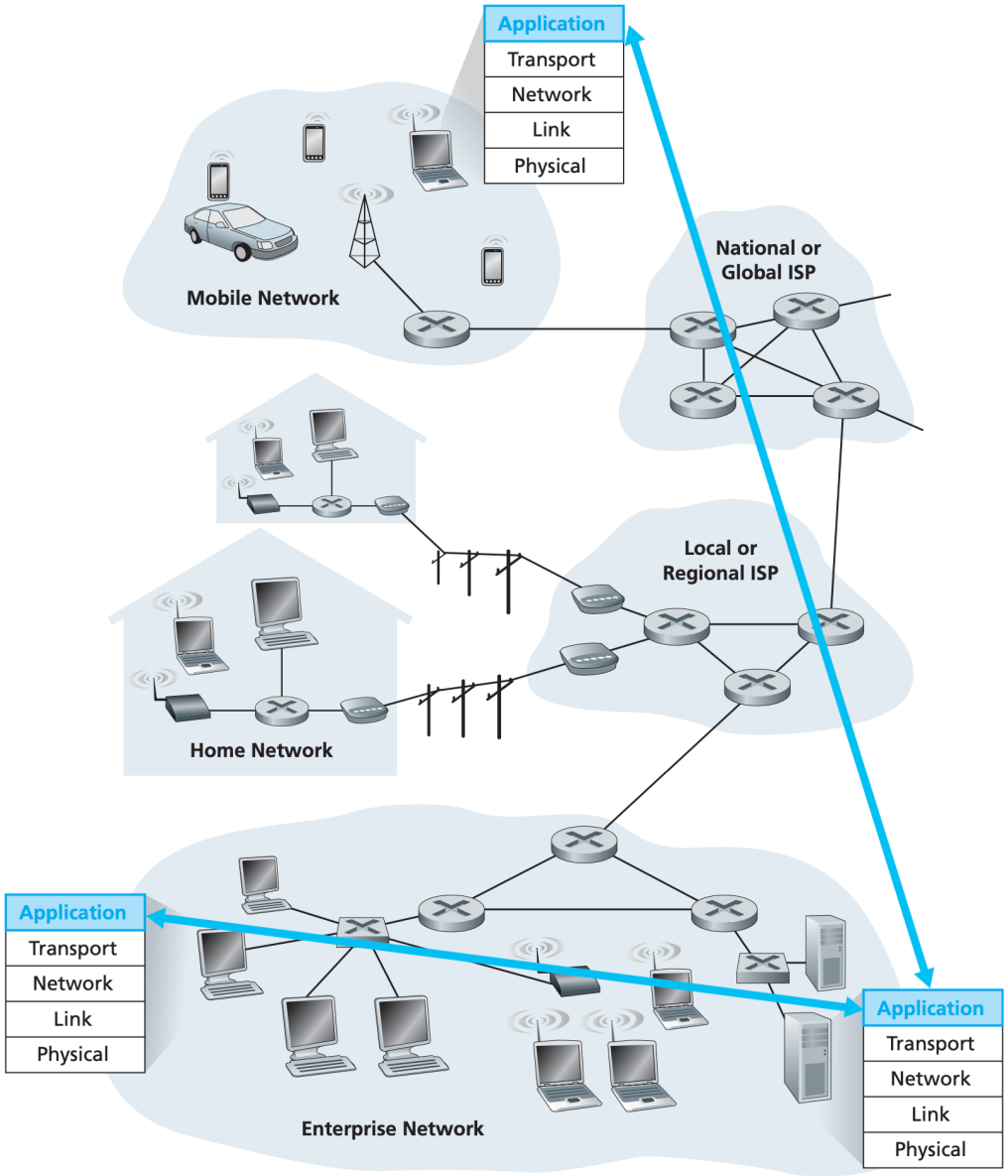
Application Layer

Hailiang Zhao @ ZJU.CS.CCNT
http://hliangzhao.me

This slide is a reminder for chapter 2 of the book *Computer Networking: A Top-Down Approach.*

Principles of Network Applications



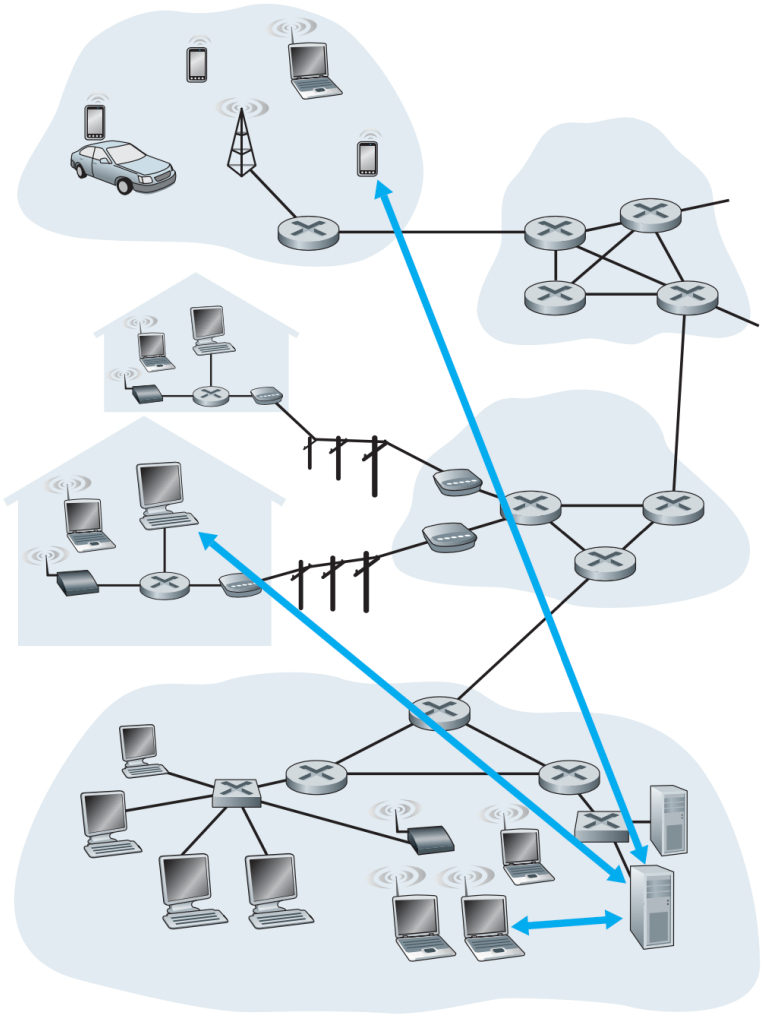
- Network application development is writing programs that *run on different end systems* and communicate with each other over the network
- We do not need to write software that runs on network-core devices, such as routers or link-layer switches

Communication for a network application takes place between end systems at the application layer

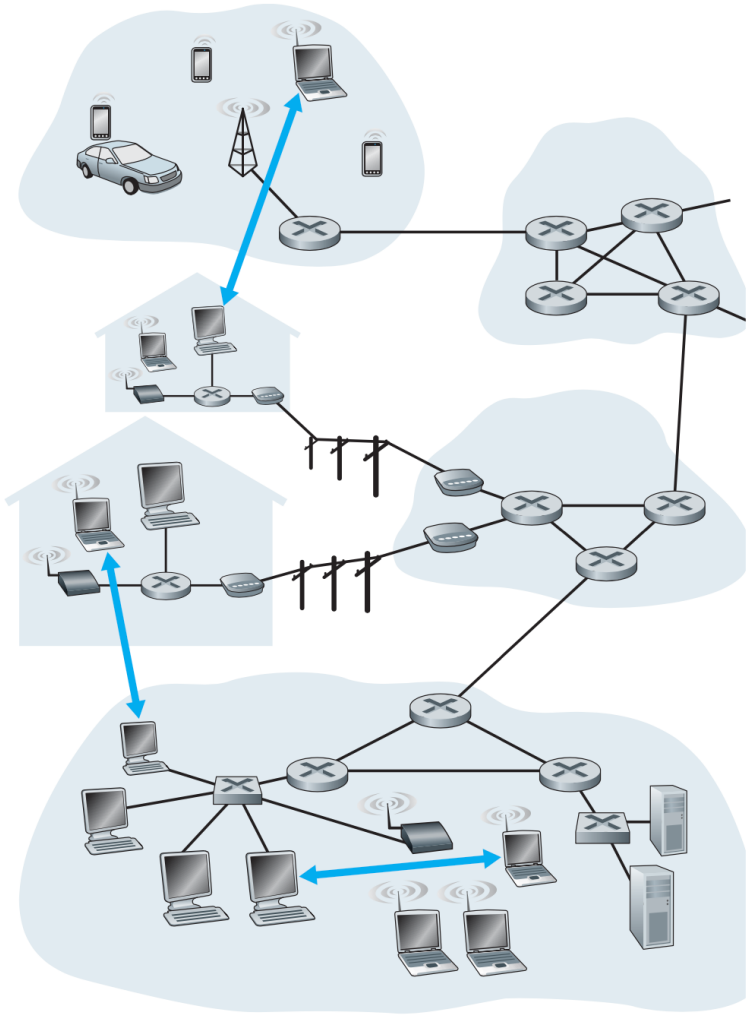
Network Application Architectures

- Client-server architecture
  - an always-on host, called the *server (has a fixed IP address and always on)*
  - hosts send service requests, called *client (contact server by sending a packet to the server's IP address)*
  - a popular social-networking site can quickly become overwhelmed if it has only one server handling all of its requests. For this reason, a data center, housing a large number of hosts, is often used to create a powerful *virtual server*
- P2P architecture
  - the application exploits direct communication between pairs of intermittently connected hosts, called *peers*
  - examples: file sharing (e.g., BitTorrent), peer-assisted, download acceleration (e.g., Xunlei), Internet Telephony (e.g., Skype), and IPTV (e.g., Kankan and PPstream)
  - advantage: self-scalability
  - challenges: ISP friendly, security, incentives
- Their hybrid
  - for many instant messaging applications, servers are used to track the IP addresses of users, but user-to-user messages are sent directly between user hosts (without passing through intermediate servers)

Network Application Architectures



a. Client-server architecture



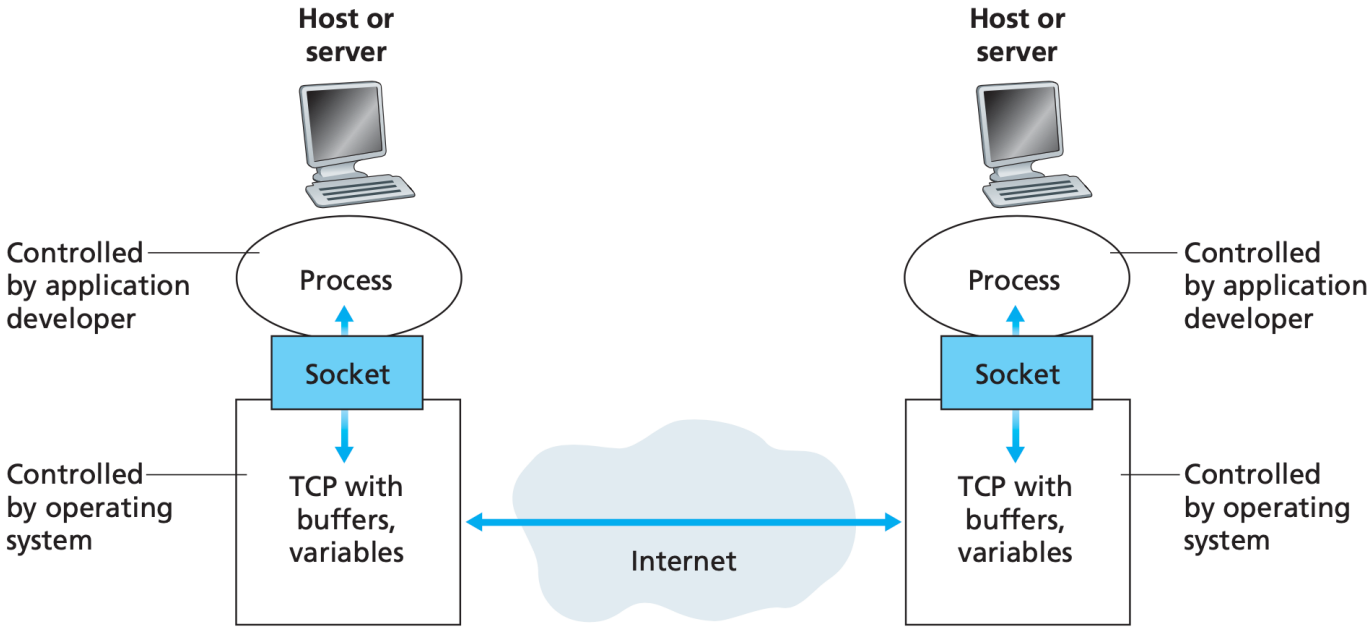
b. Peer-to-peer architecture

Processes Communicating

- *A process* can be thought of as a program that is running within an end system
  - When processes are running on the same end system, they can communicate with each other with interprocess communication
  - Processes on two different end systems communicate with each other by exchanging messages across the computer network
- Client and server process
  - In the context of a communication session between a pair of processes, the process that *initiates* the communication (that is, initially contacts the other process at the beginning of the session) is labeled as the client. The process that *waits to be contacted to* begin the session is the server
- A process sends messages into, and receives messages from, the network through a software interface called a *socket*
  - A socket is the interface between the application layer and the transport layer within a host
  - A socket is the API between the application and the network
  - The application developer has control of **everything** on the application-layer side of the socket but has little control of the transport-layer side of the socket. The only control that the application developer has on the transport-layer side is
    - (1) the choice of transport protocol and
    - (2) perhaps the ability to fix a few transport-layer parameters such as maximum buffer and maximum segment sizes

Processes Communicating

- A socket is the interface between the application layer and the transport layer within a host (cont'd)
- Addressing process
  - host is identified by its IP address
  - A destination port number is used to identify the receiving process (more specifically, the receiving socket) running in this host



Application processes, sockets, and underlying transport protocol

Transport Services Provided by the Internet

- TCP
  - connection-oriented service (exchange control info. before flowing data)
  - reliable data transfer service
  - congestion control
  - because privacy and other security issues have become critical for many applications, the Internet community has developed an enhancement for TCP, called *Secure Sockets Layer (SSL)*
- UDP
  - a no-frills, lightweight transport protocol, providing minimal services

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

Requirements of selected network applications

Transport Services Provided by the Internet

- Transport protocol services can be evaluated along four dimensions:
  - reliable data transfer (✓ TCP)
  - throughput (not guaranteed, but can be satisfactory with smart design)
  - timing (not guaranteed , but can be satisfactory with smart design)
  - security (✓ TCP)

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

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

Application Layer Protocols

- An application layer should define:
  - The types of messages exchanged, for example, request messages and response messages
  - The syntax of the various message types, such as the fields in the message and how the fields are delineated
  - The semantics of the fields, that is, the meaning of the information in the fields
  - Rules for determining when and how a process sends messages and responds to messages
- An application-layer protocol is only one piece of a network application
  - The Web application consists of many components, including
    a standard for document formats (that is, HTML),
    Web browsers (for example, Firefox and Microsoft Internet Explorer),
    Web servers (for example, Apache and Microsoft servers), and
    an application-layer protocol HTTP (defines the format and sequence of messages exchanged between browser and Web server)

Application Layer Protocols

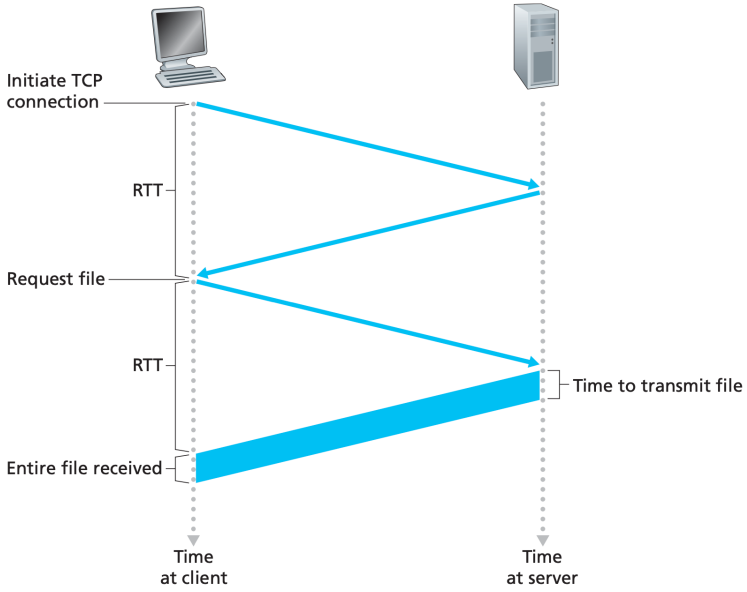
- The Web (HTTP)
- File transfer (FTP)
- E-mail (SMTP)
- DNS
- P2P applications

The Web and HTTP

- Overview of HTTP
  - The Hyper Text Transfer Protocol (HTTP), the Web's application-layer protocol, is at the heart of the Web
  - A **Web page** (also called a document) consists of objects. An object is simply a file—such as an HTML file, a JPEG image, a Java applet, or a video clip—that is addressable by a single **URL**
  - Each URL has two components: *the hostname* of the server that houses the object and *the object's path name*
  - Web browsers (such as Internet Explorer and Firefox) implement *the client side* of HTTP
  - Web servers, which implement *the server side* of HTTP, house Web objects, each addressable by a URL
  - HTTP defines how Web clients request Web pages from Web servers and how servers transfer Web pages to clients
  - HTTP uses TCP as its underlying transport protocol

The Web and HTTP

- Overview of HTTP (cont'd)
  - HTTP need not worry about lost data or the details of how TCP recovers from loss or reordering of data within the network. That is the job of TCP and the protocols in the lower layers of the protocol stack [*encapsulation*]
  - The server sends requested files to clients without storing any state information about the client (HTTP is stateless)
- Non-persistent connection
  - each request/response pair be sent over a separate TCP connection
  - round-trip time (RTT): the time it takes for a small packet to travel from client to server and then back to the client. Roughly, the total response time is two RTTs plus the transmission time at the server of the HTML file



Back-of-the-envelope calculation for the time needed to request and receive an HTML file

The Web and HTTP

- Persistent connection [in default]
  - this is more efficiently
- HTTP message format
  - HTTP request message

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
Connection: close
User-agent: Mozilla/5.0
Accept-language: fr
```

- The first line of an HTTP request message is called the **request line**; the subsequent lines are called the **header lines**. The request line has three fields: the method field, the URL field, and the HTTP version field. The method field can take on several different values, including GET, POST, HEAD, PUT, and DELETE. The great majority of HTTP request messages use the GET method

The Web and HTTP

- HTTP message format
  - HTTP request message (cont'd)

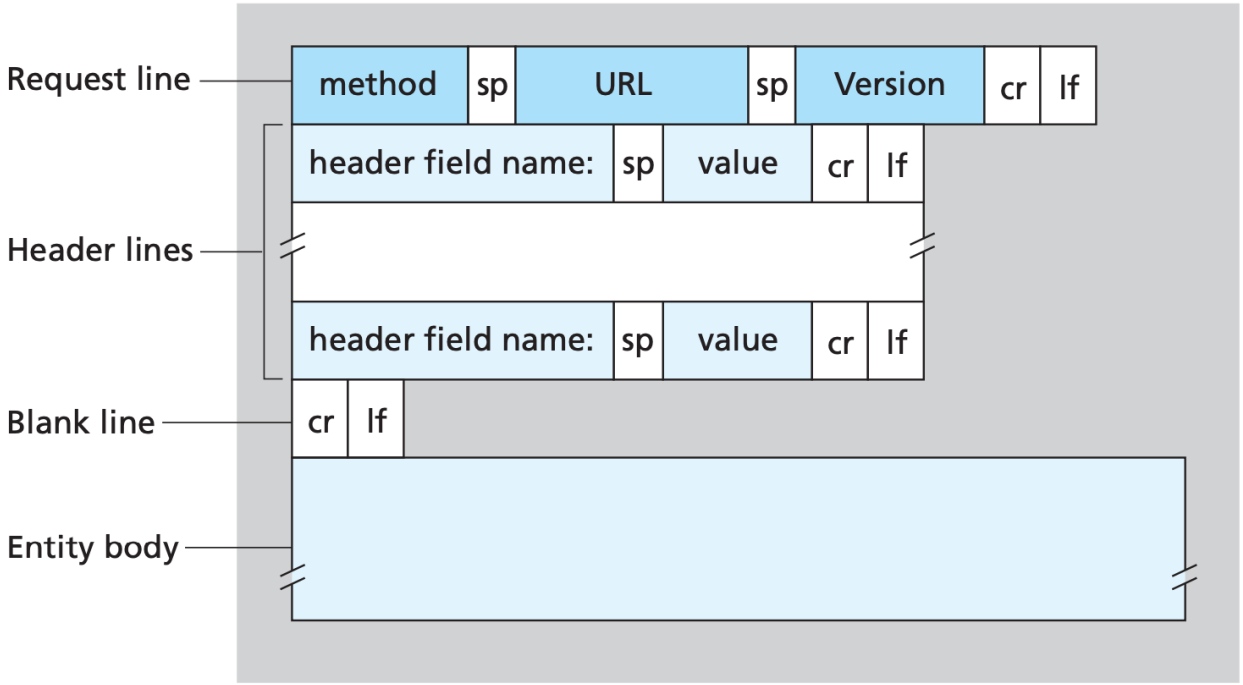
```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
Connection: close
User-agent: Mozilla/5.0
Accept-language: fr
```

    - The information provided by `Host: www.someschool.edu` is required by Web proxy caches
    - By including the `Connection: close` header line, the browser is telling the server that it doesn't want to bother with *persistent* connections
    - The next line `User-agent: Mozilla/5.0` is useful because the server can actually send different versions of the same object to different types of user agents. (Each of the versions is addressed by the same URL.) [webpage adaption]
    - The entity body is empty with the GET method, but is used with the POST method. An HTTP client often uses the POST method when the user fills out a form—for example, when a user provides search words to a search engine

      (e.g., `www.somesite.com/animalsearch?monkeys&bananas`)

The Web and HTTP

- HTTP message format
  - HTTP request message (cont'd)



General format of an HTTP request message

The Web and HTTP

- HTTP message format
  - HTTP response message

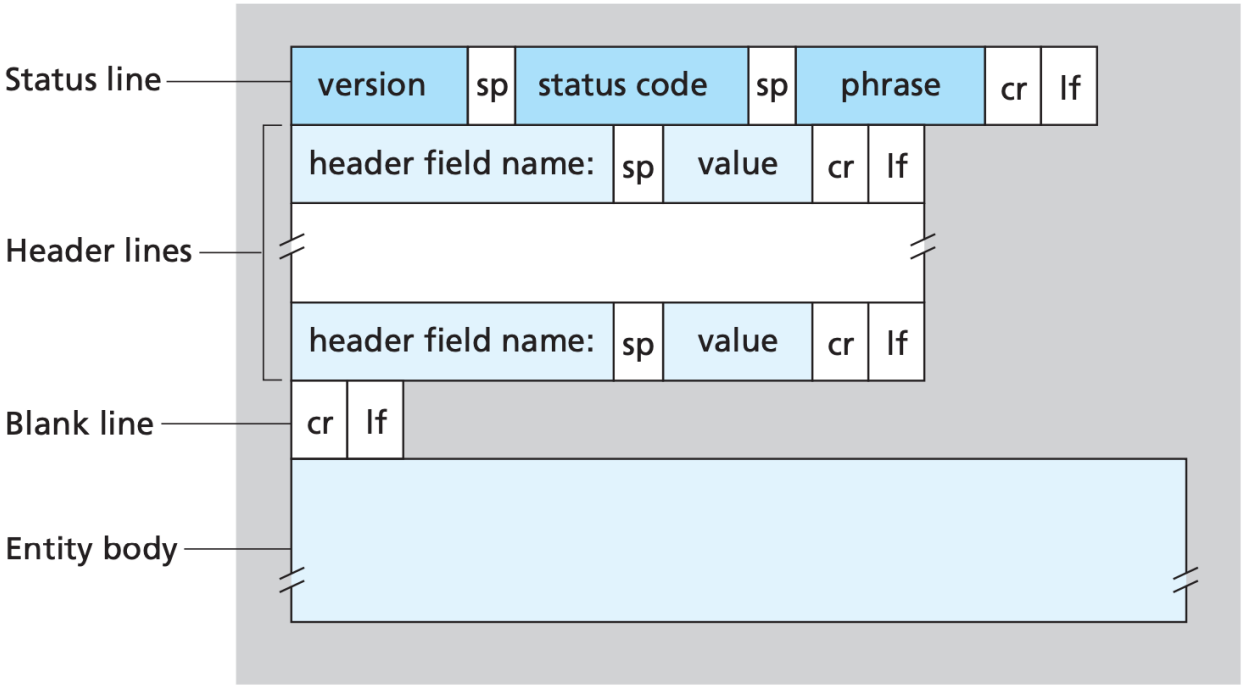
```
HTTP/1.1 200 OK Connection: close
Date: Tue, 09 Aug 2011 15:44:04 GMT
Server: Apache/2.2.3 (CentOS)
Last-Modified: Tue, 09 Aug 2011 15:11:03 GMT
Content-Length: 6821
Content-Type: text/html

(data data data data data ...)
```

    - An initial status line, six header lines, and then the entity body
    - The status line has three fields: the protocol version field, a status code, and a corresponding status message
      - 200 OK, 301 Moved Permanently, 400 Bad Request, 404 Not Found, 505 HTTP Version Not Supported
    - *The Date: header* is the time when the server retrieves the object from its file system, inserts the object into the response message, and sends the response message
    - *The Last-Modified: header* is critical for object caching, both in the local client and in network cache servers (also known as proxy servers)

The Web and HTTP

- HTTP message format
  - HTTP response message (cont'd)



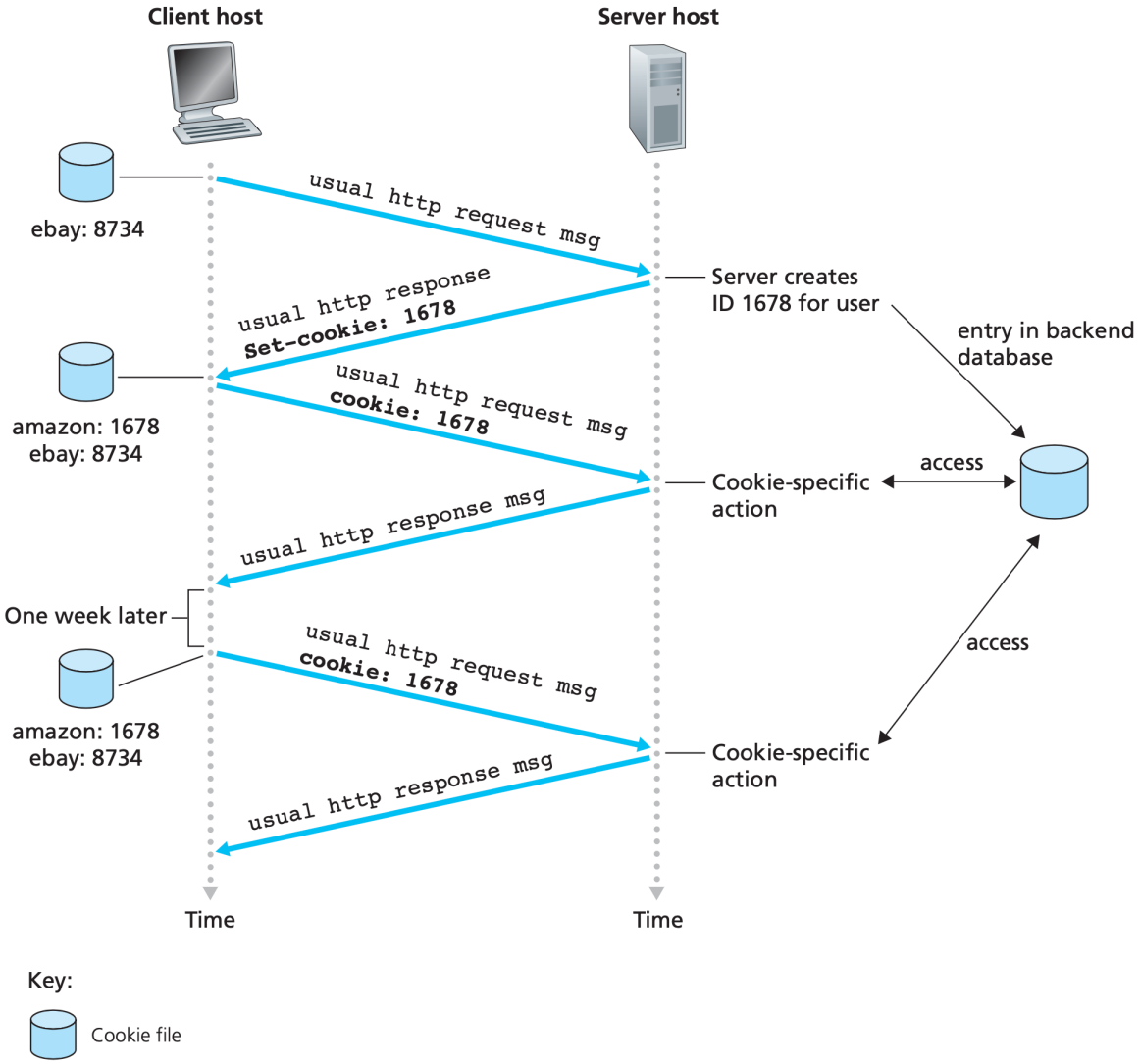
General format of an HTTP response message

The Web and HTTP

- User-server interaction: cookies
  - Although web server is stateless, it is often desirable for a Web site to identify users, either because the server wishes to restrict user access or because it wants to serve content as a function of the user identity
  - Use cookies! it has four components:
    (1) a cookie header line in the HTTP response message;
    (2) a cookie header line in the HTTP request message;
    (3) a cookie file kept on the user's end system and managed by the user's browser; and
    (4) a back-end database at the Web site
  - Typical usage: shopping cart
  - *Cookies can be used to create a user session layer on top of stateless HTTP*

The Web and HTTP

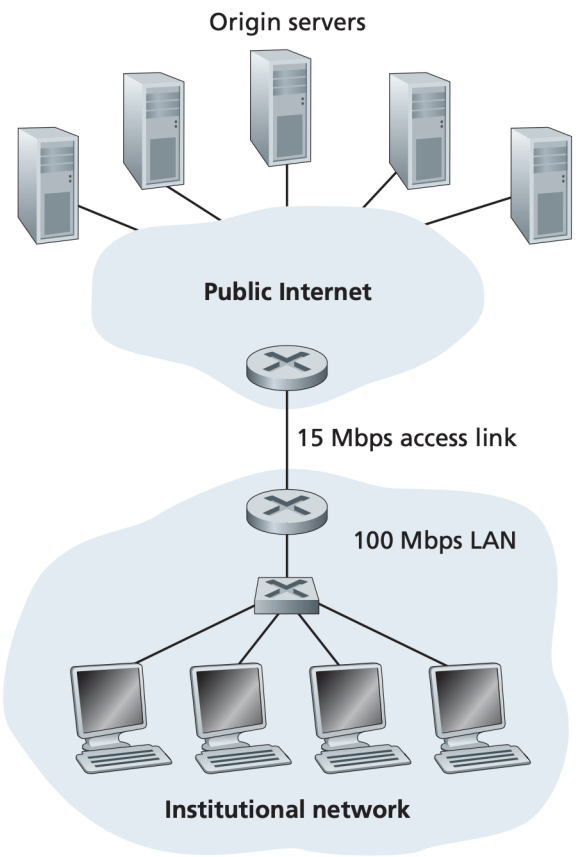
- User-server interaction: cookies (cont'd)



Keeping user state with cookies

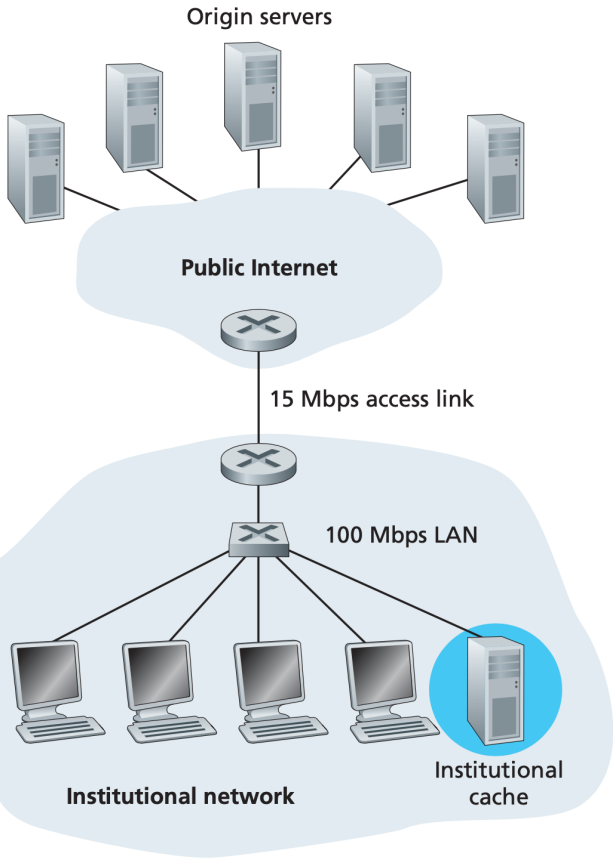
The Web and HTTP

- Web caching
  - A Web cache—also called a proxy server — is a network entity that satisfies HTTP requests on the behalf of an origin Web server
  - Why we use web caching?
    - substantially reduce the response time and traffic



Instead of upgrading access link bandwidth, adding a cache costs much lower

→



The Web and HTTP

- Web caching (cont'd)
  - Content Distribution Networks (CDNs) are designed based on web caching: shared CDNs and dedicated CDNs
- The conditional GET
  - a mechanism allows a cache to verify whether its obj. is updated
  - the conditional GET

```
GET /fruit/kiwi.gif HTTP/1.1
Host: www.exotiquecuisine.com
If-modified-since: Wed, 7 Sep 2011 09:23:24
```

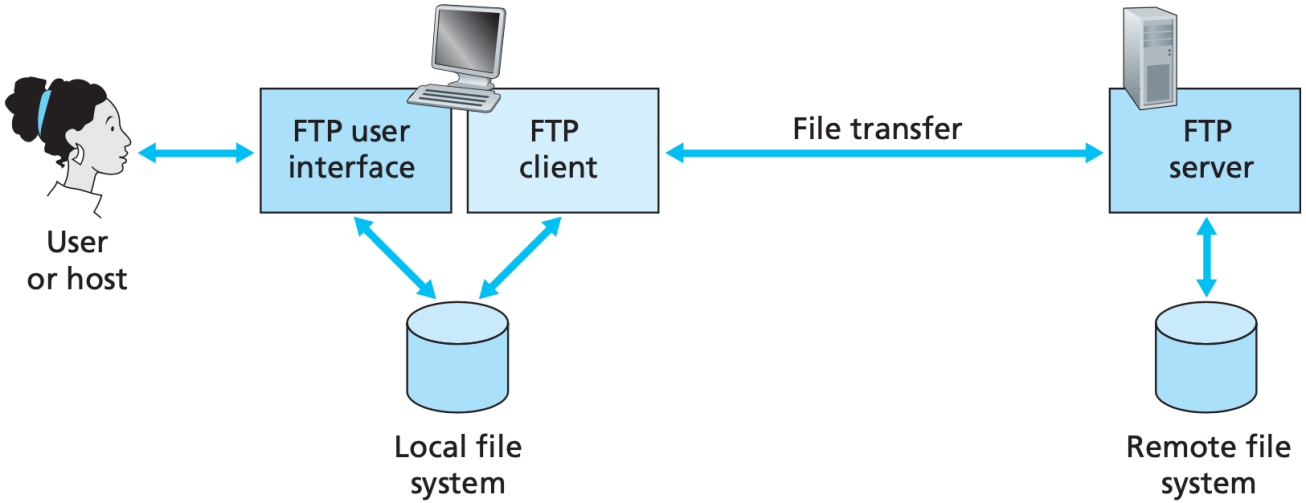
  - the corresponding response

```
HTTP/1.1 304 Not Modified
Date: Sat, 15 Oct 2011 15:39:29
Server: Apache/1.3.0 (Unix)

(empty entity body)
```

File Transfer and FTP

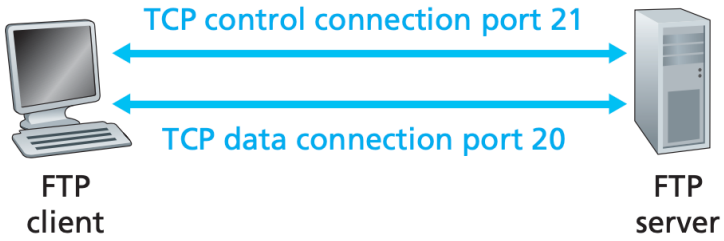
- The user first provides the hostname of the remote host, causing the FTP client process in the local host to establish a TCP connection with the FTP server process in the remote host
- The user then provides the user identification and password, which are sent over the TCP connection as part of FTP commands
- Once the server has authorized the user, the user copies one or more files stored in the local file system into the remote file system (or vice versa)



FTP moves files between local and remote file systems

File Transfer and FTP

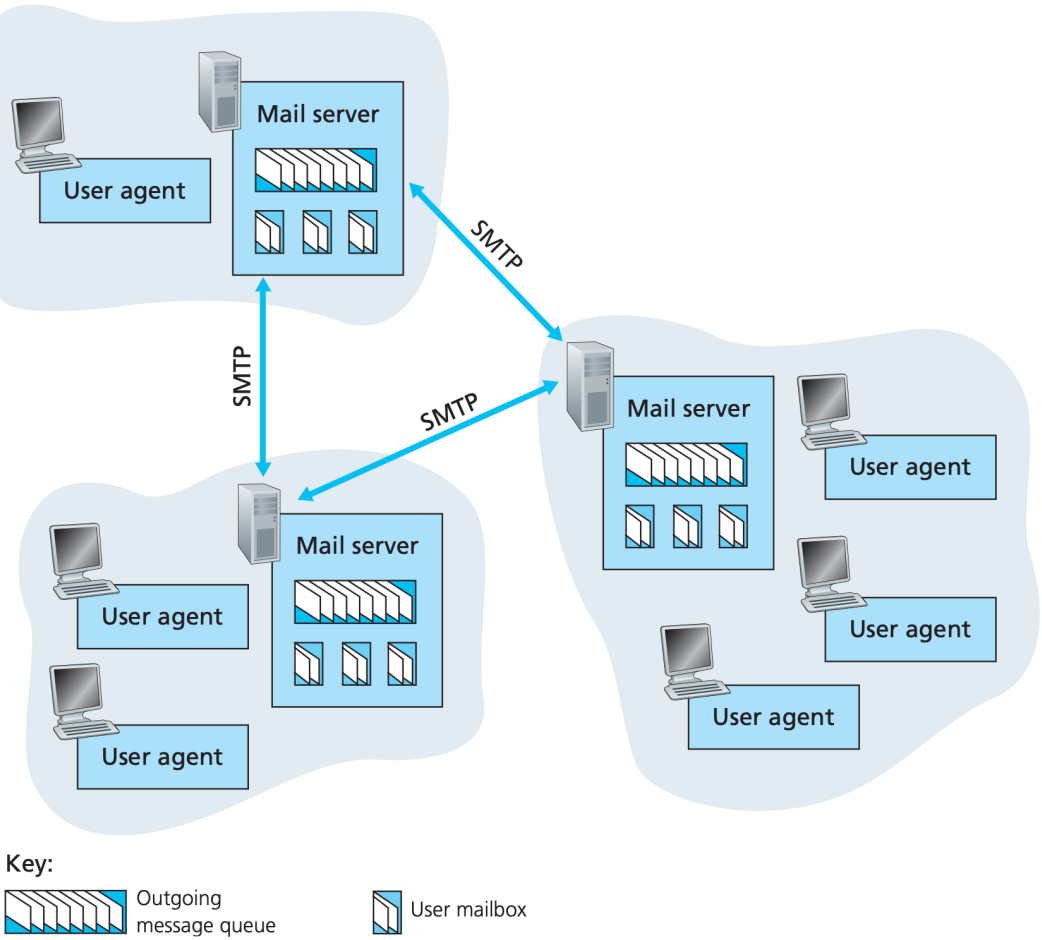
- FTP uses two parallel TCP connections to transfer a file, a control connection and a data connection
  - The control connection is used for sending control information such as user identification, password, commands to change remote directory, and commands to "put" and "get" files
  - The data connection is used to actually send a file
- FTP sends exactly one file over the data connection and then closes the data connection. If, during the same session, the user wants to transfer another file, FTP opens another data connection
- FTP maintains **state** (current DIR)



Control and data connections

E-mail and SMTP

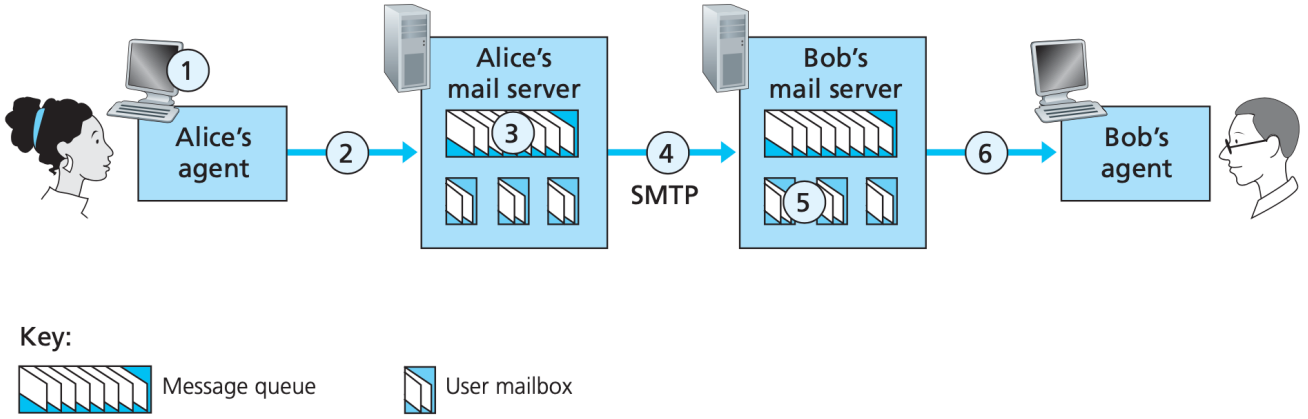
- The Internet e-mail system has three components
  - user agents
  - mail servers
  - SMTP



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

E-mail and SMTP

- Procedure:
  - Alice invokes her user agent for e-mail, provides Bob's e-mail address (for example, `bob@someschool.edu`), composes a message, and instructs the user agent to send the message
  - Alice's user agent sends the message to her mail server, where it is placed in a message queue (SMTP) [step 1 is required because re-try can be executed]
  - The client side of SMTP, running on Alice's mail server, sees the message in the message queue. It opens a TCP connection to an SMTP server, running on Bob's mail server
  - After some initial SMTP handshaking, the SMTP client sends Alice's message into the TCP connection (SMTP on top of TCP)
  - At Bob's mail server, the server side of SMTP receives the message. Bob's mail server then places the message in Bob's mailbox
  - Bob invokes his user agent to read the message at his convenience (POP3, IMAP, or HTTP)



E-mail and SMTP

- SMTP does not normally use intermediate connection, but a direct connection from the sender's mail server to the receiver's mail server
- How SMTP transfer the msg?
  - TCP establish connection from client SMTP to port 25 of server SMTP
  - app-layer handshake: SMTP client indicates the e-mail address of the sender and the e-mail address of the recipient
  - client SMTP sends the msg
- Compare with HTTP
  - HTTP is **a pull protocol** while SMTP is **a push protocol**
  - SMTP requires each message, including the body of each message, to be in 7-bit ASCII format
  - HTTP encapsulates each object in its own HTTP response message. Internet mail places all of the message's objects into one message

E-mail and SMTP

- Mail access protocols
  - a typical user runs a user agent on the local PC but accesses its mailbox stored on an always-on shared mail server. This mail server is shared with other users and is typically maintained by the user's ISP (for example, university or company)
  - The last step is a pull process while SMTP is a pull protocol. So how the recipient pull emails from his mail server? → mail access protocols!
    - Post Office Protocol — Version 3 (POP3)

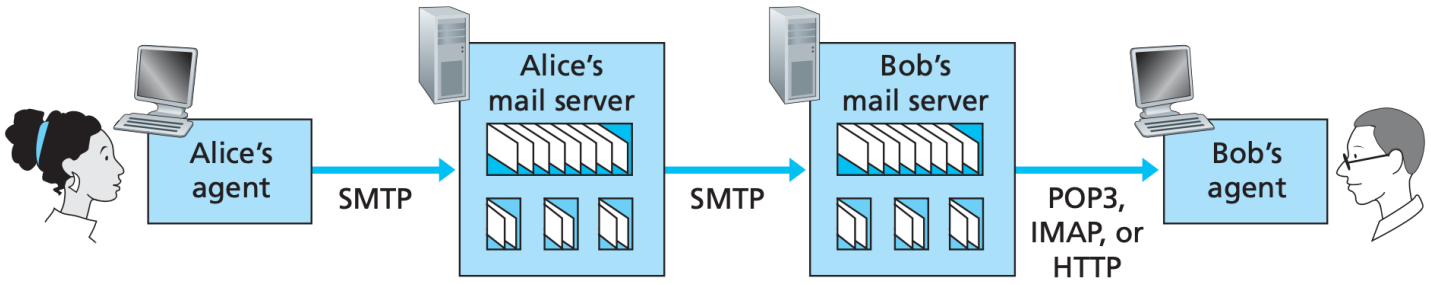
      [user agent opens a TCP conn. to port 110 of mail server]
    - Internet Mail Access Protocol (IMAP)

      [unlike POP3, an IMAP server maintains user state information across IMAP sessions—for example, the names of the folders and which messages are associated with which folders. Also permit a user agent to obtain components of messages]
    - HTTP

      [web-based email is the most popular case nowadays. User agent is an ordinary Web browser, and the user communicates with its remote mailbox via HTTP]

E-mail and SMTP

- Conclusion
  - SMTP is used to transfer mail from the sender's mail server to the recipient's mail server; SMTP is also used to transfer mail from the sender's user agent to the sender's mail server. A mail access protocol, such as POP3, IMAP, and HTTP, is used to transfer mail from the recipient's mail server to the recipient's user agent



Pay attention to these protocols used

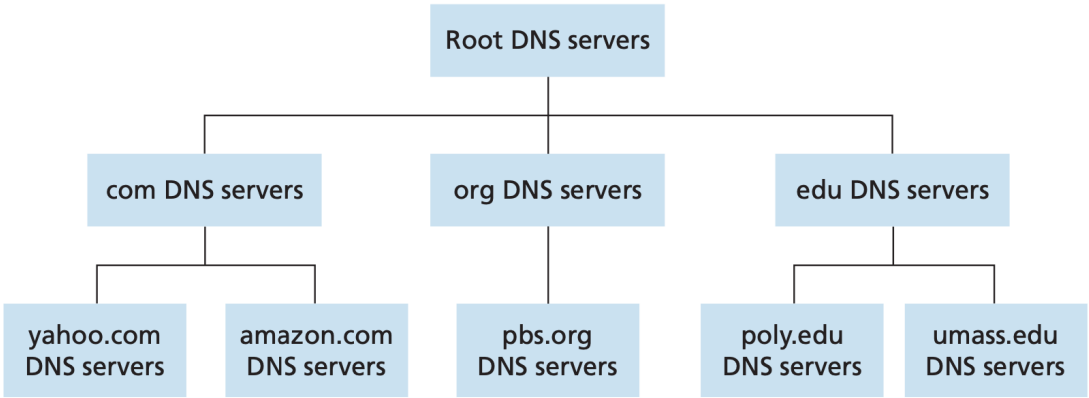
Domain Name System (DNS)

- DNS is (1) a distributed database implemented in a hierarchy of DNS servers, and (2) an application-layer protocol that allows hosts to query the distributed database
- DNS is commonly employed by other application-layer protocols — including HTTP, SMTP, and FTP — *to translate user-supplied hostnames to IP addresses*
- DNS protocol run over UDP on port 53
- How HTTP parses a hostname:
  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.
  4. The DNS client eventually 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.
- The desired IP address is often cached in a "nearby" DNS server, which helps to reduce DNS network traffic as well as the average DNS delay

Domain Name System (DNS)

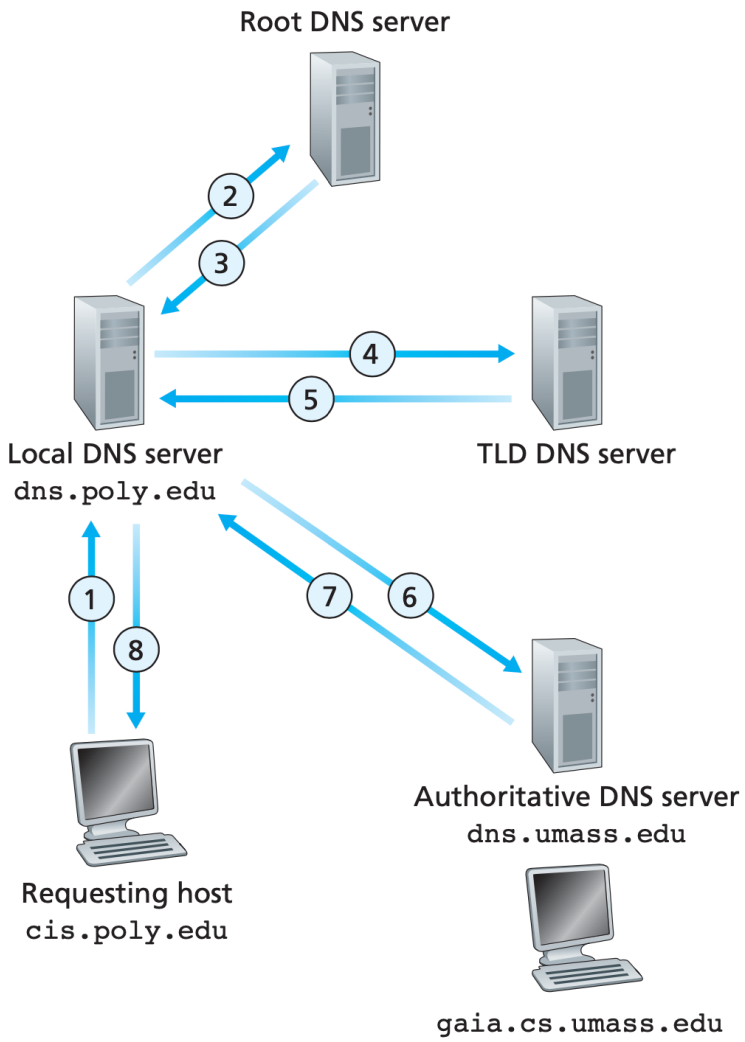
- Other important services provided by DNS:
  - host aliasing (CNAME)
  - mail server aliasing
  - load distribution

  A hostname may be replicated over many web servers with different IP addresses. These IP addresses are associated with one canonical hostname. DNS rotation distributes the traffic onto different web servers for load balancing.
- How to query IP address from those distributed and hierarchical DNS DB?
  - three classes of DNS servers — root DNS servers [for redundancy], top-level domain (TLD) DNS servers [com, org, net, edu, gov, uk, fr, ca, and jp, etc], and authoritative DNS servers [organization with publicly accessible hosts on the Internet must provide publicly accessible DNS records that map the names of those hosts to IP addresses] — organized in a hierarchy
  - besides, Each ISP — such as a university, an academic department, an employee's company, or a residential ISP—has a local DNS server (aka default name server)



Domain Name System (DNS)

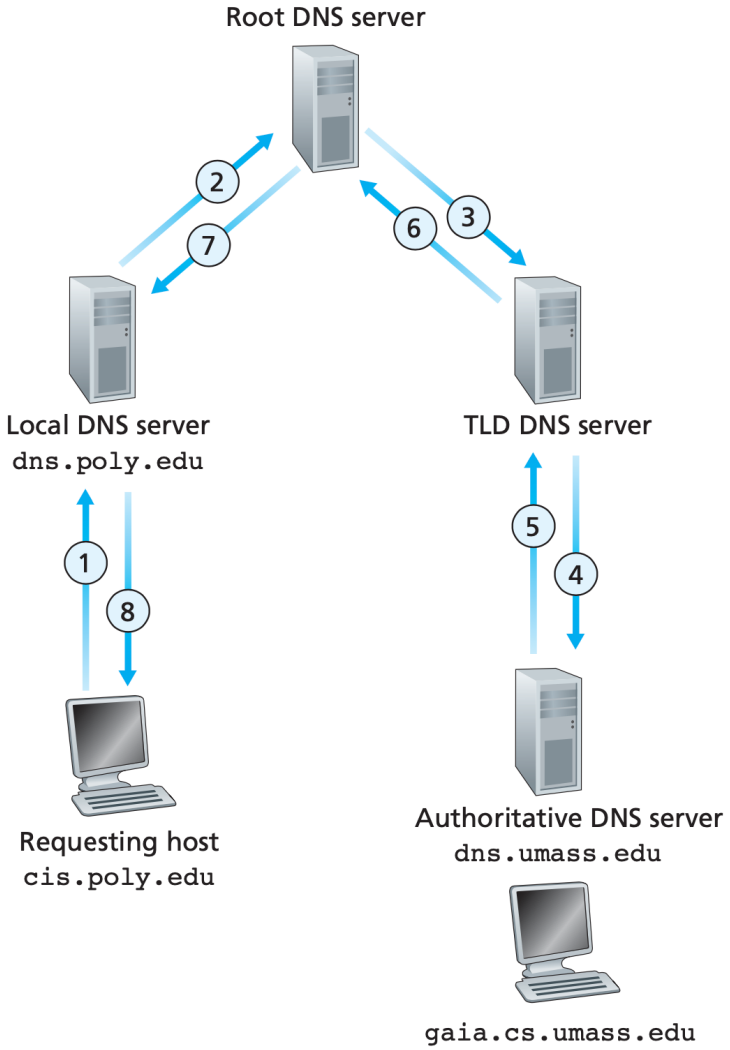
- Query pattens
  - recursive and iterative patterns



The query from the requesting host to the local DNS server is recursive, and the remaining queries are iterative (this is the commonly used pattern)

Domain Name System (DNS)

- Query pattens (cont'd)
  - recursive patterns



Recursive queries in DNS

Domain Name System (DNS)

- DNS caching
  - If a hostname/IP address pair is cached in a DNS server and another query arrives to the DNS server for the same hostname, the DNS server can provide the desired IP address, even if it is not authoritative for the hostname
- DNS records
  - it's a four tuple: (`Name, Value, Type, TTL`)
    - `TTL` is the time to live of the record
    - `Type`:

      (1) **A** [the standard hostname-to-IP address mapping, `Name` is a hostname, `Value` is the IP address]

      (2) **NS** [`Name` is a domain (such as `foo.com`) and `Value` is the hostname of an authoritative DNS server that knows how to obtain the IP addresses for hosts in the domain]

      (3) **CNAME** [`Value` is a canonical hostname for the alias hostname `Name`]

      (4) **MX** [`Value` is the canonical name of a mail server that has an alias hostname `Name`]
  - If a DNS server is authoritative for a particular hostname, then the DNS server will contain a Type A record for the hostname (not authoritative may cache a A record)
  - If a server is not authoritative for a hostname, then the server will contain a Type NS record for the domain that includes the hostname; it will also contain a Type A record that provides the IP address of the DNS server in the Value field of the NS record

Domain Name System (DNS)

- DNS messages
  - both query and reply messages have the same format
  - 12 bytes header section, question section, answer section, authority section, additional section
- Inserting records into the DNS DB
  - register your domain name at a registrar
  - provide the registrar with the names and IP addresses of your primary and secondary authoritative DNS servers
  - for each of these two authoritative DNS servers, the registrar would then make sure that a Type NS and a Type A record are entered into the TLD com servers

    e.g.,

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

P2P applications

- P2P file systems
  - In P2P file distribution, each peer can redistribute any portion of the file it has received to any other peers, thereby assisting the server in the distribution process. As of 2012, the most popular P2P file distribution protocol is BitTorrent
  - Distribute time analysis

    - The distribution time is the time it takes to get a copy of the file to all N peers

    - For CS architecture:

$$D_{cs} \geq \max\left\{\frac{NF}{u_s}, \frac{F}{\min\{d_1, ..., d_N\}}\right\}$$

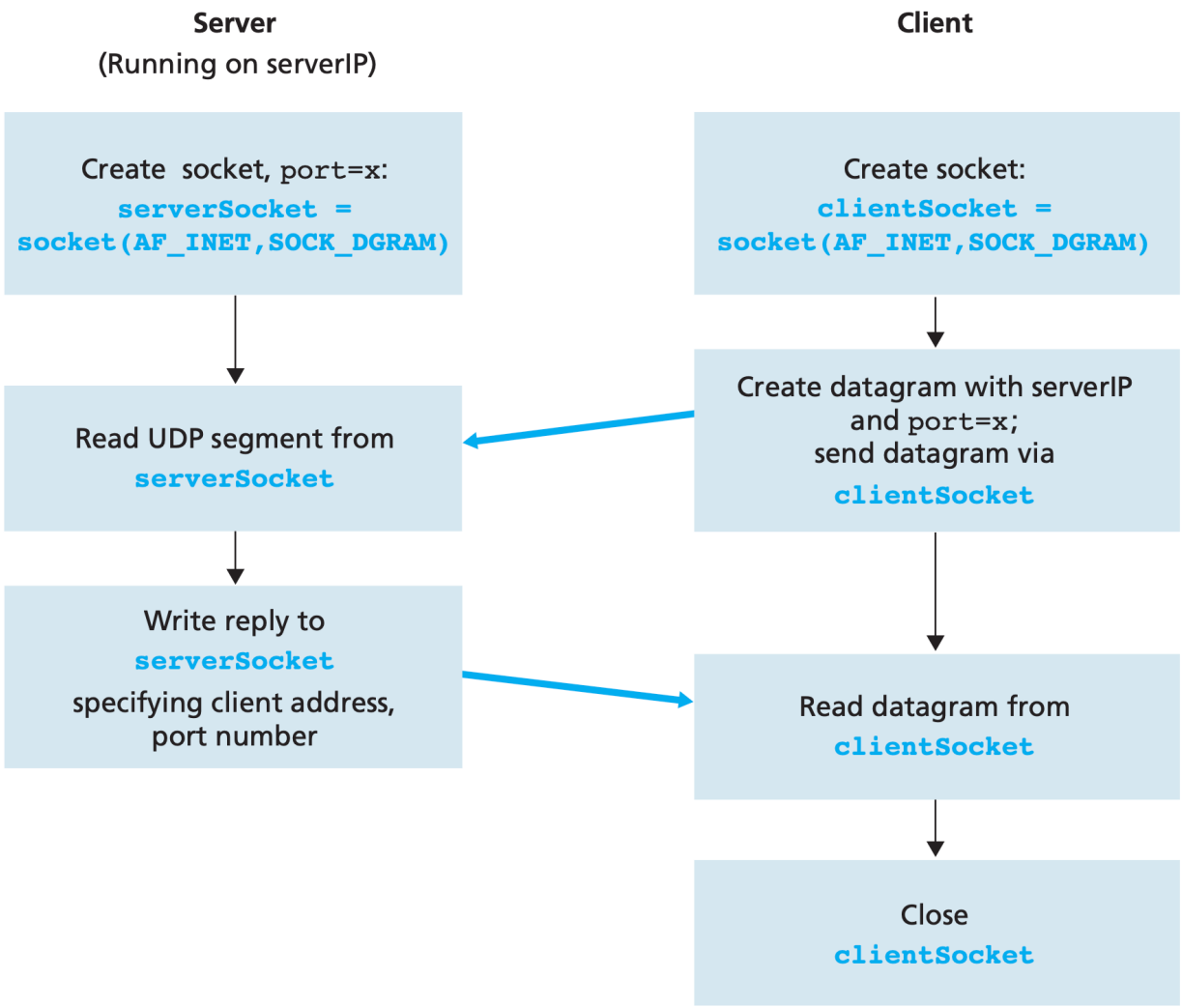
    - For P2P architecture:

$$D_{p2p} \geq \max\left\{\frac{F}{u_s}, \frac{F}{\min\{d_1, ..., d_N\}}, \frac{NF}{u_s + \sum_{i=1}^{N} u_i}\right\}$$

- Distributed Hash Tables (DHT)

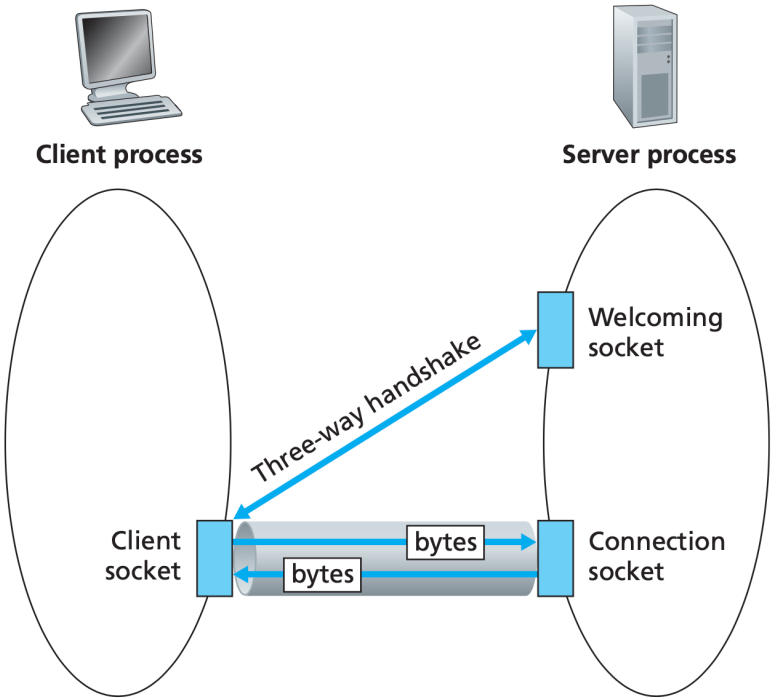
Socket Programming

- Socket programming with UDP



Socket Programming

- Socket programming with TCP
  - the welcoming door is a TCP socket object that we call `serverSocket`; the newly created socket dedicated to the client making the connection is called `connectionSocket`



The `TCPServer` process has two sockets

Socket Programming

- Socket programming with TCP (cont'd)