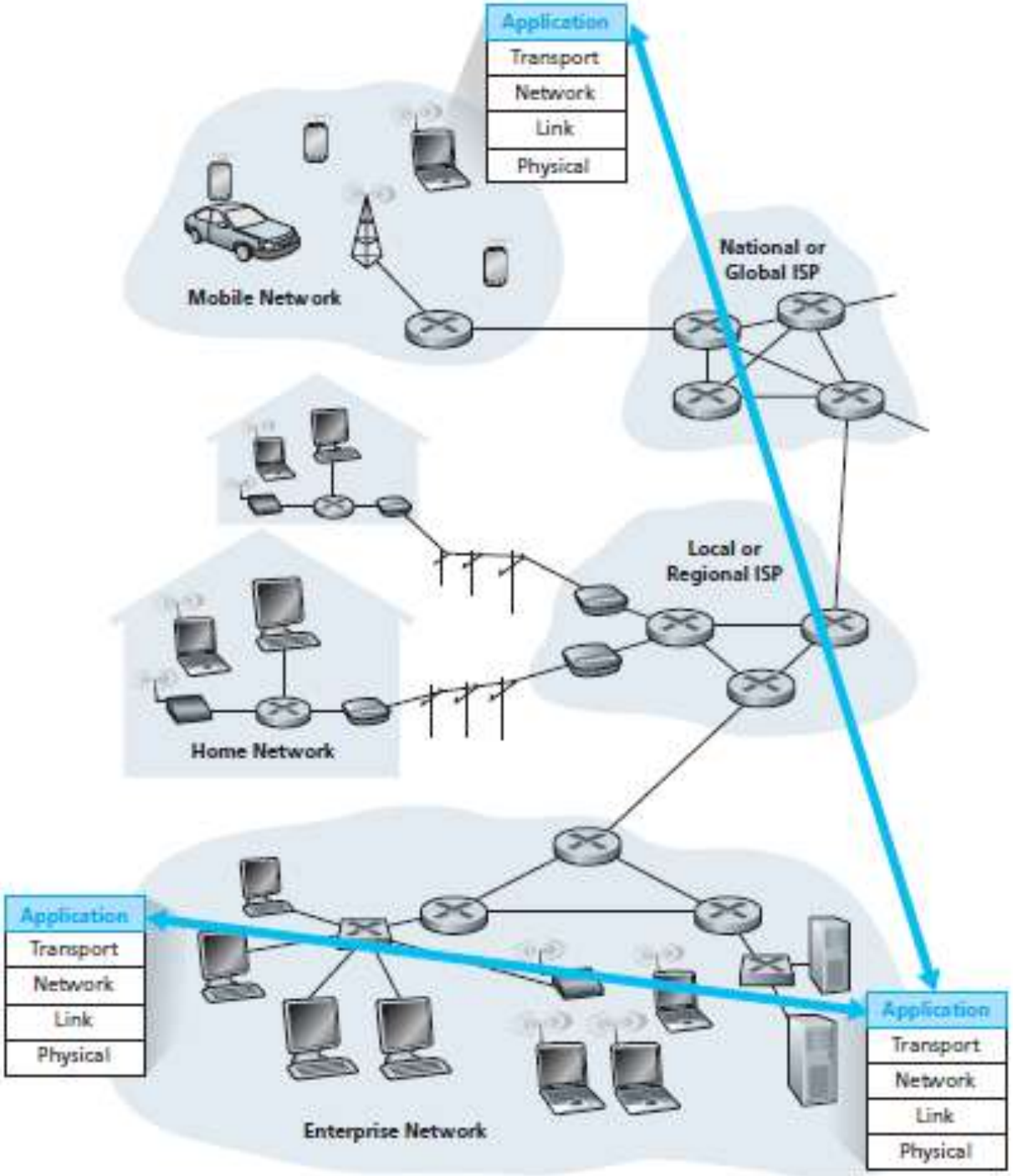
Module 1 – Part 2

APPLICATION LAYER

Introduction

Application layer

- Provides services to the user.
- Communication is provided using a logical connection.
  - two application layers assume that there is an imaginary direct connection through which they can send and receive messages.


Application
Transport
Network
Link
Physical
Mobile Network
National or
Global ISP
Local or
Regional ISP
Home Network
Application
Transport
Network
Link
Physical
Enterprise Network
Application
Transport
Network
Link
Physical

Why application layer is different from other layers?

Why application layer is different from other layers?

- It is the highest layer in the suite.
- The protocols in this layer do not provide services to any other protocol in the suite;
  - they only receive services from the protocols in the transport layer.
- Protocols can be removed from this layer easily.
- New protocols can be also added to this layer.

Principles of Network Applications

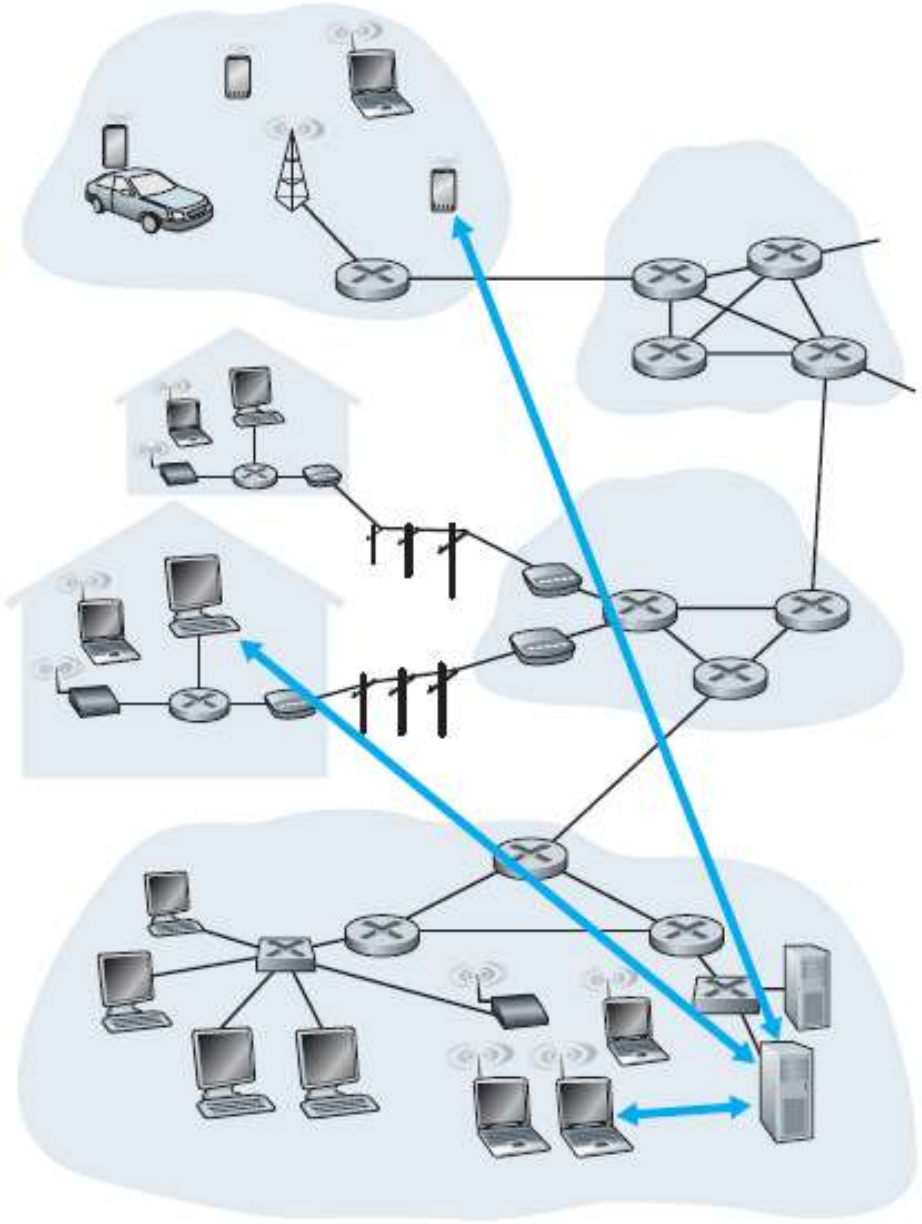
- Writing programs that run on different end systems and communicate with each other over the network.

Application-Layer Paradigms

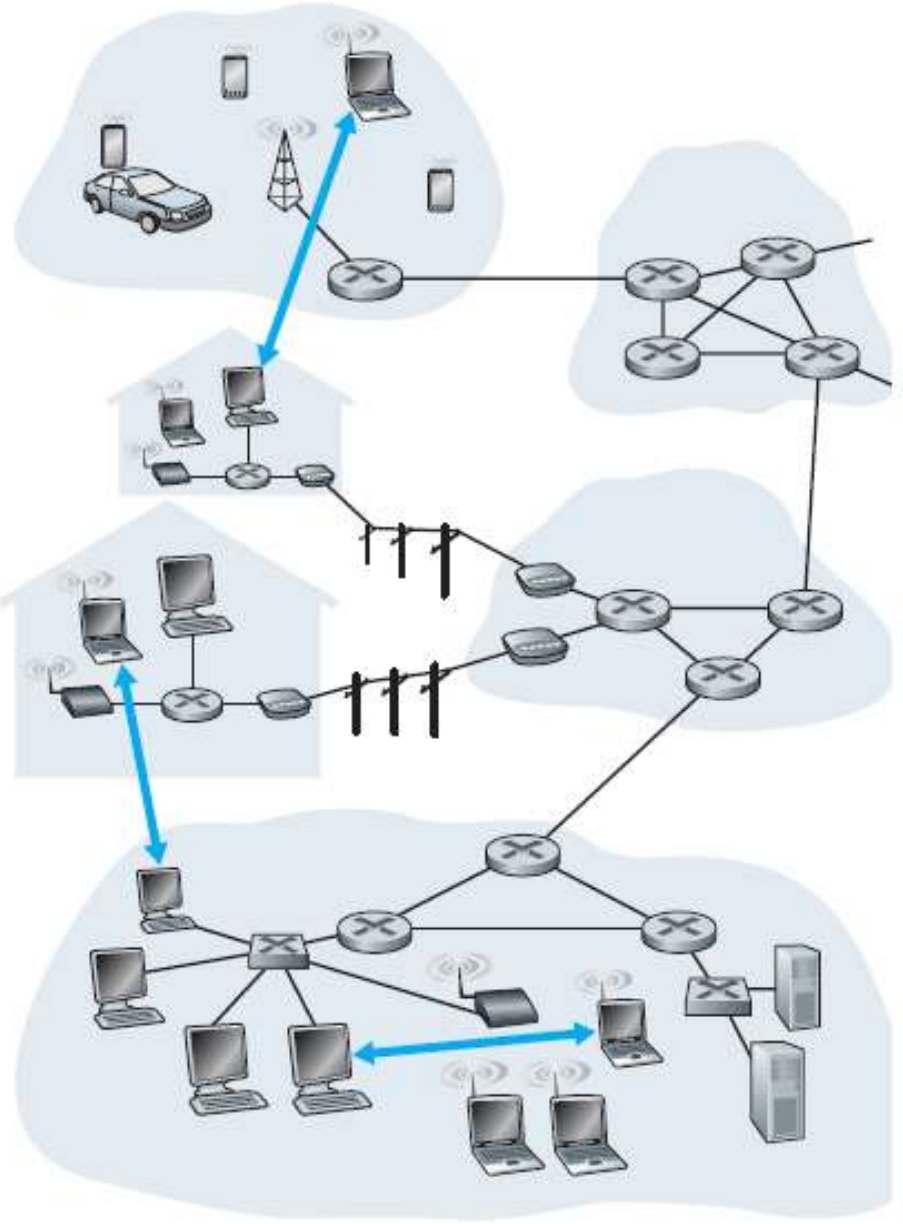
- Client-server paradigm
- Peer-to-peer paradigm

Network Application Architectures

- The **application architecture** is designed by the application developer and dictates how the application is structured over the various end systems.
- An application developer chooses the application architecture from the two architectural paradigms used in modern network applications:
  - The client-server architecture or
  - The peer-to-peer (P2P) architecture



a. Client-server architecture



b. Peer-to-peer architecture

Processes Communicating

- A process can be thought of as a program that is running within an end system.
- Processes on two different end systems communicate with each other by exchanging **messages** across the computer network.
  - A sending process creates and sends messages into the network;
  - A receiving process receives these messages and possibly responds by sending messages back.

Client and Server Processes

- In the context of a communication session between a pair of processes,
  - *The process that initiates the communication is labeled as the **client**.*
  - *The process that waits to be contacted to begin the session is the **server**.*

- Example
  - In the web :
    - A browser process initializes contact with a web server process;
    - Client - browser process
    - Server - web server process
  - In P2P file sharing :
    - Peer A asks peer B to send a specific file,
    - Client - Peer A
    - Server - Peer B

Socket

- Any message sent from one process to another must go through the underlying network.
- A process sends messages into, and receives messages from, the network through a software interface called a **socket.**
- It is also referred to as the **Application Programming Interface (API)** between the application and the network.

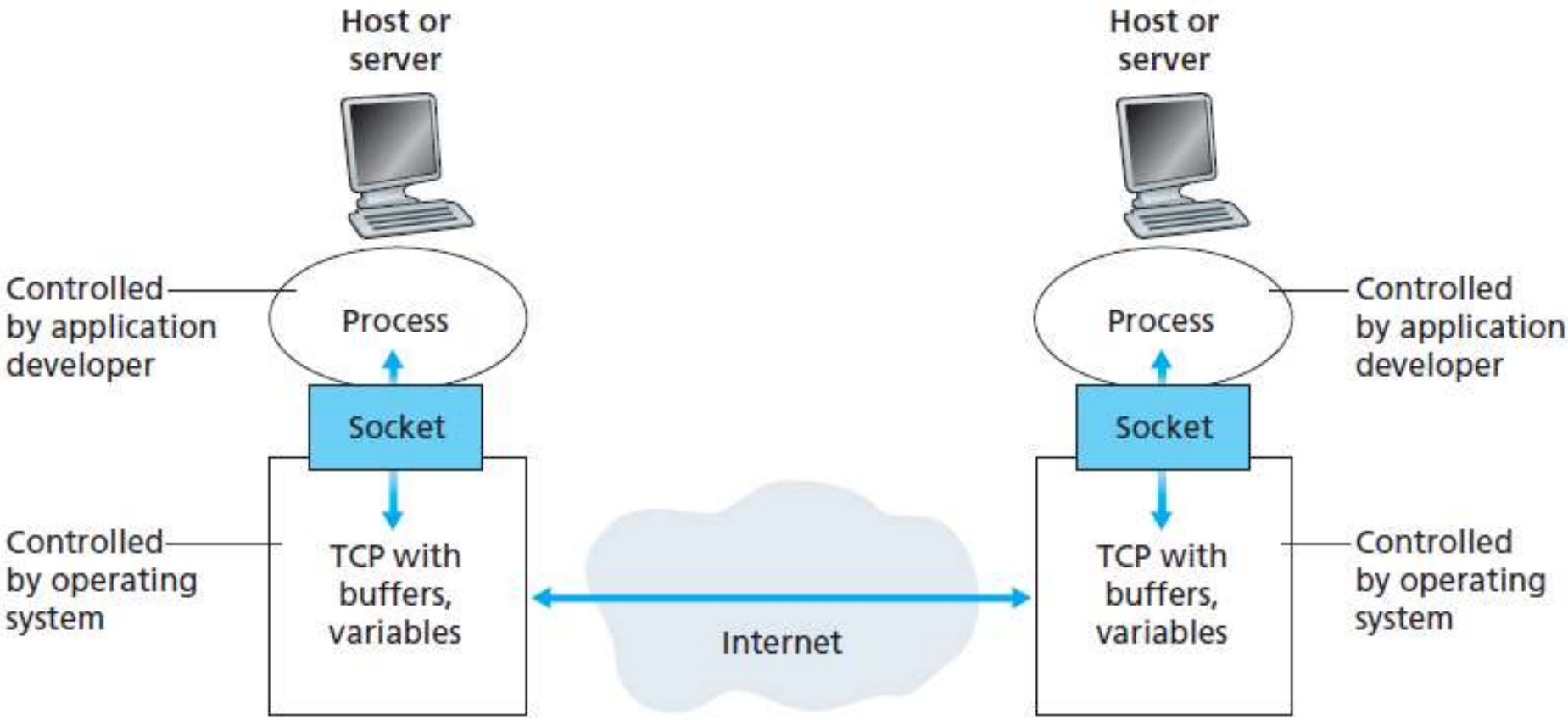


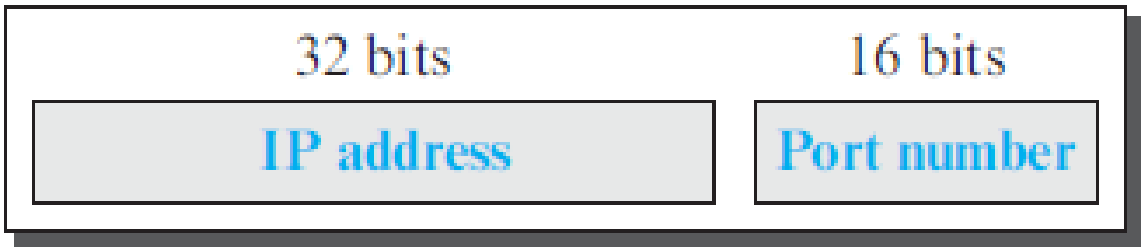
Figure ♦ Application processes, sockets, and underlying transport protocol

Socket Addresses

- A socket address should define
  1. The computer on which a client or a server is running.
  2. The application program running on the client or the server.

Socket Addresses

- A socket address should define
  1. The computer on which a client or a server is running.
  2. The application program running on the client or the server.
- That is, a socket address should be a combination of an **IP address** and a **port number**.



Socket Address

Transport Services Available to Applications

- Reliable data transfer
  - No loss applications
  - Loss-tolerant applications
- Throughput
  - Bandwidth-sensitive applications
  - Elastic applications
- Timing
  - Interactive real-time applications
  - Non-real-time applications
- Security
  - Confidentiality
  - Data integrity
  - End-point authentication

Application	Data Loss	Throughput	Time-Sensitive
File transfer/download	No loss	Elastic	No
E-mail	No loss	Elastic	No
Web documents	No loss	Elastic (few kbps)	No
Internet telephony/ Video conferencing	Loss-tolerant	Audio: few kbps–1Mbps Video: 10 kbps–5 Mbps	Yes: 100s of msec
Streaming stored audio/video	Loss-tolerant	Same as above	Yes: few seconds
Interactive games	Loss-tolerant	Few kbps–10 kbps	Yes: 100s of msec
Instant messaging	No loss	Elastic	Yes and no

Figure ♦ Requirements of selected network applications

Internet transport protocols services

TCP service:

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

UDP service:

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

Application	Application-Layer Protocol	Underlying Transport Protocol
Electronic mail	SMTP [RFC 5321]	TCP
Remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP [RFC 2616]	TCP
File transfer	FTP [RFC 959]	TCP
Streaming multimedia	HTTP (e.g., YouTube)	TCP
Internet telephony	SIP [RFC 3261], RTP [RFC 3550], or proprietary (e.g., Skype)	UDP or TCP

Figure ♦ Popular Internet applications, their application-layer protocols, and their underlying transport protocols

Securing TCP

TCP & UDP

- no encryption
- cleartext passwords are sent into socket
  - travels over Internet in cleartext.

Secure Sockets Layer (SSL)

- provides encrypted TCP connection
- data integrity
- end-point authentication

SSL is at app layer

- Apps use SSL libraries, which "talk" to TCP

SSL socket API

- cleartext passwords sent into SSL socket, encrypted, and travels over Internet.

Application-Layer Protocols

- An **application-layer protocol** defines how an application's processes, running on different end systems, pass messages to each other.

Application layer protocol defines

1. Types of messages exchanged,
   - E.g. Request, response
2. Message syntax:
   - What fields in messages & how fields are delineated
3. Message semantics
   - Meaning of information in fields
4. Rules
   - For when and how processes send & respond to messages

Open protocols:

- Defined in RFCs
- Allows for interoperability
- E.g. HTTP, SMTP

Proprietary protocols:

- E.g. Skype

The Web and HTTP

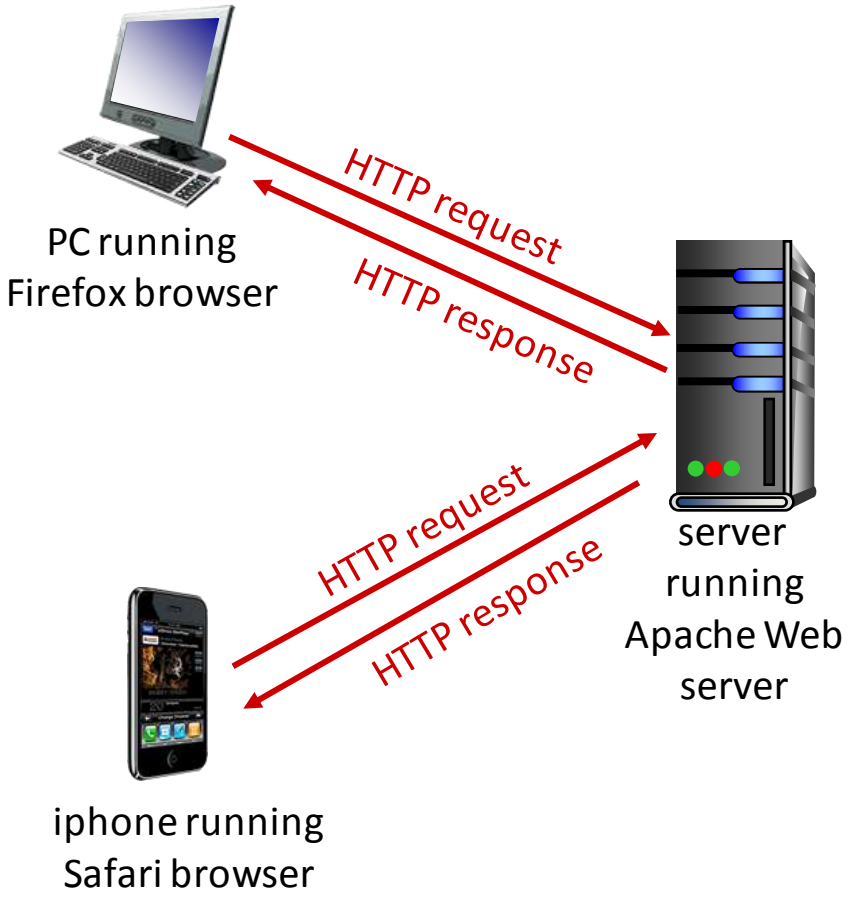
- World Wide Web [Tim-Berners-Lee 1994]
- Web operates *on demand.*
- *Web page* consists of *objects.*
  - Object can be HTML file, JPEG image, Java applet, audio file,…
  - Web page consists of *base HTML-file* which includes *several referenced objects.*
  - Each object is addressable by a *URL,* e.g.,

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

host name | path name

HTTP: HyperText Transfer Protocol

- Web's application layer protocol
- Client/server model
  - *client:* browser that requests, receives, (using HTTP protocol) and "displays" web objects
  - *server:* Web server sends (using HTTP protocol) objects in response to requests.



HTTP Overview

Uses TCP:

- Client initiates TCP connection (creates socket) to server, port 80
- Server accepts TCP 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.

HTTP connections

Non-persistent HTTP

- At most one object sent over TCP connection.
  - Connection then closed.
- Downloading multiple objects required multiple connections.

Persistent HTTP

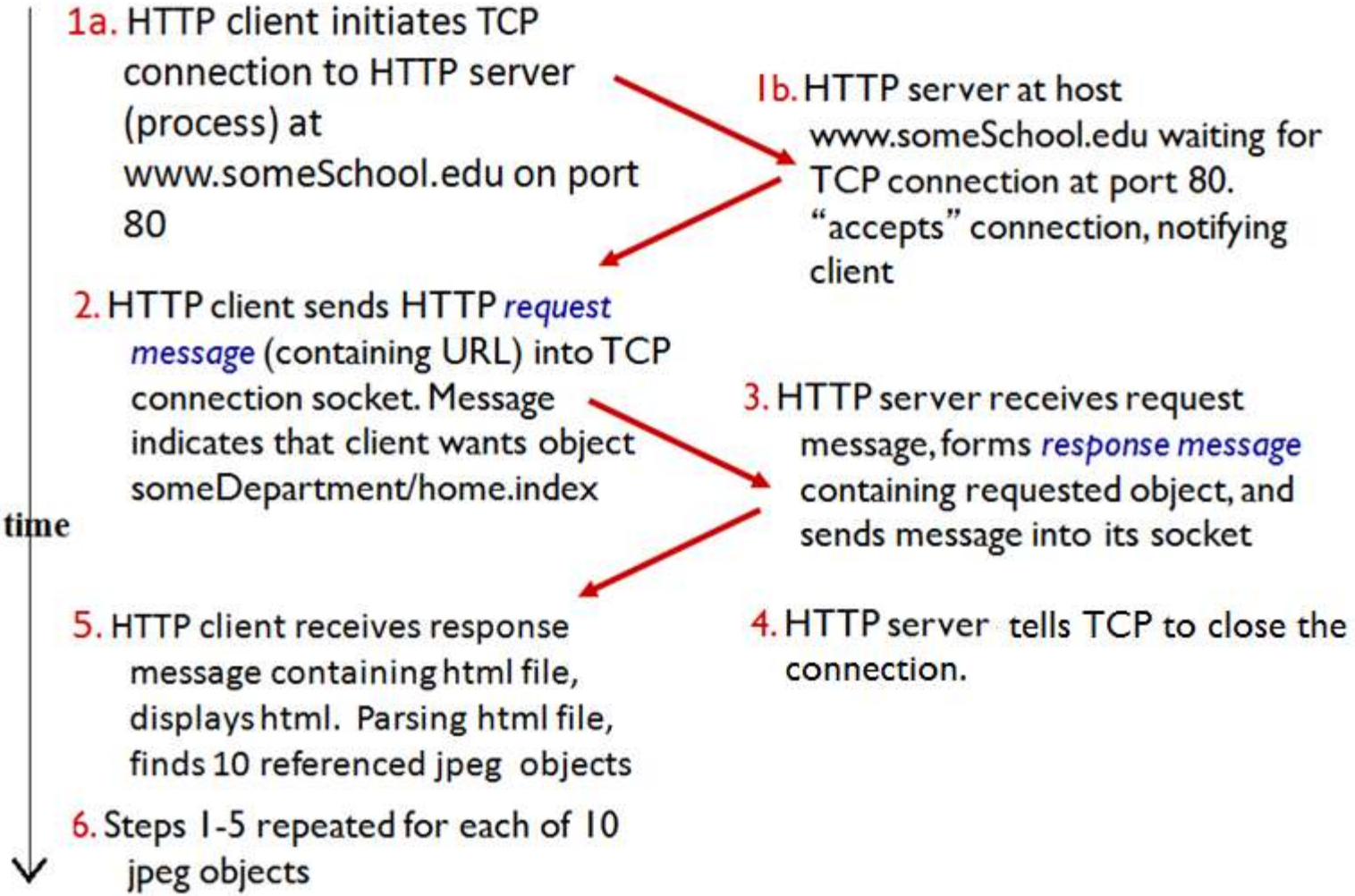
- Multiple objects can be sent over single TCP connection between client and server.

Non-persistent HTTP

- Suppose user enters URL:

  `www.someSchool.edu/someDepartment/home.index`

  – contains text, references to 10 jpeg images.



Non-persistent HTTP: Response Time

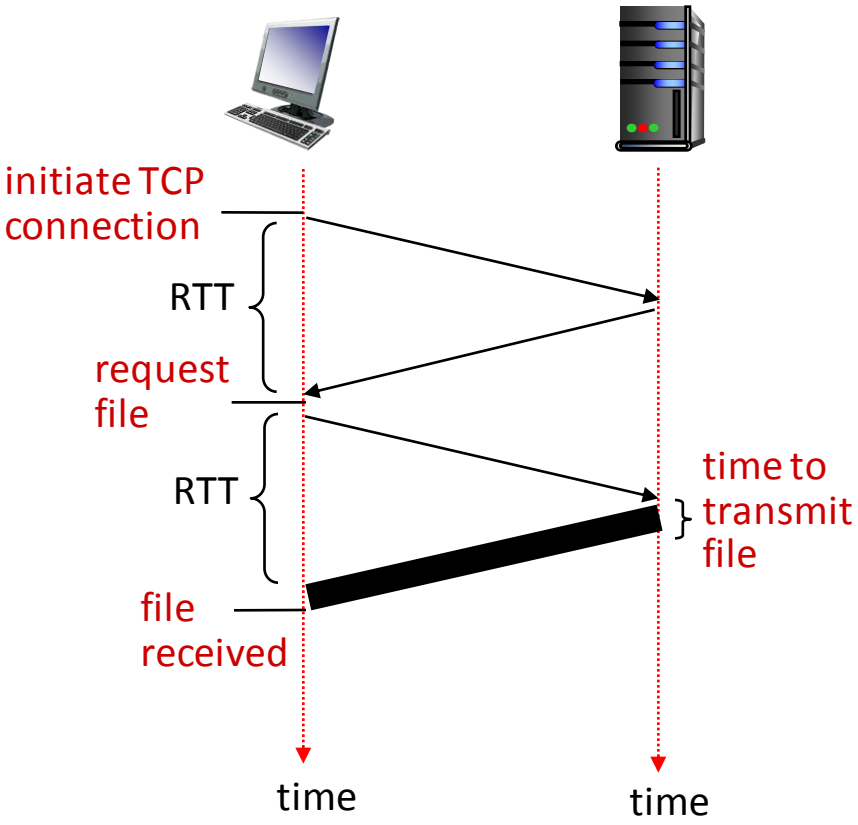
Round-Trip Time (RTT):

- Time for a small packet to travel from client to server and back.

HTTP 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
- Non-persistent HTTP response time =

2RTT+ file transmission time



Persistent HTTP

Non-persistent 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:

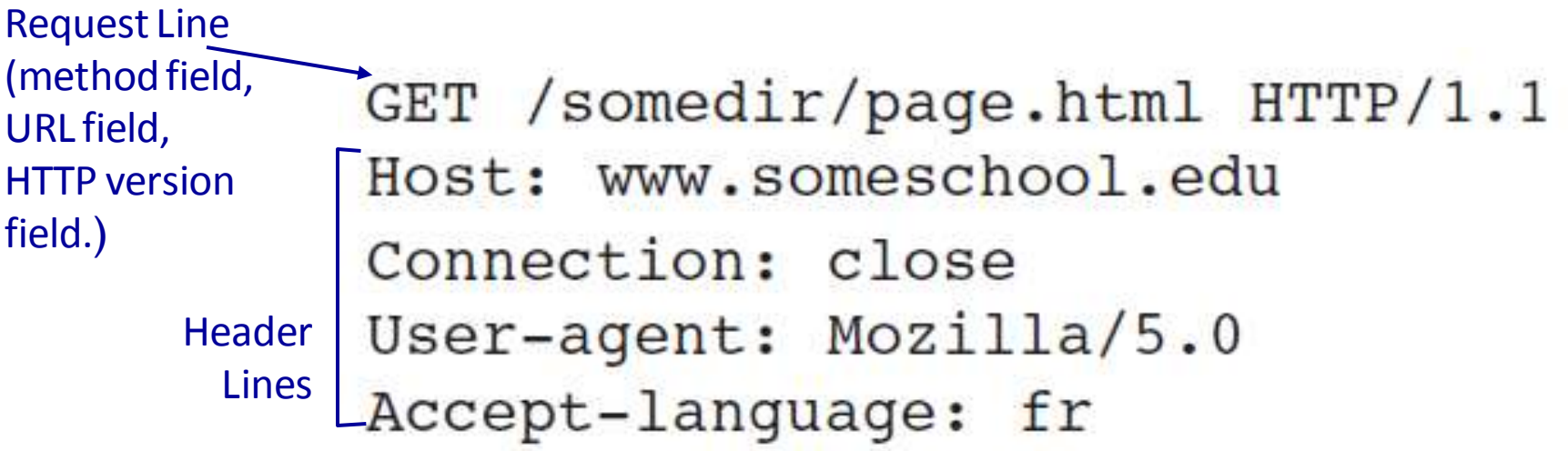
- Server leaves connection open after sending response
- Subsequent HTTP messages between same client/server sent over open connection
- Client sends requests as soon as it encounters a referenced object
- As little as one RTT for all the referenced objects

HTTP Messages

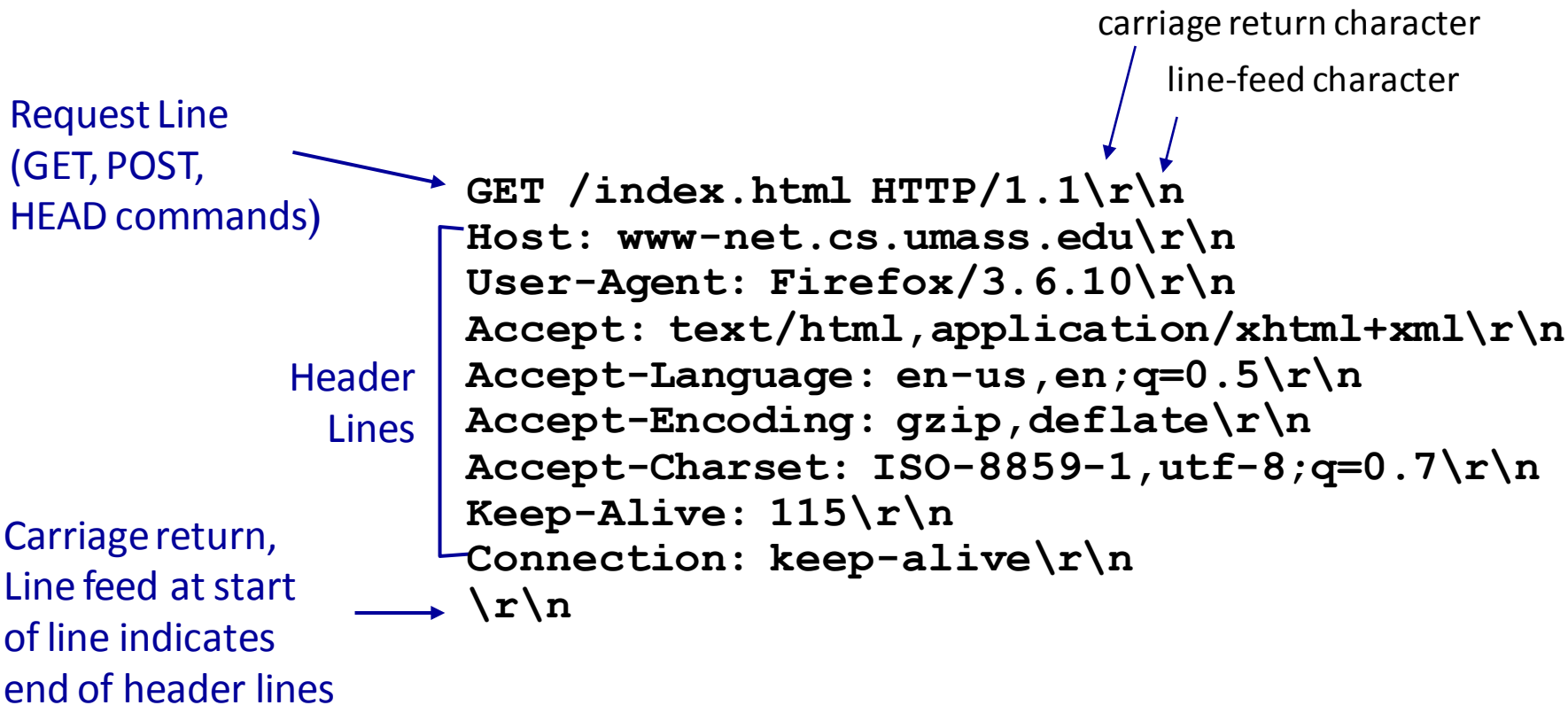
- Two types of HTTP messages:
  - *Request*
  - *Response*

HTTP Request Message

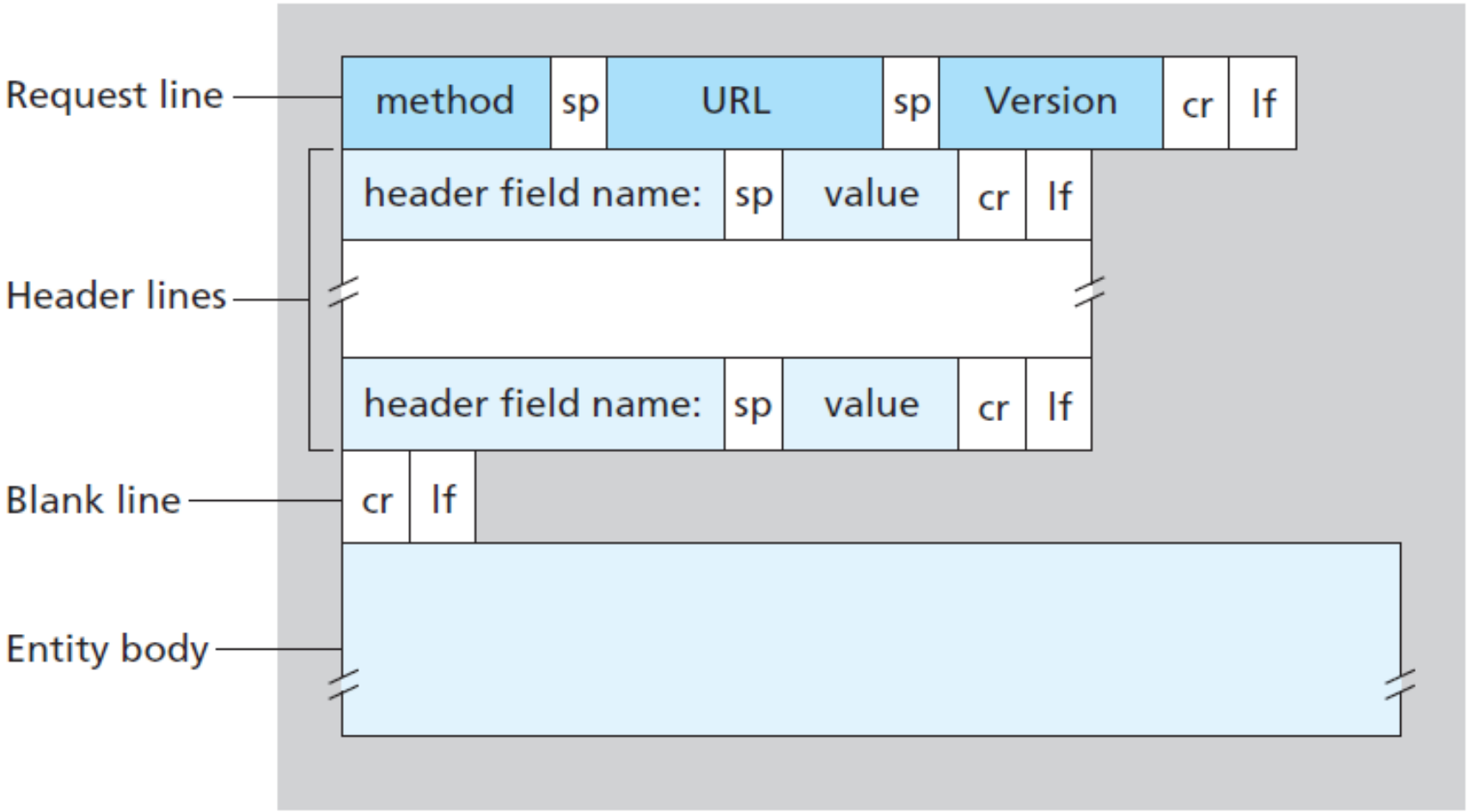
- HTTP request message:
  - ASCII (human-readable format)



HTTP Request Message…



HTTP Request Message: General Format



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
```

Method types

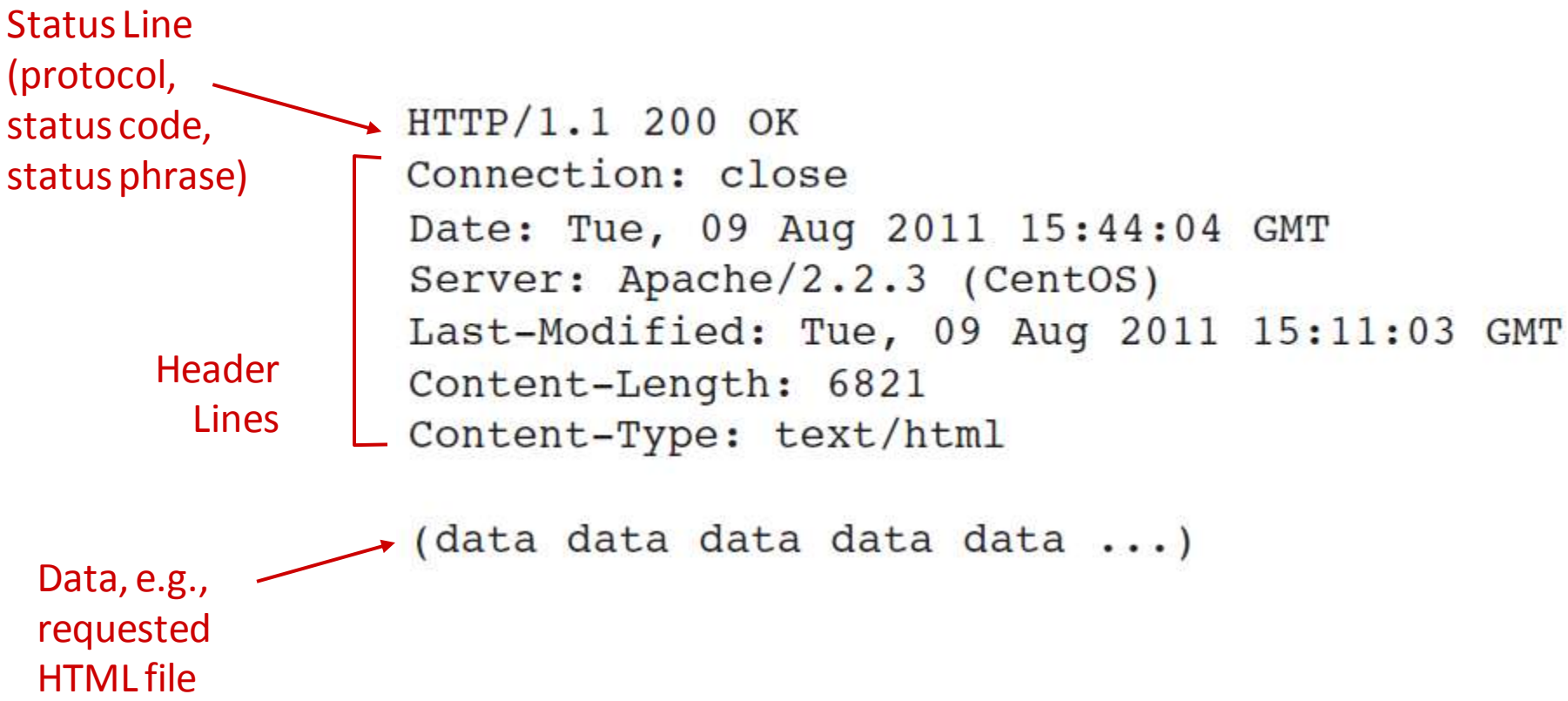
HTTP/1.0:

- GET
  - Requested object identified in the URL field.
- POST
  - Specific contents of the web page depend on what the user entered into the form fields and put it in 'entity body'.
- HEAD
  - Asks server to leave requested object out of response.
  - Used for debugging.

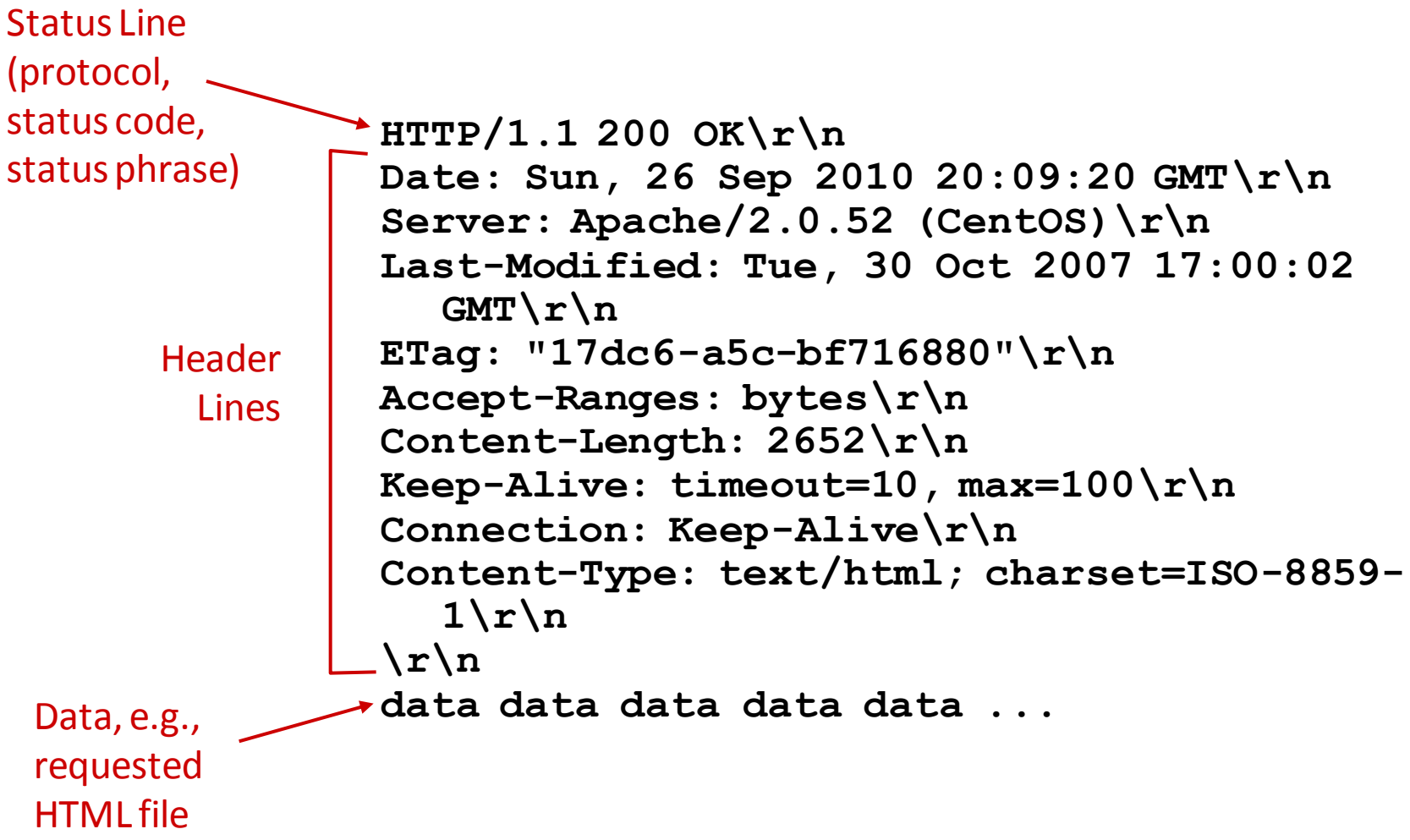
HTTP/1.1:

- GET, POST, HEAD
- PUT
  - Uploads file in entity body to path specified in URL field
- DELETE
  - Deletes file specified in the URL field

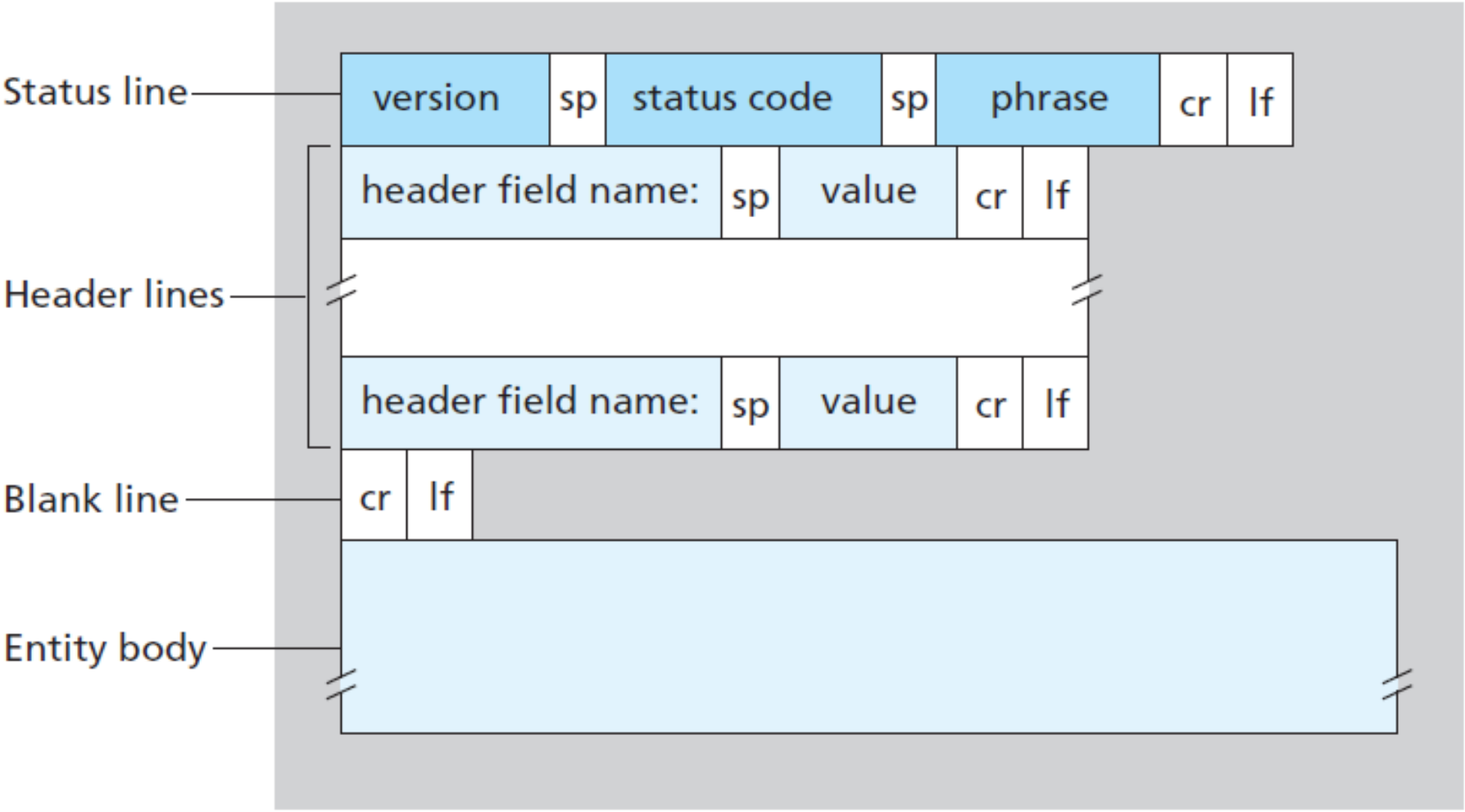
HTTP Response Message



HTTP Response Message…



HTTP Response Message: General Format



HTTP Response Status Codes

- Status code appears in 1st line in server-to-client response message.
- Some sample codes:

200 OK

- request succeeded, requested object later in this msg

301 Moved Permanently

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

400 Bad Request

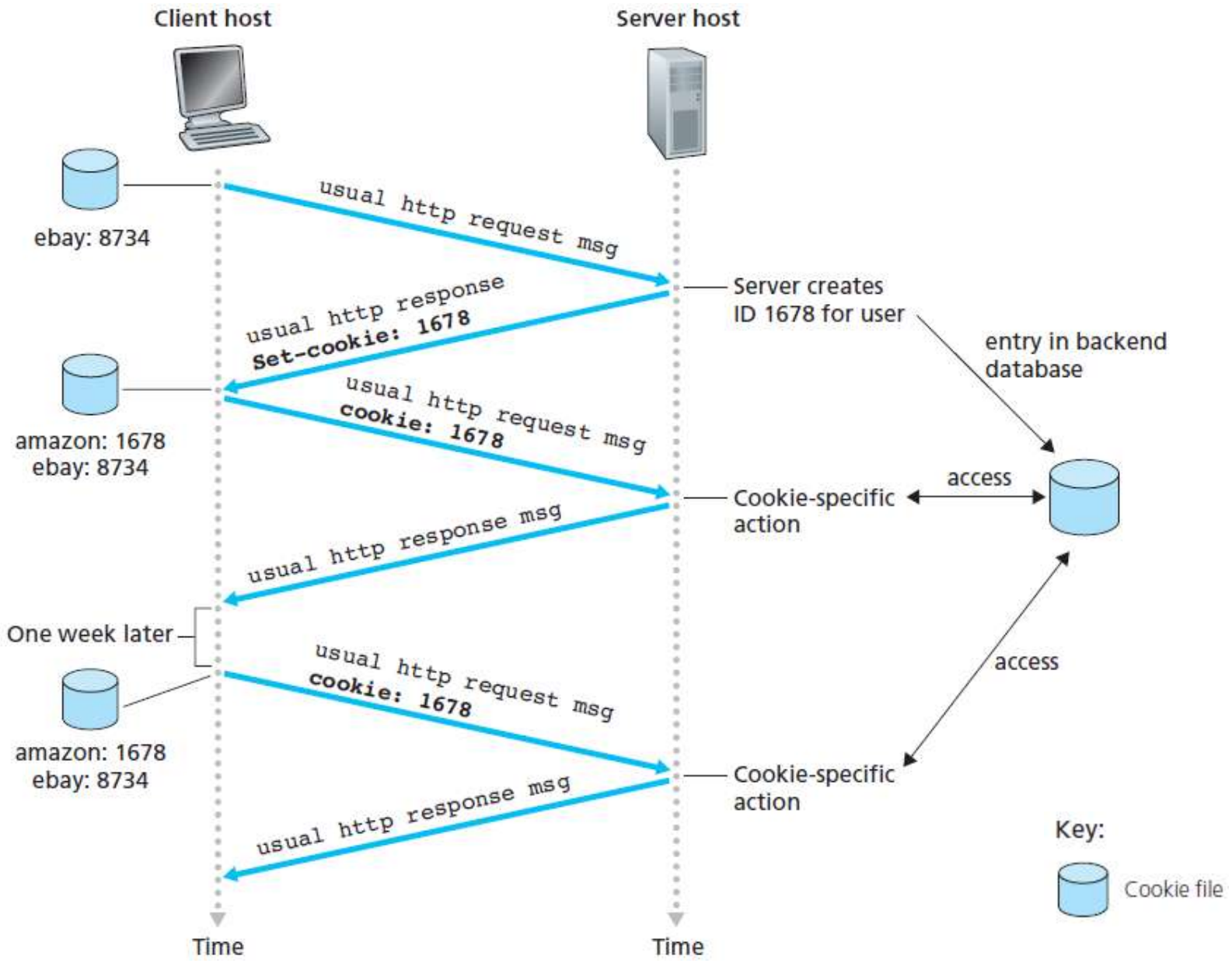
- request msg not understood by server

404 Not Found

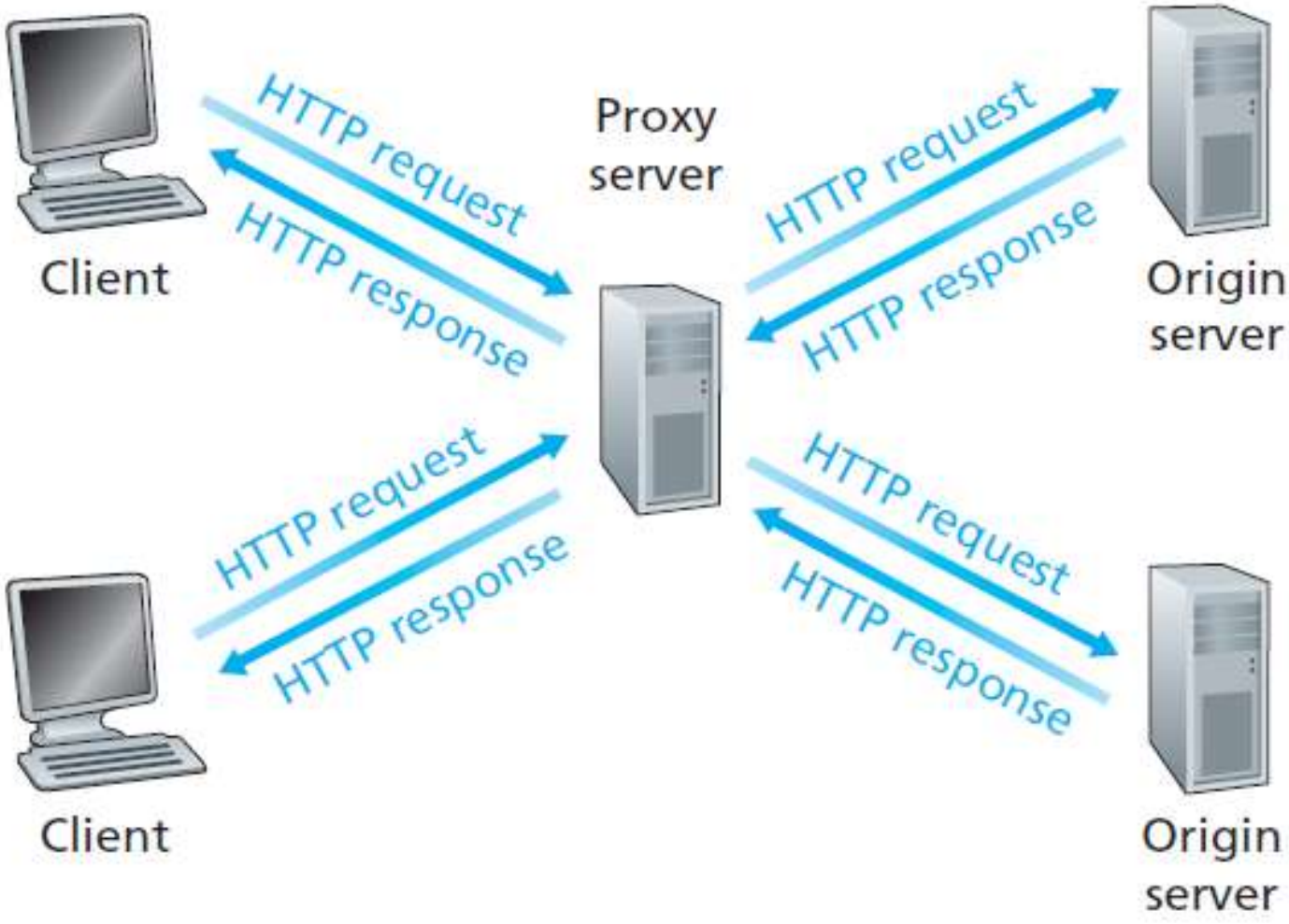
- requested document not found on this server

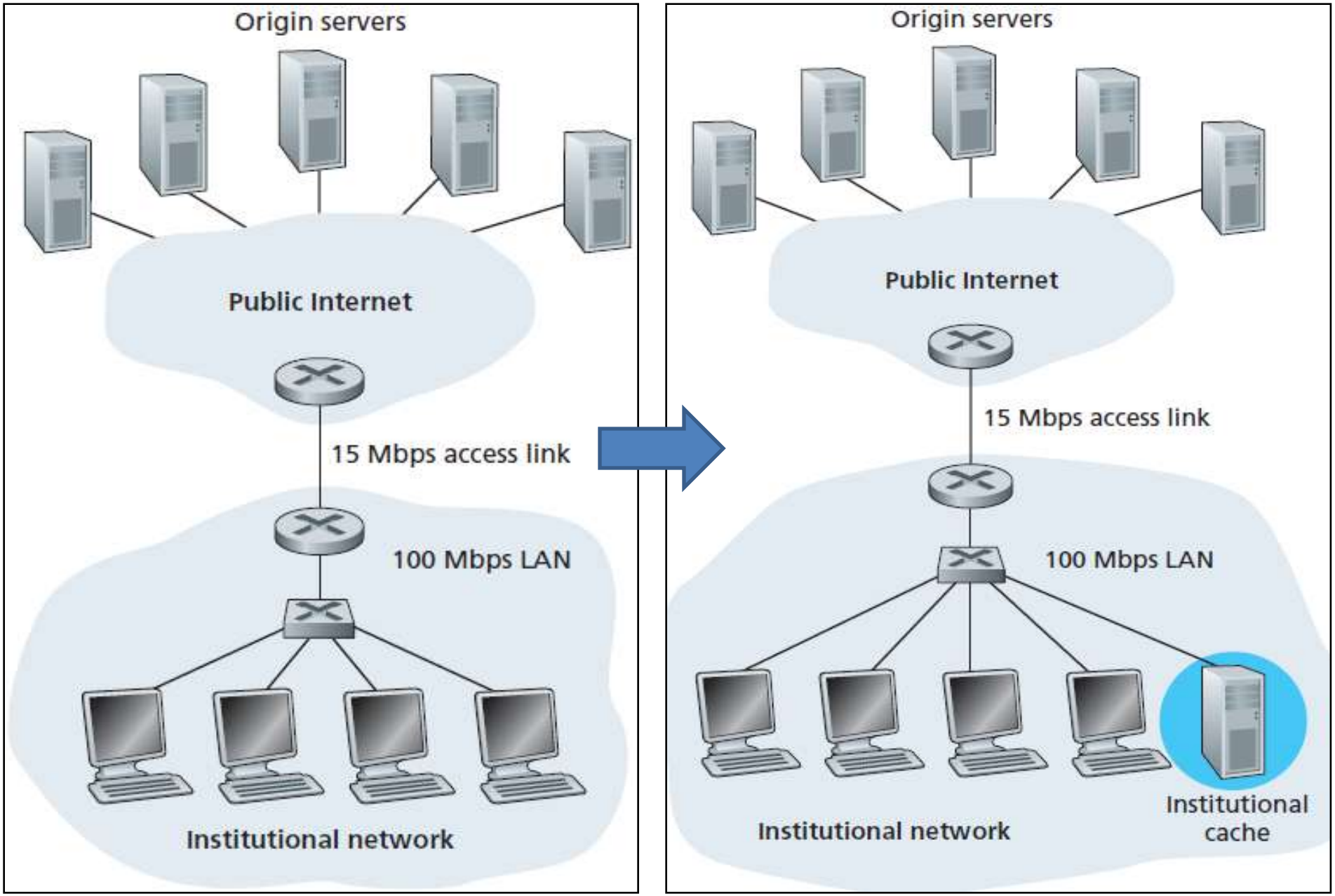
505 HTTP Version Not Supported

Keeping user state with Cookies



Web Caches (Proxy Server)





a) Bottleneck between an institutional network and the Internet

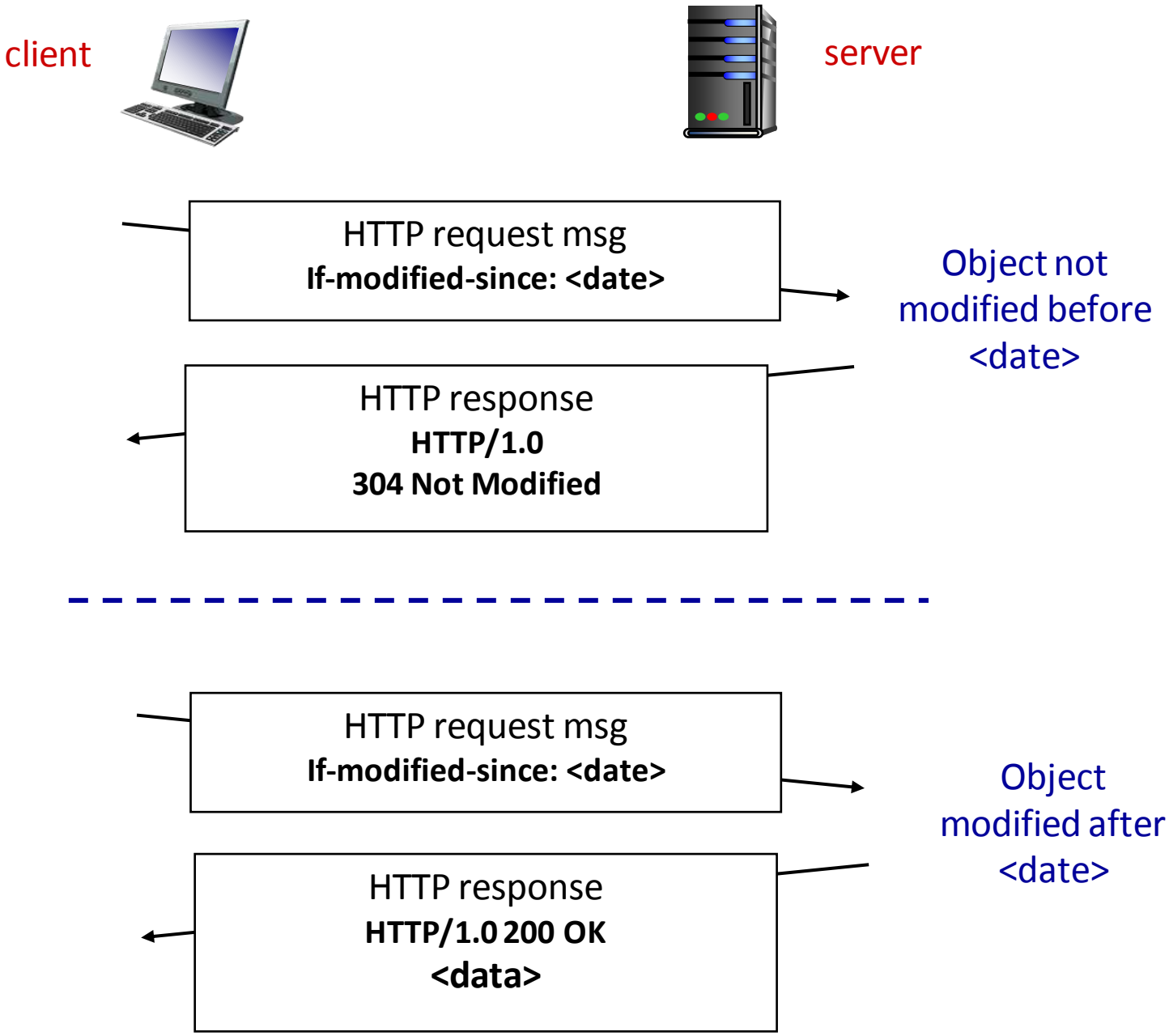
b) Adding a cache to the institutional network

Conditional GET

- *Goal:* don't send object if cache has up-to-date cached version.
  - No object transmission delay
  - Lower link utilization
- *Cache:* specify date of cached copy in HTTP request.

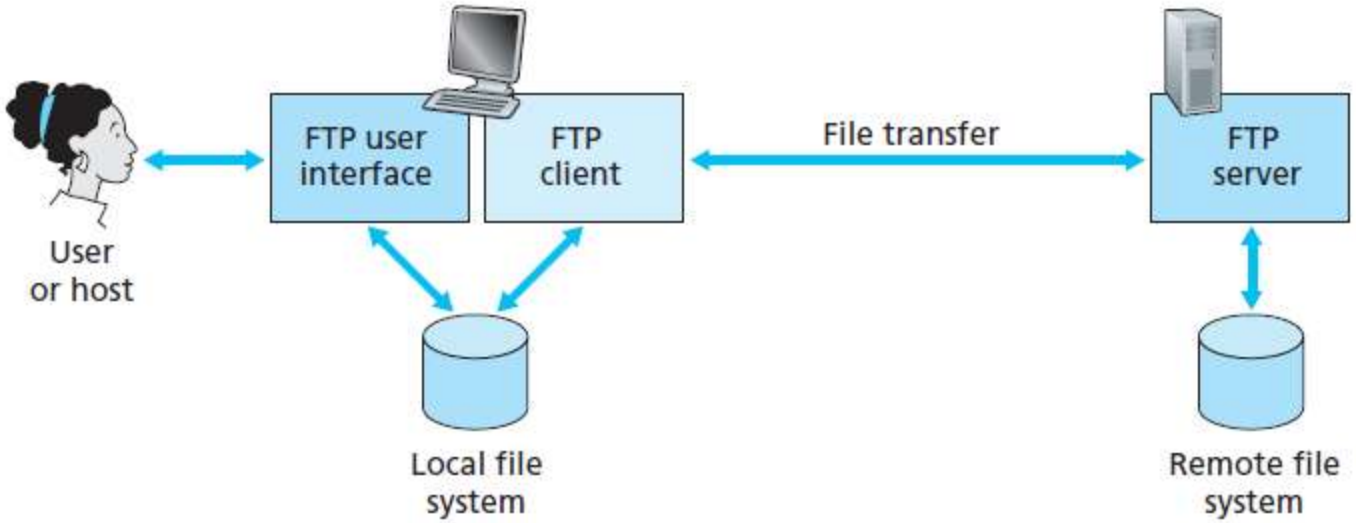
  `If-modified-since: <date>`
- *Server:* response contains no object if cached copy is up-to-date:

  `HTTP/1.0 304 not modified`


client
server
HTTP request msg
If-modified-since: <date>
Object not modified before <date>
HTTP response
HTTP/1.0
304 Not Modified
HTTP request msg
If-modified-since: <date>
Object modified after <date>
HTTP response
HTTP/1.0 200 OK
<data>

File Transfer Protocol (FTP)

- Transfer file to/from remote host.
- Client/server model
  - Client: initiates transfer (either to/from remote)
  - Server: remote host
- Ports: 20 and 21



Control and data connections in FTP

- FTP uses two parallel TCP connections to transfer a file.
- The **control connection** is used for sending control information between the two hosts.
  - information such as user identification, password, commands to change remote directory, and commands to "put" and "get" files.
- The **data connection** is used to actually send a file.

- FTP client contacts FTP server at port 21, using TCP.
- Client authorized over control connection.
- Client browses remote directory, sends commands over control connection.
- When server receives file transfer command, *server* opens *2nd* TCP data 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.

FTP Commands and Replies

Sample commands:

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

Sample replies:

- Three-digit numbers, with an optional message.
- **331 Username OK, password required**
- **125 data connection already open; transfer starting**
- **425 Can't open data connection**
- **452 Error writing file**

Electronic Mail in the Internet

- E-mail is an asynchronous communication medium.
  - People send and read messages when it is convenient for them.
  - Fast, easy to distribute, and inexpensive.
  - Has features, including messages with attachments, hyperlinks, HTML-formatted text, and embedded photos.
- Internet mail system has three major components:
  - **User agents,**
  - **Mail servers,** and
  - **Simple Mail Transfer Protocol (SMTP)**

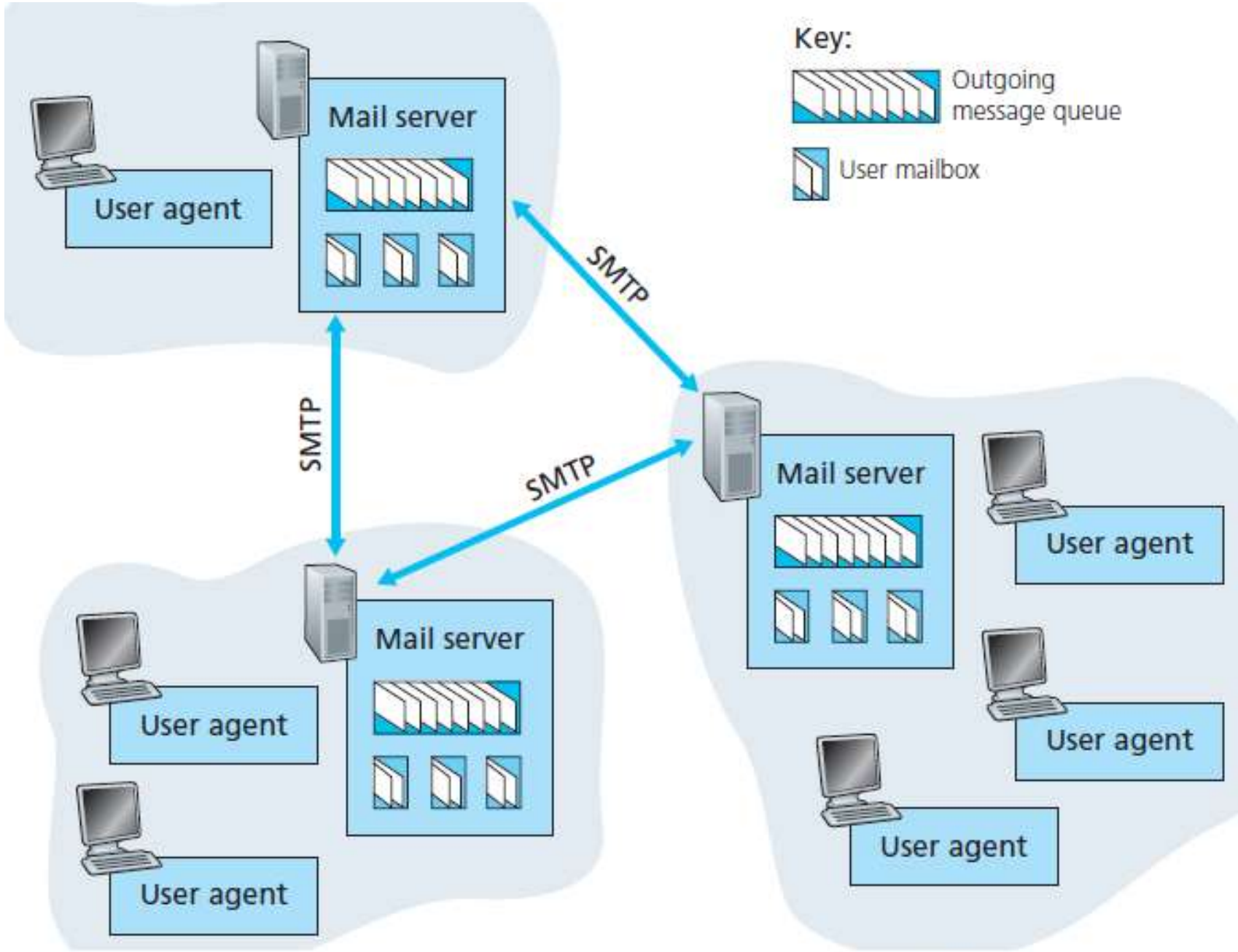


Figure ♦ A high-level view of the Internet e-mail system

Electronic mail…

User Agent

- "Mail reader"
- Composing, editing, reading mail messages.
- E.g., Outlook, thunderbird, iphone mail client
- Outgoing, incoming messages stored on server.

Mail Servers:

- *Mailbox* contains incoming messages for user.
- *Message queue* of outgoing (to be sent) mail messages.

SMTP protocol

- *Protocol* between mail servers to send email messages.
  - client: sending mail server
  - "server": receiving mail server

Simple Mail Transfer Protocol (SMTP)

- RFC 5321
- 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 (like HTTP, FTP)
  - Commands: ASCII text
  - Response: status code and phrase

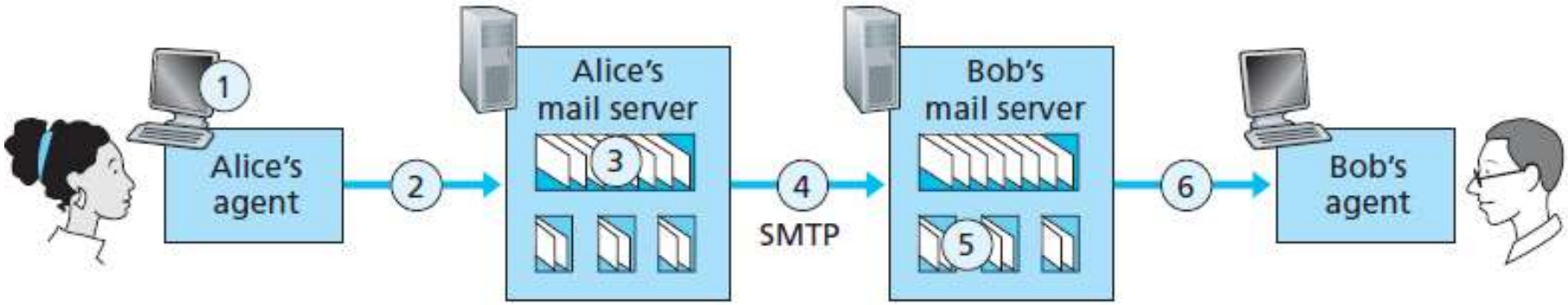
SMTP Commands

Keyword	*Argument(s)*	*Description*
HELO	Sender's host name	Identifies itself
MAIL FROM	Sender of the message	Identifies the sender of the message
RCPT TO	Intended recipient	Identifies the recipient of the message
DATA	Body of the mail	Sends the actual message
QUIT		Terminates the message
RSET		Aborts the current mail transaction
VRFY	Name of recipient	Verifies the address of the recipient
NOOP		Checks the status of the recipient
TURN		Switches the sender and the recipient
EXPN	Mailing list	Asks the recipient to expand the mailing list.
HELP	Command name	Asks the recipient to send information about the command sent as the argument
SEND FROM	Intended recipient	Specifies that the mail be delivered only to the terminal of the recipient, and not to the mailbox
SMOL FROM	Intended recipient	Specifies that the mail be delivered to the terminal *or* the mailbox of the recipient
SMAL FROM	Intended recipient	Specifies that the mail be delivered to the terminal *and* the mailbox of the recipient

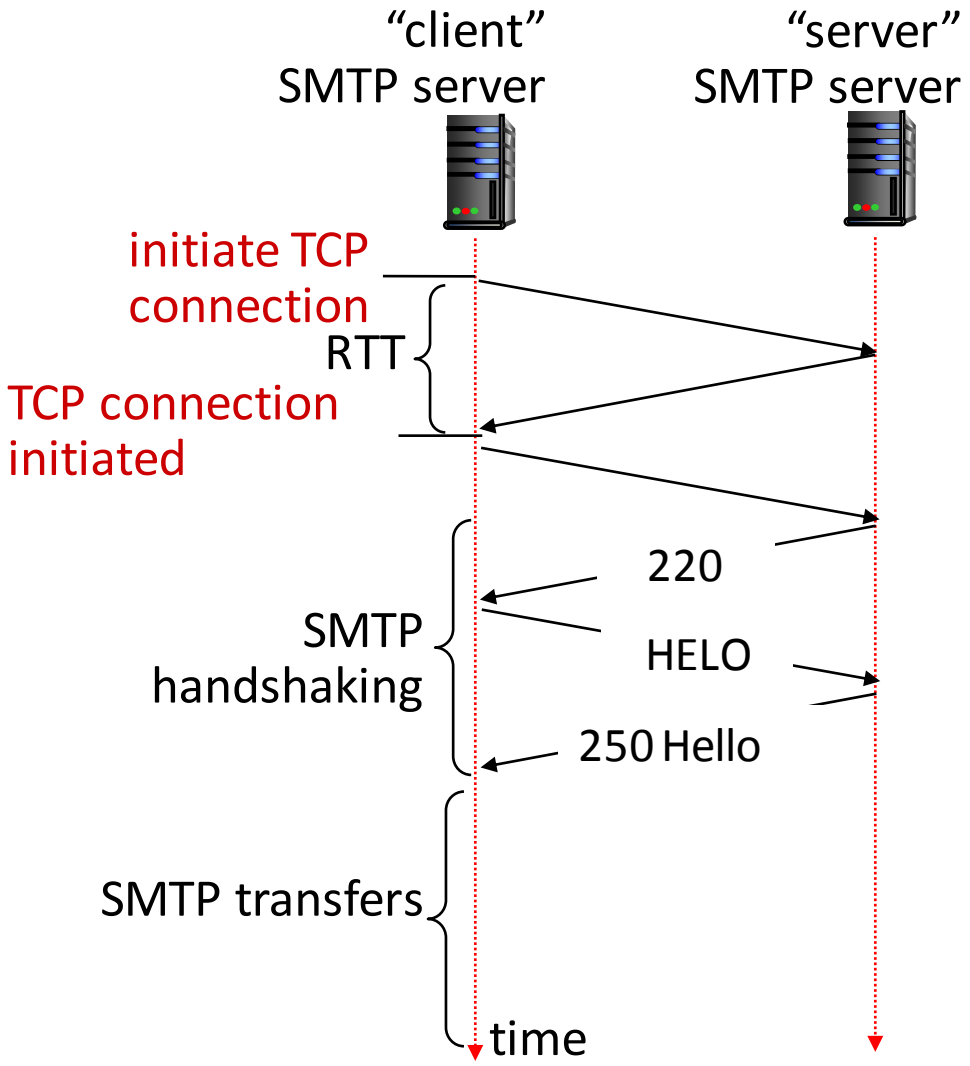
Responses

Code	*Description*
Positive Completion Reply	
211	System status or help reply
214	Help message
220	Service ready
221	Service closing transmission channel
250	Request command completed
251	User not local; the message will be forwarded
Positive Intermediate Reply	
354	Start mail input
Transient Negative Completion Reply	
421	Service not available
450	Mailbox not available
451	Command aborted: local error
452	Command aborted; insufficient storage
Permanent Negative Completion Reply	
500	Syntax error; unrecognized command
501	Syntax error in parameters or arguments
502	Command not implemented
503	Bad sequence of commands
504	Command temporarily not implemented
550	Command is not executed; mailbox unavailable
551	User not local
552	Requested action aborted; exceeded storage location
553	Requested action not taken; mailbox name not allowed
554	Transaction failed

Scenario



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



- 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.

SMTP : Comparison with HTTP

HTTP

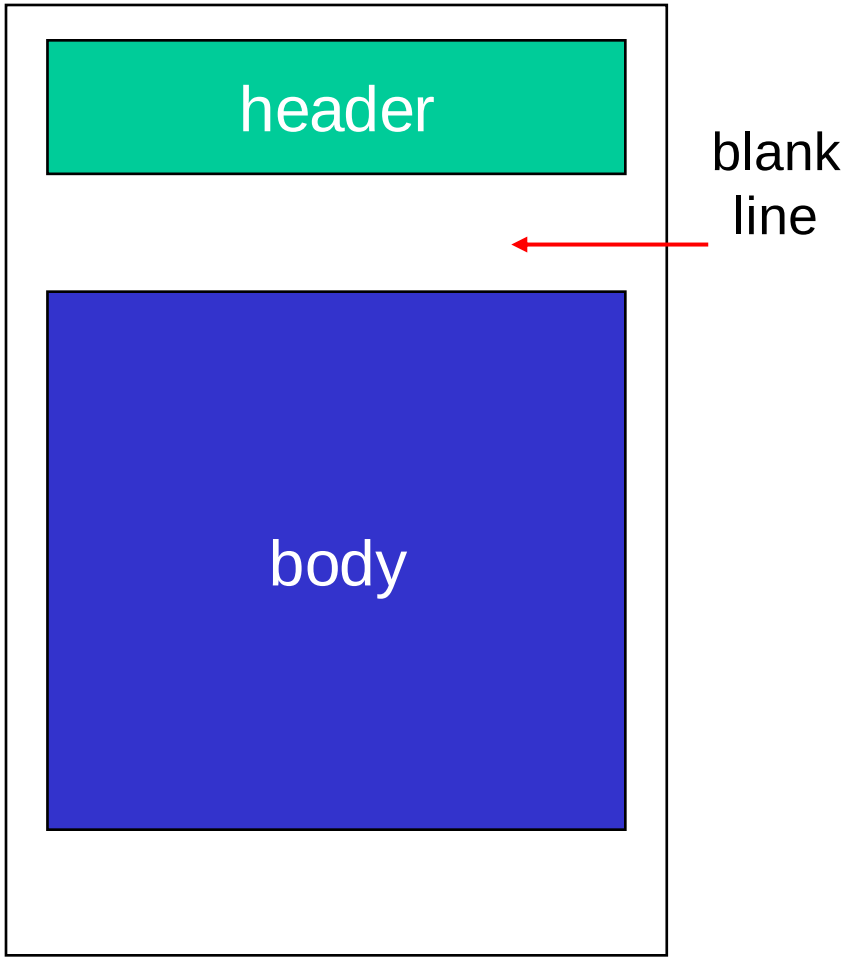
- HTTP: pull
- Have ASCII command/response interaction, status codes.
- Each object encapsulated in its own response msg.

SMTP

- SMTP: push
- Have ASCII command/response interaction, status codes
- Multiple objects sent in multipart msg.

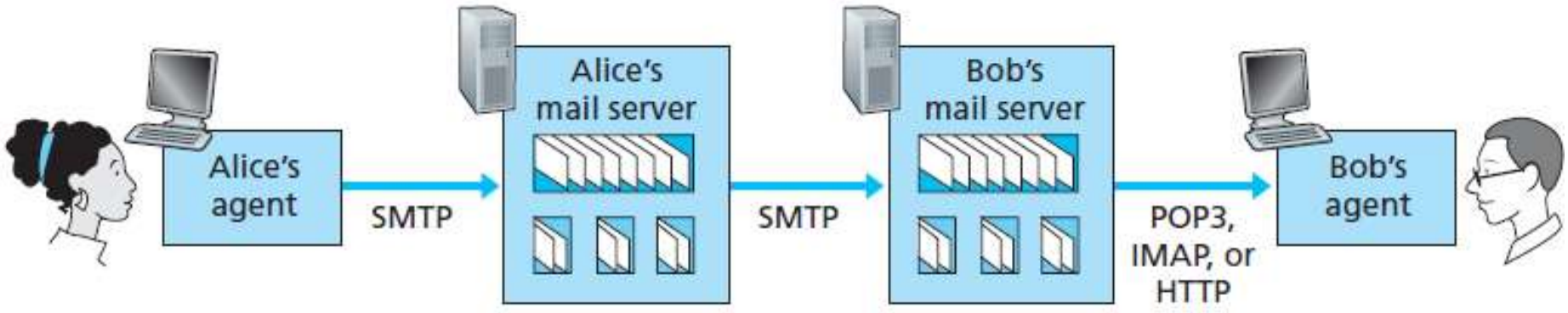
Mail Message Formats

- RFC 5322: standard for text message format:
- Header lines, e.g.,
  - To:
  - From:
  - Subject:
- Body: the "message"
  - ASCII characters only



Mail Access Protocols

- SMTP: delivery/storage to receiver's server
- Mail access protocol: retrieval from server
  - POP: **Post Office Protocol** [RFC 1939]: authorization, download
  - IMAP: **Internet Mail Access Protocol** [RFC 3501]: more features, including manipulation of stored msgs on server
  - HTTP: Gmail, Hotmail, Yahoo! Mail, etc.



POP3 Protocol

- **Post Office Protocol—Version 3**
- Simple mail access protocol.
- POP3 begins when the user agent (the client) opens a TCP connection to the mail server (the server) on port 110.
- With the TCP connection established, POP3 progresses through three phases:
  - Authorization,
  - Transaction,
  - Update.

Authorization Phase:

- The user agent sends a username and a password to authenticate the user.
- Client commands:
  - **`user`**: declare username
  - **`pass`**: password
- Server responses
  - **`+OK`**
  - **`-ERR`**

Transaction Phase

- The user agent retrieves messages.
- The user agent can mark messages for deletion, remove deletion marks, and obtain mail statistics.
- Client commands:
  - **`list`**: list message numbers
  - **`retr`**: retrieve message by number
  - **`dele`**: delete
  - **`quit`**

Update phase

- Occurs after the client has issued the `quit` command, ending the POP3 session.
- At this time, the mail server deletes the messages that were marked for deletion.

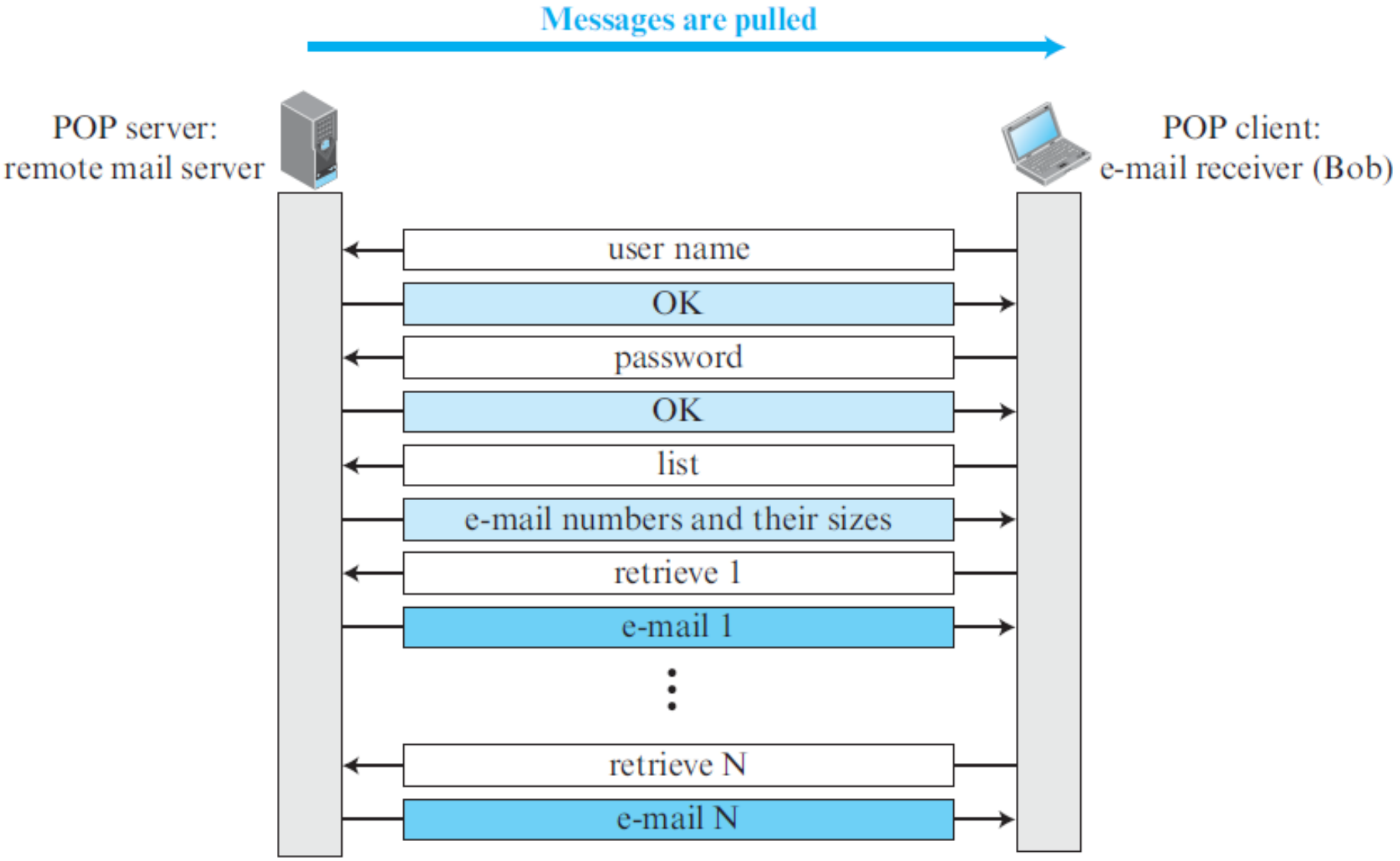

Messages are pulled
POP server:
remote mail server
POP client:
e-mail receiver (Bob)
user name
OK
password
OK
list
e-mail numbers and their sizes
retrieve 1
e-mail 1
retrieve N
e-mail N

Figure: POP3

POP3 and IMAP

- POP3 has two modes:
  - **Delete mode**
  - **Keep mode.**
- In the delete mode, the mail is deleted from the mailbox after each retrieval.
- In the keep mode, the mail remains in the mailbox after retrieval.
- POP3 is stateless across sessions

IMAP

- Keeps all messages in one place: at server
- Allows user to organize messages in folders
- Keeps user state across sessions:
- Names of folders and mappings between message ids and folder name

DNS: Domain Name System

Introduction

- Internet hosts can be identified by
  - IP address (32 bit)
    - E.g., 121.7.106.83
  - Hostname
    - Human readable
    - E.g., www.yahoo.com, www.ktu.edu.in, www.eurecom.fr

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

DNS: Domain Name System

- The main task of **DNS** is translates hostnames to IP addresses.
- *DNS is*
  - *A distributed database* implemented in hierarchy of many *name servers.*
  - *An application-layer protocol* that allows hosts to query the distributed database.
- The **DNS servers** are UNIX machines running the Berkeley Internet Name Domain (BIND) software.
- The DNS protocol runs over UDP.
- Uses port 53.

DNS Working

- Consider what happens when a browser requests the URL

  `www.someschool.edu/index.html`

  1. The user machine runs the client side of the DNS application.
  2. The browser extracts the hostname, **`www.someschool.edu`**, from the URL and passes the hostname to the client side of the DNS application.
  3. The DNS client sends a query containing the hostname to a DNS server in port 53.
  4. The DNS client receives a reply, which includes the IP address for the hostname.
  5. Once the browser receives the IP address from DNS, it can initiate a TCP connection to the HTTP server process located at port 80 at that IP address.

Services Provided by DNS

1. Translating hostnames to IP addresses.
2. Host aliasing.
   - DNS can be invoked by an application to obtain the canonical hostname for a supplied alias hostname as well as the IP address of the host.
3. Mail server aliasing.
   - DNS can be invoked by a mail application to obtain the canonical hostname.
4. Load distribution.
   - For replicated Web servers, a *set of* IP addresses is associated with one canonical hostname.
   - DNS rotation distributes the traffic among the replicated servers.

DNS is distributed!

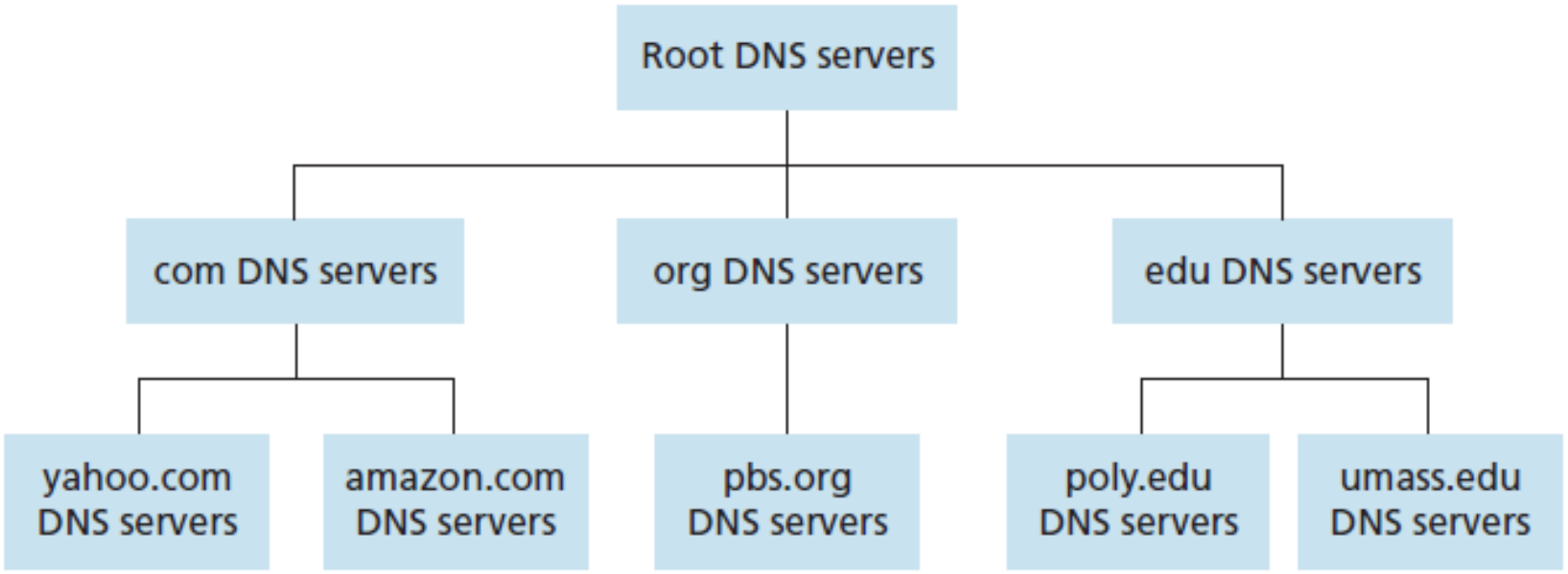
Why not centralize DNS?

- The problems with a centralized design include:
  - Single point of failure
  - Traffic volume
  - Distant centralized database
  - Maintenance

Answer : doesn't scale!

DNS: A Distributed, Hierarchical Database

- There are three classes of DNS servers:
  - Root DNS servers,
  - Top-Level Domain (TLD) DNS servers, and
  - Authoritative DNS servers.



Client wants IP for www.amazon.com:

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

DNS: Root Name Servers

- There are 13 root DNS servers (labeled A through M), most of which are located in North America.
  - Each "server" is actually a network of replicated servers, for both security and reliability purposes.
  - Refer → https://root-servers.org/

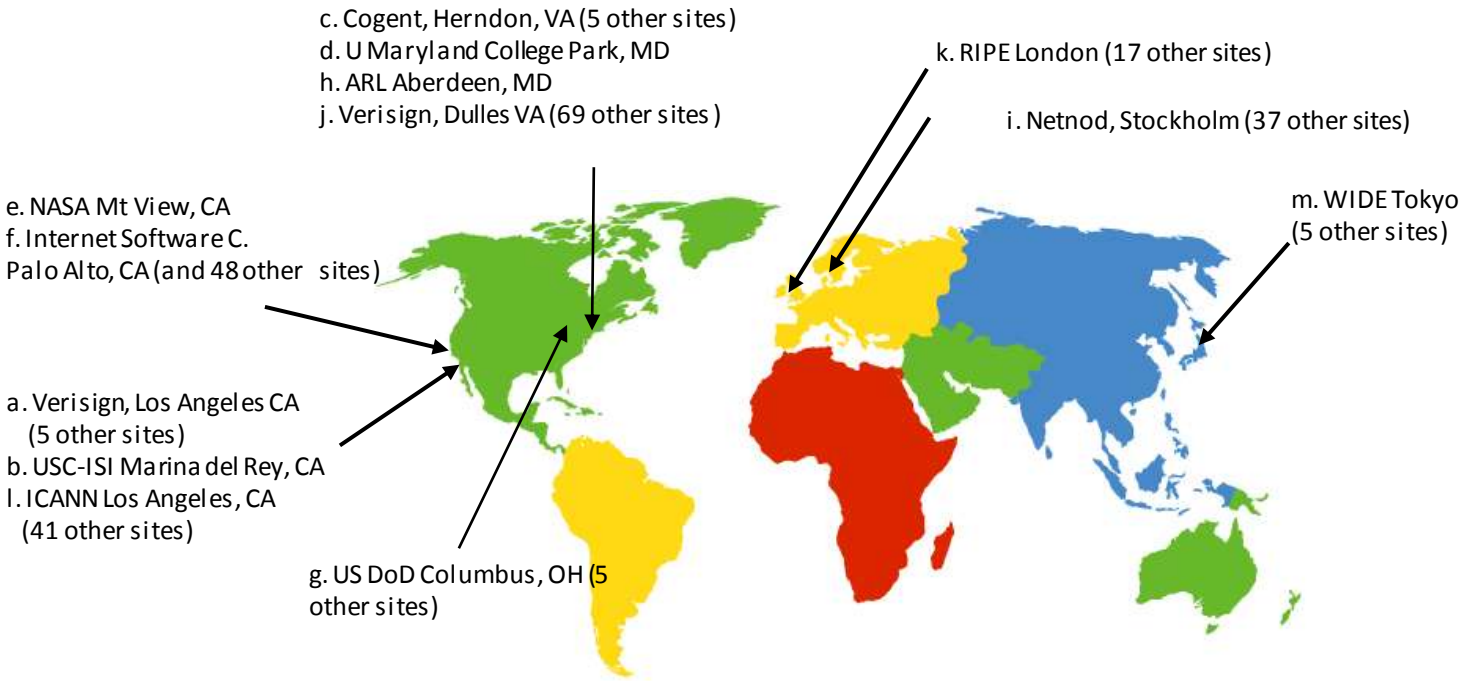


Fig: DNS root servers in 2012

TLD and Authoritative Servers

Top-level domain (TLD) servers:

- Responsible for **`com, org, net, edu, aero, jobs, museums`**, and all top-level country domains, e.g., **`uk, fr, ca, jp.`**
- The company Verisign Global Registry Services maintains the TLD servers for the **com** top-level domain.
- The company Educause maintains the TLD servers for the **edu** top-level domain.
- Refer → https://www.iana.org/domains/root/db

Authoritative DNS servers:

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

Local DNS name server

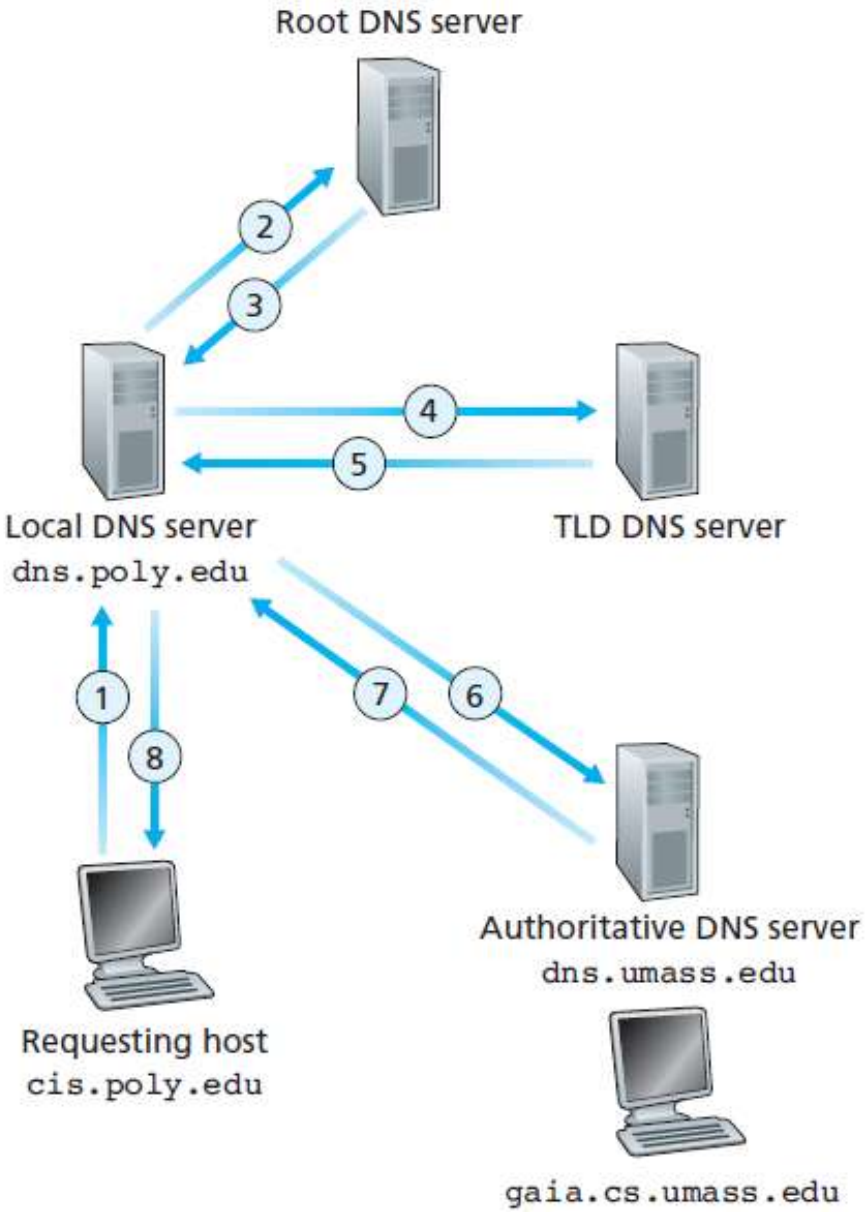
- Does not strictly belong to hierarchy.
- Each ISP (residential ISP, company, university) has one.
  - Also called "default name server"
- When host makes DNS query, query is sent to its local DNS server.
  - Has local cache of recent name-to-address translation pairs (but may be out of date!).
  - Acts as proxy, forwards query into hierarchy.

DNS Name Resolution Example

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

Iterated query:

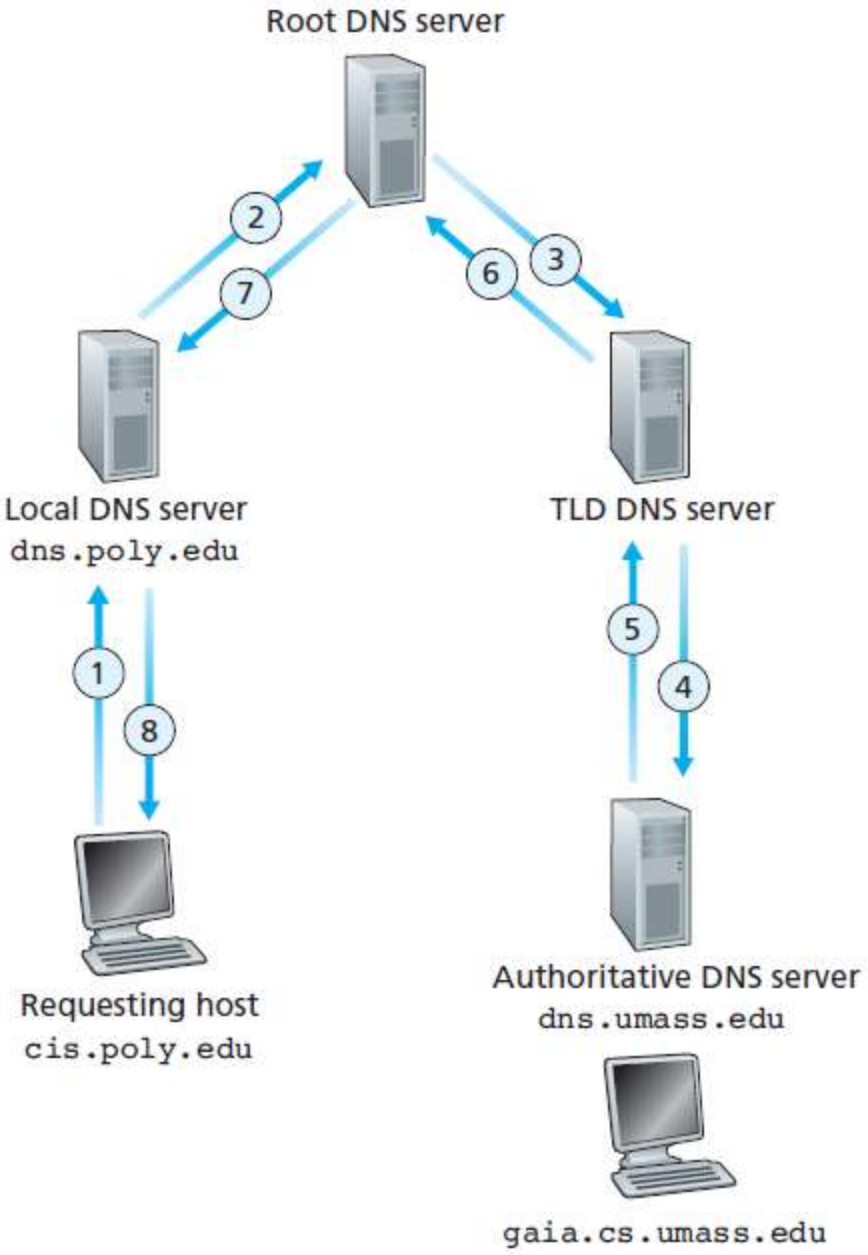
- Contacted server replies with name of server to contact.



DNS Name Resolution Example

Recursive query:

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



DNS Caching

- Once (any) name server learns mapping, it *caches* mapping.
  - Cache entries timeout (disappear) after some time (TTL).
    - Often set to two days.
  - TLD servers typically cached in local name servers.
    - Thus root name servers not often visited.
- Cached entries may be *out-of-date*.
  - If name host changes IP address, may not be known Internet-wide until all TTLs expire

DNS Records

- The DNS distributed database store **resource records (RRs)** that provide hostname-to-IP address mappings.
- Each DNS reply message carries one or more resource records.
- A resource record is a four-tuple that contains the following fields:

(Name, Value, Type, TTL)

DNS records…

- TTL is the time to live of the resource record.
  - It determines when a resource should be removed from a cache.
- The meaning of Name and Value depend on Type:

type=A

- **Name** is hostname
- **Value** is IP address

type=NS

- **Name** is domain (e.g., Foo.Com)
- **Value** is hostname of authoritative name server for this domain

type=CNAME

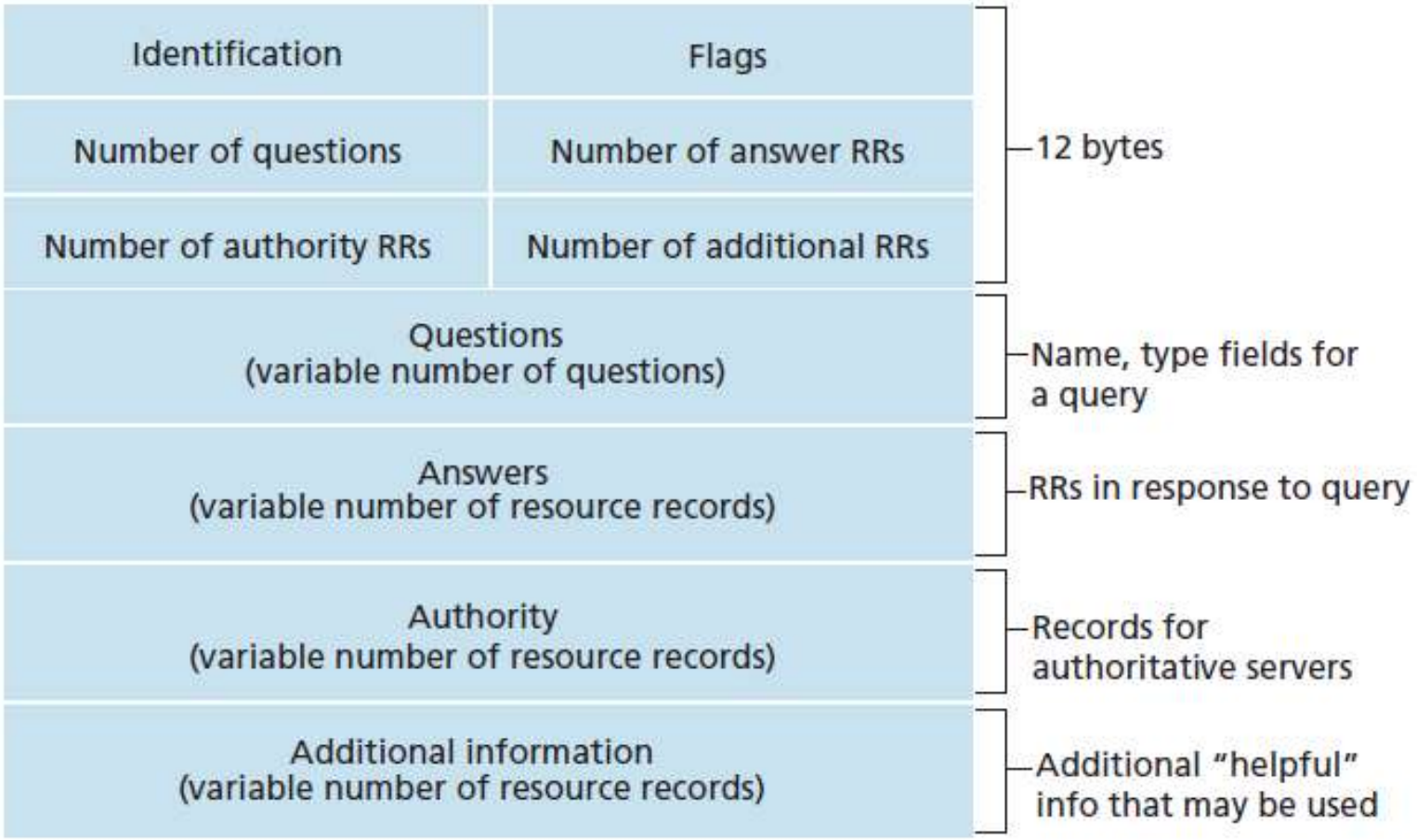
- **Name** is alias name for some "canonical" (the real) name
  - www.ibm.com is really servereast.backup2.ibm.com
- **Value** is canonical name.

type=MX

- **Value** is name of mailserver associated with **Name**

DNS Messages

- These are the only two kinds of DNS messages.
- Both query and reply messages have the same format.



Message header

- Identification:
  - 16 bit number for query,
  - reply to query uses same number.
- Flags:
  - 1-bit query/reply flag [query (0) , reply (1)]
  - 1-bit authoritative flag (in a reply message)
  - 1-bit recursion-desired flag
  - 1-bit recursion available flag (in a reply message)
- Number of occurrences of the four types of data sections.

Inserting records into DNS

- Example: new startup "Network Utopia"
- Register name networkuptopia.com at *DNS registrar* (e.g., Network Solutions)
  - Provide names, IP addresses of authoritative name server (primary and secondary)
  - Registrar inserts two RRs into .com TLD server:
    (networkutopia.com, dns1.networkutopia.com, NS)
    (dns1.networkutopia.com, 212.212.212.1, A)
- Create authoritative server type A record for www.networkuptopia.com; type MX record for networkutopia.com

DNS registrar

- A **registrar** is a commercial entity that
  - verifies the uniqueness of the domain name,
  - enters the domain name into the DNS database
  - collects a small fee for its services.
- Prior to 1999, there is a single registrar, Network Solutions.
- Internet Corporation for Assigned Names and Numbers (ICANN) accredits the various registrars.
- http://www.internic.net.

DNS: *Update* option

- More recently, an UPDATE option has been added to the DNS protocol
  - allow data to be dynamically added or deleted from the database via DNS messages.
  - [RFC 2136] and [RFC 3007] specify DNS dynamic updates.