Computer Networks

Malik Algazaeery

Chapter2-Application Layer

Chapter-2. agenda:

- Principles of Network Applications.
- Network Application Architectures.
- The Interface Between the Process and the Computer Network.
- Addressing Processes.
- Transport Services Available to Applications.
- Transport Services Provided by the Internet:
  - TCP Services
  - UDP Services
- Application layer protocols:
  - HTTP and the Web: HTTP, Cookies, Web caching.
  - E-Mail in the Internet: SMTP, IMAP.
  - DNS the Domain name service.
- P2P Applications and BitTorrent.
- Summery.

Application Layer

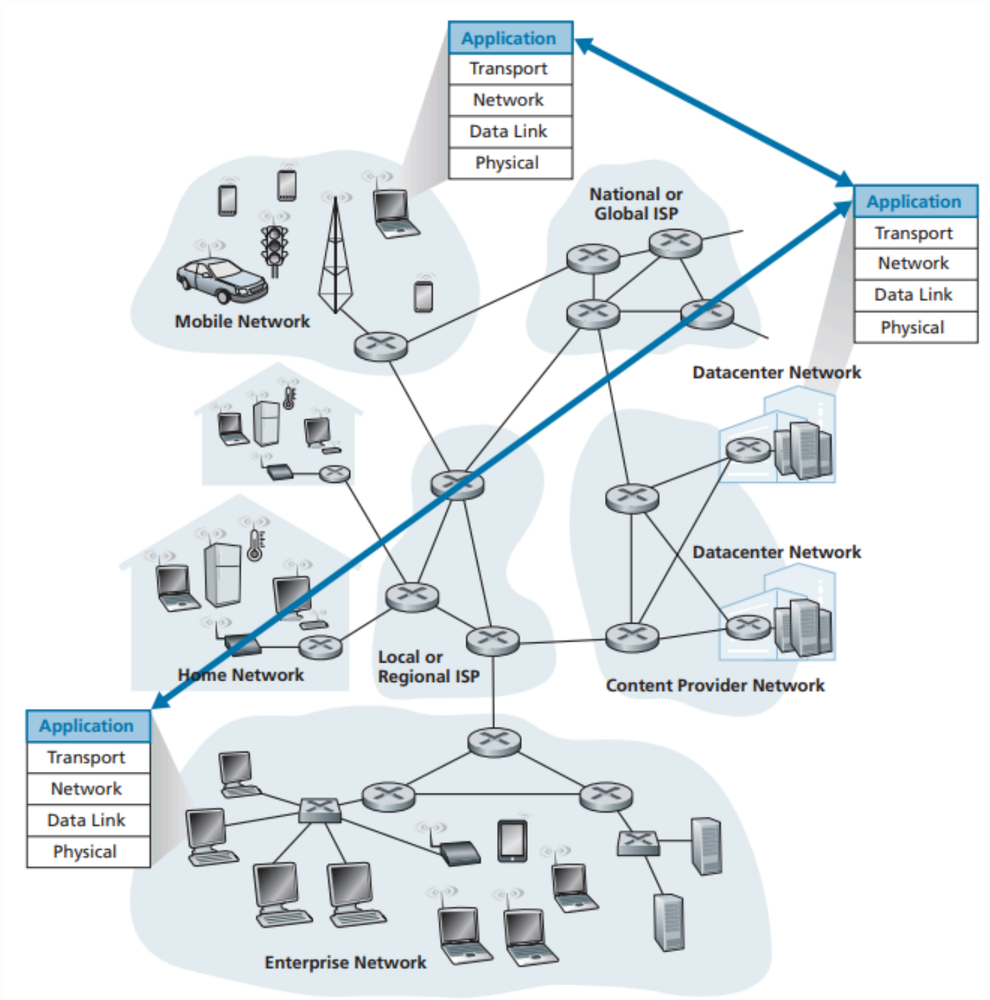
These applications have been the driving force behind the Internet's success, motivating people in homes, schools, governments, and businesses to make the Internet an integral part of their daily activities.

- Internet applications include the classic text-based applications that became popular in the 1970s and 1980s: text e-mail, remote access to computers, file transfers, and newsgroups. They include the killer application of the mid-1990s, the World Wide Web, encompassing Web surfing, search, and electronic commerce.

- highly compelling applications continue to emerge, including voice over IP and video conferencing such as Skype, Facetime, and Google Hangouts; user generated video such as YouTube and movies on demand such as Netflix; and multiplayer online games such as Second Life and World of Warcraft.

Principles of Network Applications

- Network application development is writing programs that run on different end systems and communicate with each other over the network.
- For example, in the Web application there are two distinct programs that communicate with each other: the browser program running in the user's host, and the Web server program running in the Web server host.
- when developing network application, you need to write software that will run on multiple end systems.
- No need to write software that runs on network-core devices, such as routers or switches.
- network-core devices do not function at the application layer but instead function at lower layers specifically at the network layer and below.

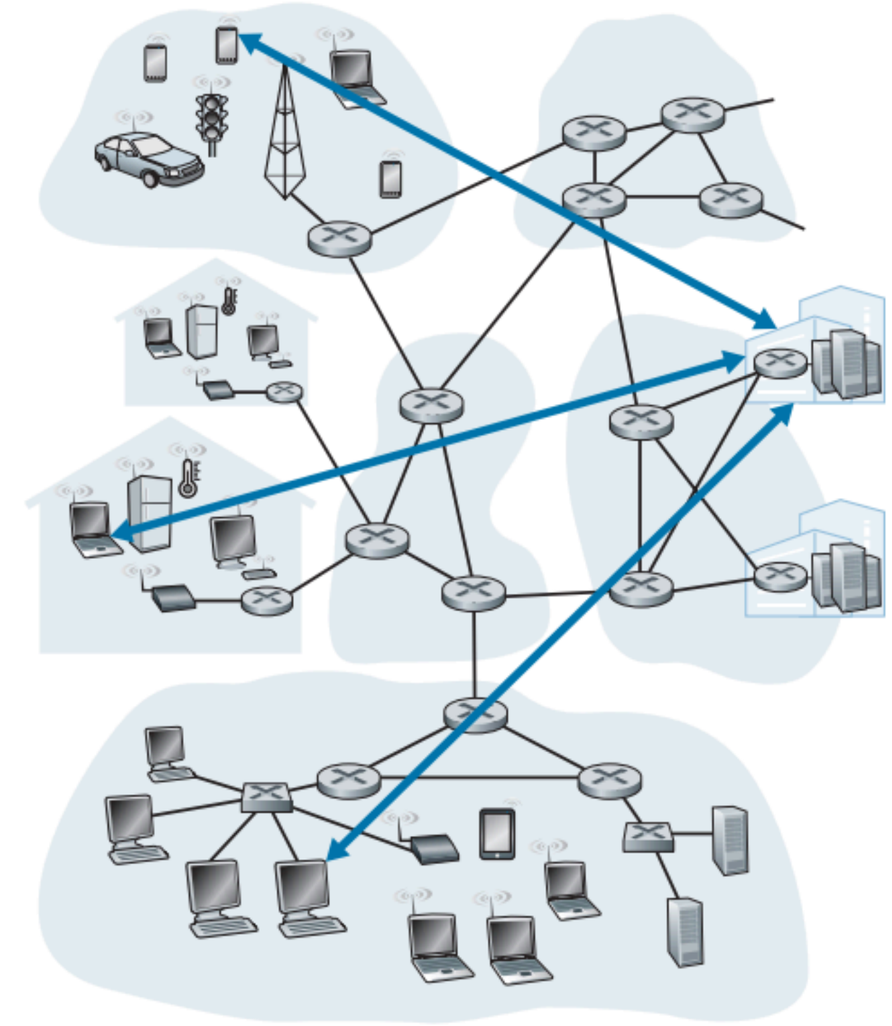


Network Application Architectures

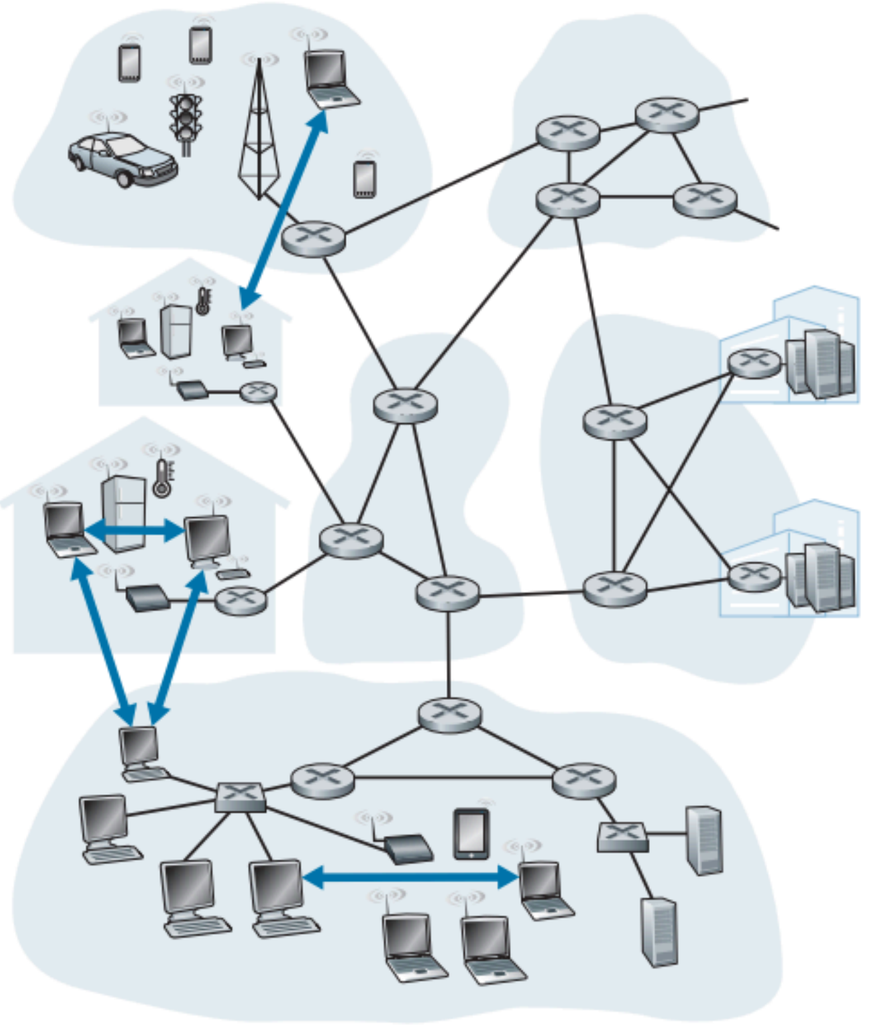
application’s architecture is distinctly different from the network architecture. The application architecture dictates how the application is structured over the various end systems. Two predominant architectural paradigms used in modern network applications: the client-server architecture or the peer-to-peer (P2P) architecture.

- In a client-server, there is an always-on host, called the server, which services requests from many other hosts, called clients. Example is the Web application.
- In client-server architecture, clients do not directly communicate with each other.
- server has a fixed, well-known address, called an IP address.
- known applications with a client-server architecture include the Web, FTP, Telnet, and e-mail.

- In a P2P architecture, there is minimal (or no) reliance on dedicated servers in data centers.
- Applications communicate between pairs of intermittently connected hosts, called peers.
- Example of a popular P2P application is the file-sharing application BitTorrent.
- Most compelling features of P2P architectures is their selfscalability.



a. Client-server architecture



b. Peer-to-peer architecture

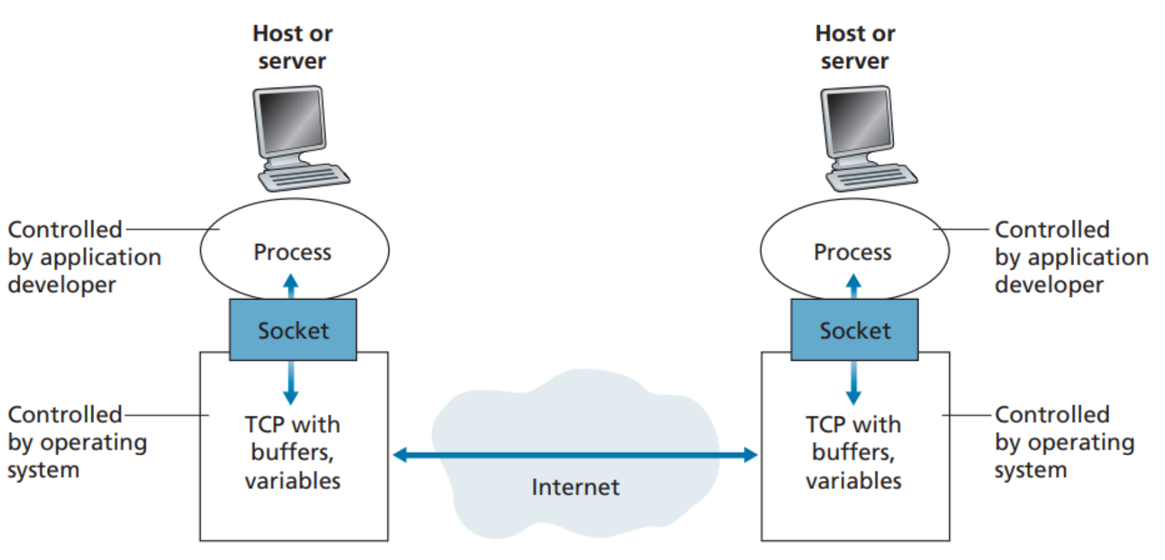
Client and Server Processes

A network application consists of pairs of processes that send messages to each other over a network.

- For example, in the Web application a client browser process exchanges messages with a Web server process.
- In a P2P file-sharing system, a file is transferred from a process in one peer to a process in another peer.
- For each pair of communicating processes, we typically label one of the two processes as the client and the other process as the server.
- in P2P file sharing, a process can be both a client and a server. Indeed, a process in a P2P file-sharing system can both upload and download files.
- 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.

The Interface Between the Process and the Computer Network

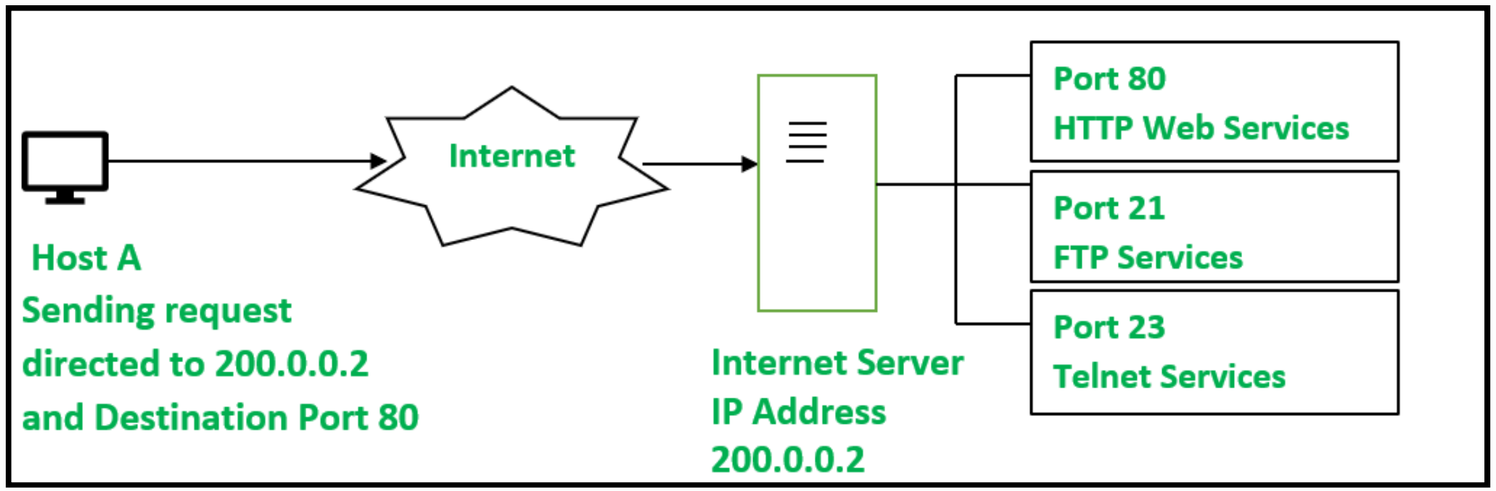
- A process sends messages into, and receives messages from, the network through a software interface called a socket.
- As shown in the figure, a socket is the interface between the application layer and the transport layer within a host. It is also referred to as the Application Programming Interface (API) between the application and the network.
- 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 transportlayer 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



In addition to knowing the address of the host (IP) to which a message is destined, the sending process must also identify the receiving process (more specifically, the receiving socket). A destination port number serves this purpose.

Addressing Processes

- In the Internet, the host is identified by its IP address. We'll discuss IP addresses in great detail in the network layer. For now, all we need to know is that an IP address is a 32-bit quantity that we can think of as uniquely identifying the host.
- In addition to knowing the address of the host to which a message is destined, the sending process must also identify the receiving process (more specifically, the receiving socket) running in the host. This information is needed because in general a host could be running many network applications. A destination port number serves this purpose.
- Popular applications have been assigned specific port numbers. For example, a Web server is identified by port number 80. A mail server process (using the SMTP protocol) is identified by port number 25.



https://www.geeksforgeeks.org/difference-between-source-port-and-destination-port/

Transport Services Available to Applications

- The Internet, provide more than one transport-layer protocol. When you develop an application, you must choose one of the available transport-layer protocols.
- How do you make this choice?
- classify the possible services along four dimensions: reliable data transfer, throughput, timing, and security.

Reliable Data Transfer:

- packets can get lost within a computer network. For example, a packet can overflow a buffer in a router, or can be discarded by a host or router.
- If a protocol provides a guaranteed data delivery (the data sent by one end of the application is delivered correctly and completely to the other end of the application) service, it is said to provide reliable data transfer.
- When a transport-layer protocol doesn’t provide reliable data transfer, some of the data sent by the sending process may never arrive at the receiving process. This may be acceptable for loss-tolerant applications

Throughput:

- The available throughput can fluctuate with time.
- Transport-layer protocol could provide guaranteed available throughput at some specified rate.
- Example, if an Internet telephony application encodes voice at 32 kbps, it needs to send data into the network and have data delivered to the receiving application at this rate. If the transport protocol cannot provide this throughput, the application would need to encode at a lower rate or may have to give up.
- Applications that have throughput requirements are said to be bandwidth-sensitive applications.
- elastic applications can make use of as much, or as little, throughput as happens to be available. ex: E-mail, file transfer, and Web transfers.

Timing:

- A transport-layer protocol can also provide timing guarantees.
- Example: every bit that the sender pumps into the socket arrives at the receiver’s socket no more than 100 msec later.
- Long delays in Internet telephony tend to result in unnatural pauses in the conversation.
- In a multiplayer game or virtual interactive environment, a long delay between taking an action and seeing the response from the environment makes the application feel less realistic.

Security:

- Transport protocol can provide an application with one or more security services.
- Example, in the sending host, a transport protocol can encrypt all data transmitted by the sending process, and in the receiving host, the transport-layer protocol can decrypt the data before delivering the data to the receiving process.

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–1 Mbps 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
Smartphone messaging	No loss	Elastic	Yes and no

Transport Services Provided by the Internet:

The Internet makes two transport protocols available to applications, UDP and TCP. When you create a new network application for the Internet, one of the first decisions you have to make is whether to use UDP or TCP. Each of these protocols offers a different set of services to the invoking applications.

TCP Services:

- Connection-oriented service: TCP has the client and server exchange transport-layer control information with each other before the application-level messages begin to flow. This so-called handshaking procedure alerts the client and server, allowing them to prepare for an onslaught of packets.
- After the handshaking phase, a TCP connection is said to exist between the sockets of the two processes.
- The connection is a full-duplex connection in that the two processes can send messages to each other over the connection at the same time.
- When the application finishes sending messages, it must tear down the connection.
- Reliable data transfer service: The communicating processes can rely on TCP to deliver all data sent without error and in the proper order.
- Congestion-control mechanism.

UDP Services:

- UDP is connectionless, so there is no handshaking before the two processes start to communicate.
- unreliable data transfer service—that is, when a process sends a message into a UDP socket, UDP provides no guarantee that the message will ever reach the receiving process.
- messages that do arrive at the receiving process may arrive out of order.
- does not include a congestion-control mechanism, so the sending side of UDP can pump data into the layer below (the network layer) at any rate it pleases.

Services Not Provided by Internet Transport Protocols:

- TCP provides reliable end-to-end data transfer. And we also know that TCP can be easily enhanced at the application layer with TLS to provide security services.
- throughput and timing guarantees services are not provided by today’s Internet transport protocols.

Application	Application-Layer Protocol	Underlying Transport Protocol
Electronic mail	SMTP [RFC 5321]	TCP
Remote terminal access	Telnet [RFC 854]	TCP
Web	HTTP 1.1 [RFC 7230]	TCP
File transfer	FTP [RFC 959]	TCP
Streaming multimedia	HTTP (e.g., YouTube), DASH	TCP
Internet telephony	SIP [RFC 3261], RTP [RFC 3550], or proprietary (e.g., Skype)	UDP or TCP

Application-Layer Protocols

- Network processes communicate with each other by sending messages into sockets.
- How are these messages structured? What are the meanings of the various fields in the messages? When do the processes send the messages?

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

- In particular, an application-layer protocol defines:
  - 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

It is important to distinguish between network applications and application-layer protocols:

- An application-layer protocol is only one piece of a network application.

- Example: The Web is a client-server application that allows users to obtain documents from Web servers on demand. The Web application consists of many components including:

    - A standard for document formats (that is, HTML).
    - Webbrowsers (for example, Chrome and Microsoft Internet Explorer).
    - Web servers for example, Apache and Microsoft servers.
    - An application-layer protocol. The Web's application-layer protocol, HTTP, defines the format and sequence of messages exchanged between browser and Web server. Thus, HTTP is only one piece of the Web application.

The Web and HTTP

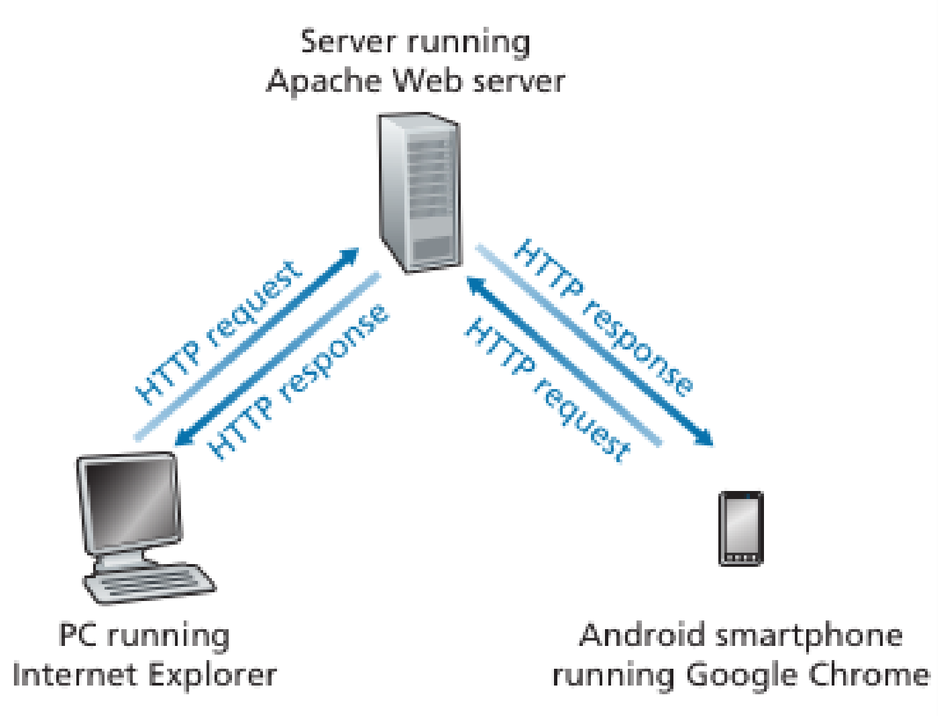
- The Web was the first Internet application that caught the general public's eye. It elevated the Internet from just one of many data networks to essentially the one and only data network.
- What appeals the most to users is that the Web operates on demand. Users receive what they want, when they want it. This is unlike traditional broadcast radio and television.
- It is enormously easy for any individual to make information available over the Web.
- The HyperText Transfer Protocol (HTTP), the Web's application-layer protocol, is at the heart of the Web.
- HTTP is implemented in two programs: a client program and a server program.
- The client program and server program, executing on different end systems, talk to each other by exchanging HTTP messages.
- HTTP defines the structure of these messages and how the client and server exchange the messages.

- A Web page consists of objects. An object is a file such as an HTML file, a JPEG image, a Javascrpt file, a CCS or a video clip that is addressable by a single URL.

- Most Web pages consist of a base HTML file and several referenced objects. Ex: A Web page contains HTML text and five JPEG images, then the Web page has six objects.

- The base HTML file references the other objects in the page with the objects' URLs.

- Each URL has two components:
  a. The hostname of the server that houses the object.
  b. And the object's path name.
     - Ex: The URL http://www.someSchool.edu/someDepartment/picture.gif has www.someSchool.edu for a hostname and /someDepartment/picture.gif for a path name.



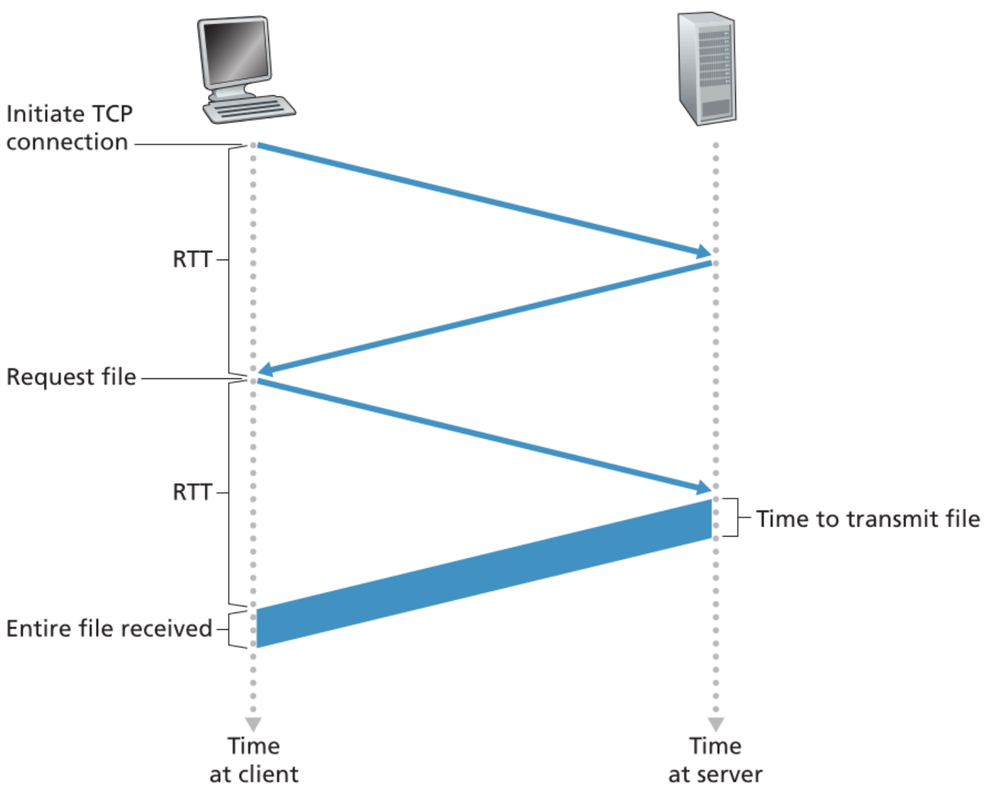
- Web browsers (such as Internet Explorer and Chrome)implement the client side of HTTP.
- Web servers implement the server side of HTTP, house Web objects, each addressable by a URL.
- HTTP defines how Web clients request Web pages from Web servers and how servers transfer Web pages to clients.
- HTTP uses TCP as its underlying transport protocol (rather than running on top of UDP).
- Because an HTTP server maintains no information about the clients, HTTP is said to be a stateless protocol.
- The Web uses the client-server application architecture.
- A Web server is always on, with a fixed IP address, and it services requests from potentially millions of different browsers.
- The original version of HTTP is called HTTP/1.0 and dates back to the early 1990’s. As of 2020, the majority of HTTP transactions take place over HTTP/1.1.

Non-Persistent and Persistent Connections

- non-persistent connections: Each request/response pair get sent over a separate TCP connection.
- persistent connections: All of the requests and their corresponding responses get sent over the same TCP connection.
- HTTP can use both non-persistent and persistent connections.
- HTTP uses persistent connections in its default mode, but can be configured to use non-persistent connections instead.

HTTP with Non-Persistent Connections

- suppose the page consists of a base HTML file and 10 JPEG images.
- he URL for the base HTML file is: http://www.someSchool.edu/someDepartment/home.index
- The HTTP client process initiates a TCP connection to the server www.someSchool.edu on port number 80.
- Associated with the TCP connection, there will be a socket at the client and a socket at the server.
- HTTP client sends an HTTP request message to the server via its socket. The request message includes the path name /someDepartment/home.index.
- HTTP server process receives the request message via its socket, retrieves the object /someDepartment/home.index from its storage, encapsulates the object in an HTTP response message, and sends the response message to the client via its socket.



- The HTTP server process tells TCP to close the TCP connection.
- The HTTP client receives the response message.
- The TCP connection terminates.
- The message indicates that the encapsulated object is an HTML file. The client extracts the file from the response message, examines the HTML file, and finds references to the 10 JPEG objects.
- The first four steps are then repeated for each of the referenced JPEG objects.
- The steps above illustrate the use of non-persistent connections, where each TCP connection is closed after the server sends the object the connection does not persist for other objects.
- HTTP/1.0 employes non-persistent TCP connections.
- Each non-persistent TCP connection transports exactly one request message and one response message.
- In the above example, when a user requests the Web page, 11 TCP connections are generated.

- We define the round-trip time (RTT), as the time it takes for a small packet to travel from client to server and then back to the client.
- The RTT includes packet-propagation delays, packet queuing delays in intermediate routers and switches, and packet-processing delays.
- When a user clicks on a hyperlink. It causes the browser to initiate a TCP connection between the browser and the Web server;
- This involves a “three-way handshake”:
  - The client sends a small TCP segment to the server.
  - The server acknowledges and responds with a small TCP segment.
  - The client acknowledges back to the server.
- The first two parts of the three-way handshake take one RTT.
- The client sends the HTTP request message combined with the third part of the three-way handshake into the TCP connection.
- The server sends the HTML file into the TCP connection. This HTTPrequest/response takes another RTT.
- Roughly; the total response time is two RTTs plus the transmission time at the server.

HTTP with Persistent Connections

- With HTTP/1.1 persistent connections, the server leaves the TCP connection open after sending a response.
- Subsequent requests and responses between the same client and server can be sent over the same connection.
- In particular, an entire Web page (in the previous example, the base HTML file and the 10 images) can be sent over a single persistent TCP connection.
- Typically, the HTTP server closes a connection when it isn’t used for a certain time (a configurable timeout interval).

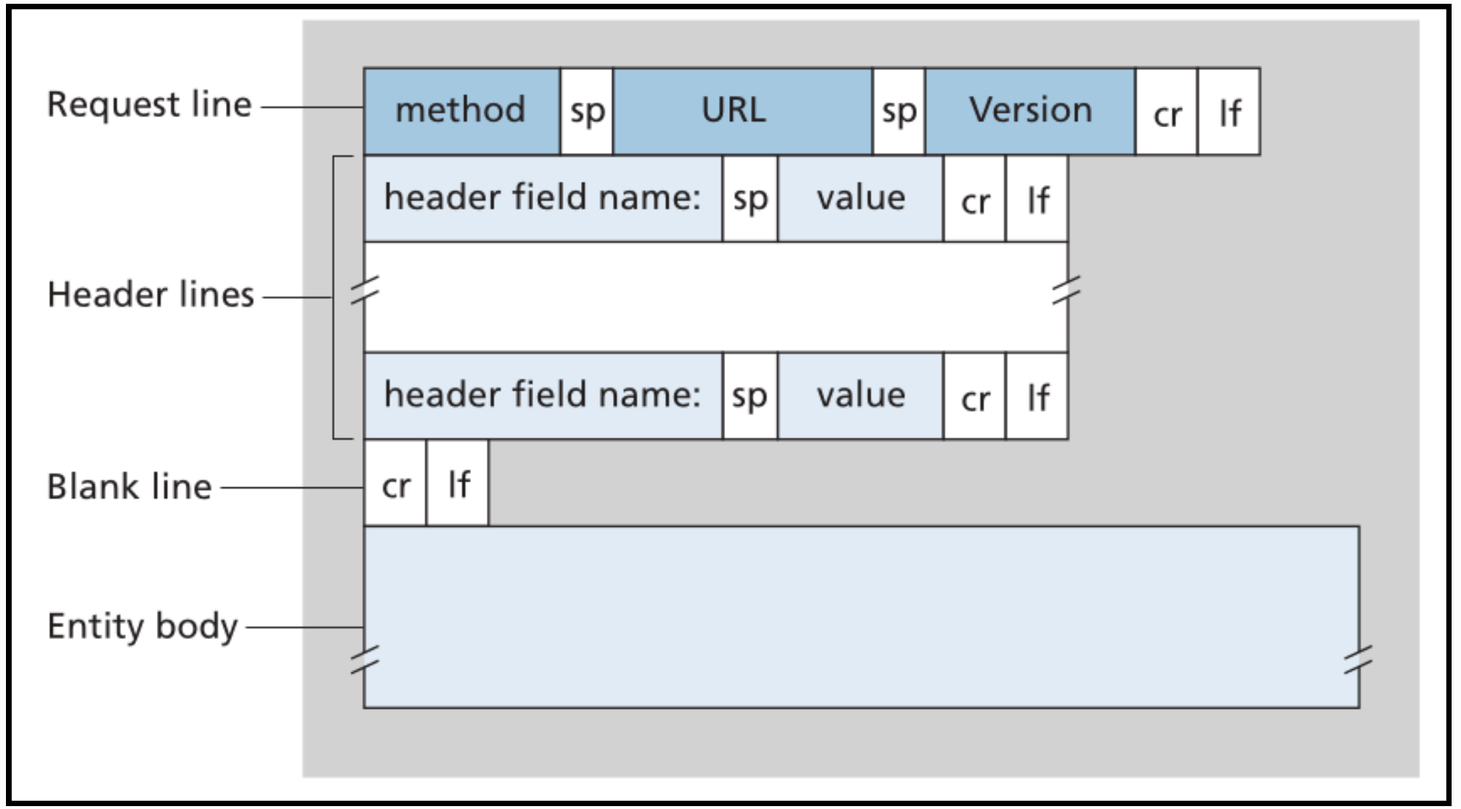
HTTP Message Format

- There are two types of HTTP messages, request messages and response messages.
- Below is a typical HTTP request message:
  - the message is written in ordinary ASCII text
  - the message consists of five lines, each followed by a carriage return and a line feed.
  - The last line is followed by an additional carriage return and line feed.
  - a request message can have many more lines or as few as one line.
  - The first line of an HTTP request message is the request line.
  - the subsequent lines are called the header lines.

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

- The request line has three fields:
  - the method field.
  - the URL field.
  - the HTTP version field.
- The method field can take on several different values, including:
  - GET.
  - POST.
  - HEAD.
  - PUT.
  - DELETE.

- The majority of HTTP request messages use the GET method.
- The GET method is used when the browser requests an object, with the requested object identified in the URL field.



- 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.
- With a POST message, the user is still requesting a Web page from the server, but the specific contents of the Web page depend on what the user entered into the form fields.
- if the value of the method field is POST, then the entity body contains what the user entered into the form fields.
- The HEAD method is similar to GET. When a server receives a request with HEAD method, it responds with an HTTP message but it leaves out the requested object.
- PUT method allows a user to upload an object to a specific path (directory) on a specific Web server.
- DELETE method allows a user, or an application, to delete an object on a Web server.

HTTP Response Message

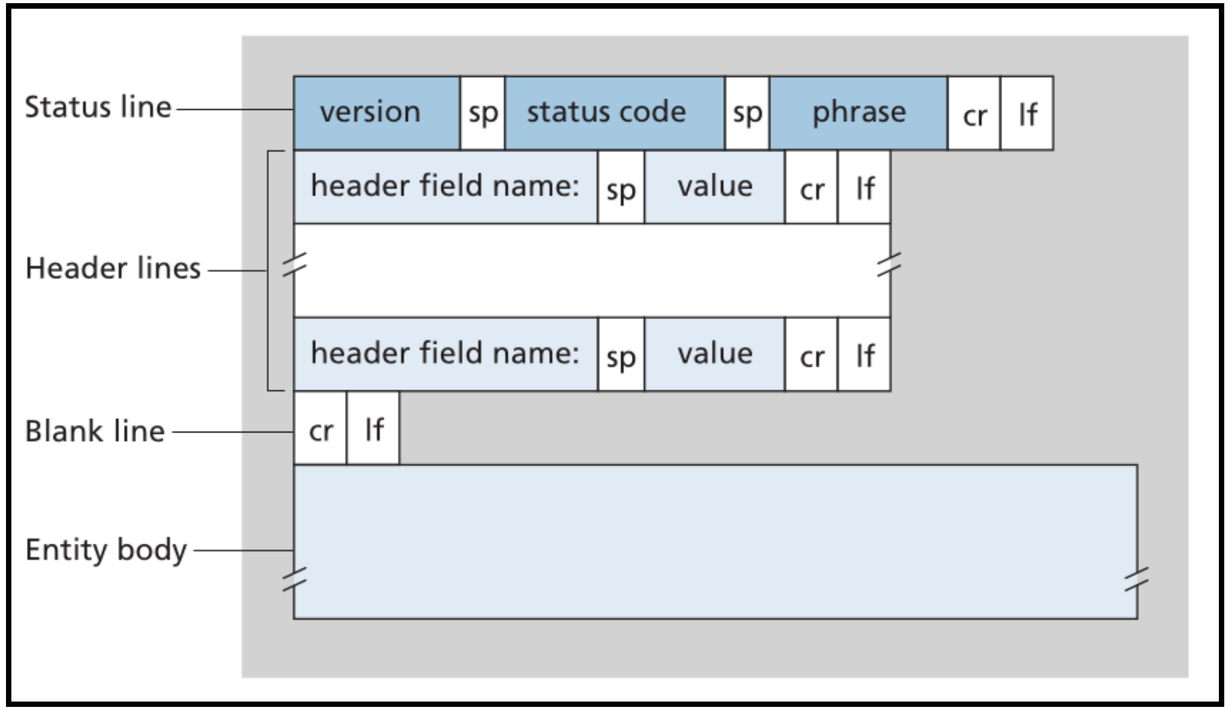
- the example response message has three sections: an initial status line, six header lines, and then the entity body.
- The entity body contains the requested object itself (represented by data data data data ...).
- The status line has three fields: the protocol version field, a status code, and a corresponding status message.
- In this example, the status line indicates that the server is using HTTP/1.1 and that everything is OK (that is, the server has found, and is sending, the requested object).

```
HTTP/1.1 200 OK
Connection: close
Date: Tue, 18 Aug 2015 15:44:04 GMT
Server: Apache/2.2.3 (CentOS)
Last-Modified: Tue, 18 Aug 2015 15:11:03 GMT
Content-Length: 6821
Content-Type: text/html

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

- The server uses the Connection: close header line to tell the client that it is going to close the TCP connection after sending the message.
- The Date: header line indicates the time and date when the HTTP response was created and sent by the server.
- The Server: header line indicates that the message was generated by an Apache Web server.
- Last-Modified: header line indicates the time and date when the object was created or last modified.
- Content-Length: header line indicates the number of bytes in the object being sent.
- Content-Type: header line indicates that the object in the entity body is HTML text.

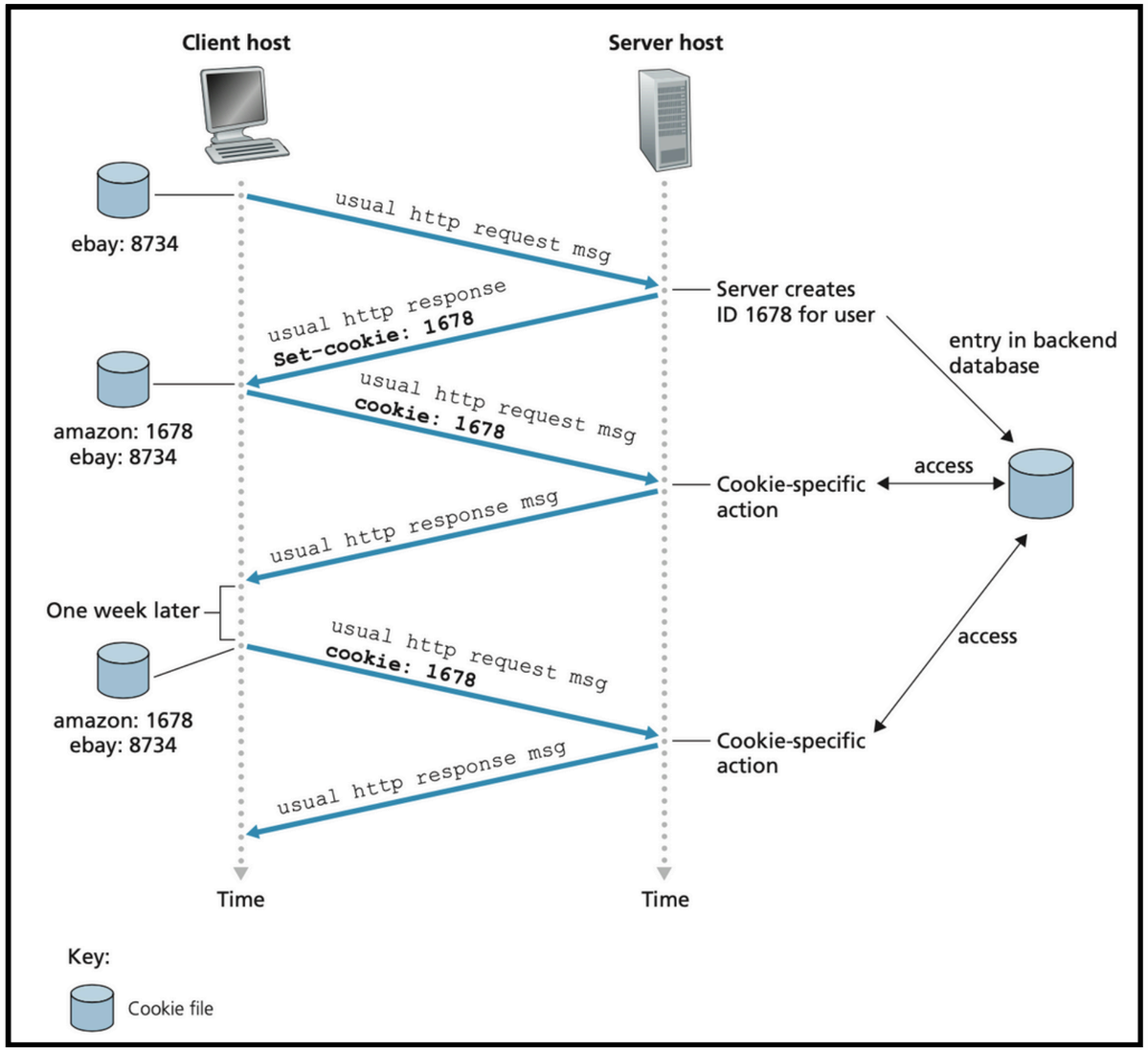
- The status code and associated phrase indicate the result of the request. Some common status codes and associated phrases include:
  - 200 OK: Request succeeded and the information is returned in the response.
  - 301 Moved Permanently: Requested object has been permanently moved.
  - 400 Bad Request: The request could not be understood by the server.
  - 404 Not Found: The requested document does not exist on this server.
  - 505 HTTP Version Not Supported: The requested HTTP protocol version is not supported by the server.



User-Server Interaction: Cookies:

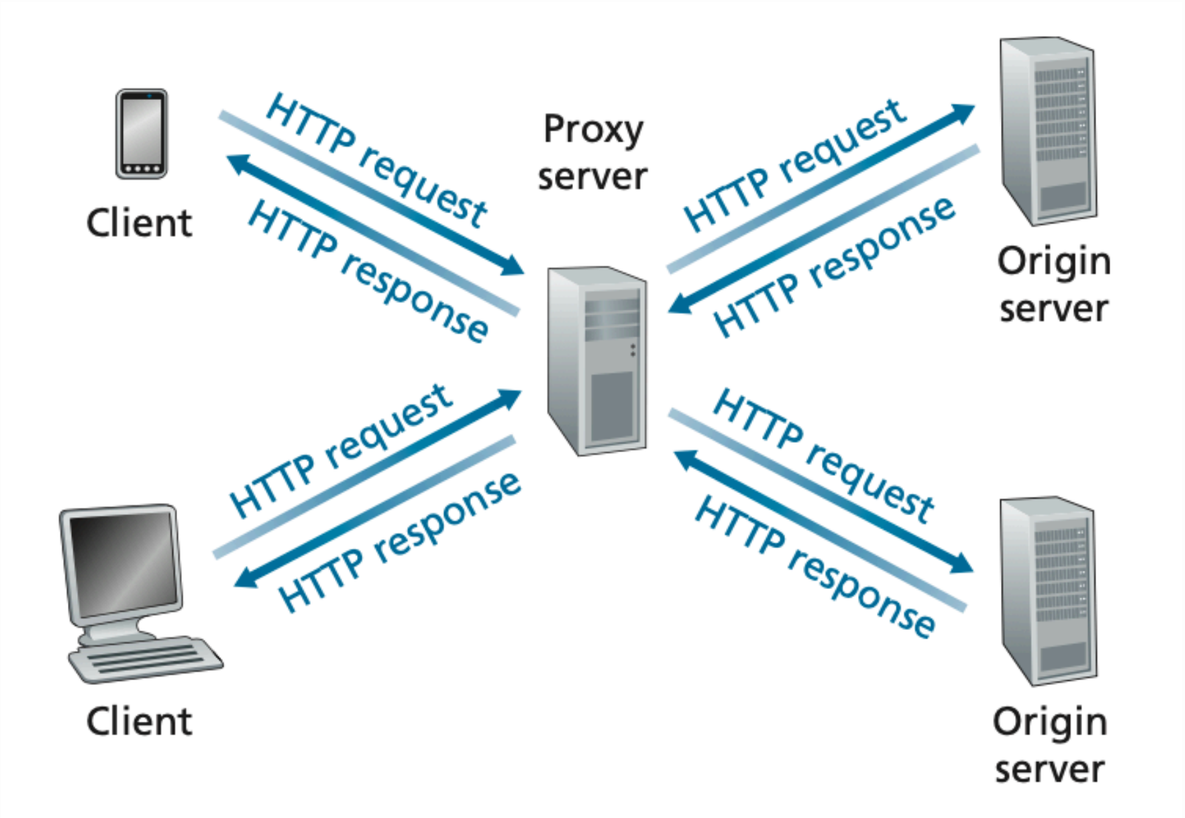
- HTTP server is stateless. This simplifies server design and has permitted engineers to develop high-performance Web servers that can handle thousands of simultaneous TCP connections.
- it is often desirable for a Web site to identify users, either because the server wishes to restrict user access or because it wants to serve content as a function of the user identity. For these purposes, HTTP uses cookies.
- Cookies, defined in [RFC 6265], allow sites to keep track of users.
- Cookie technology 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.

- When the request comes into the Web server, the server creates a unique identification number and creates an entry in its back-end database that is indexed by the identification number.
- The Web server then responds to browser, including in the HTTP response a Set-cookie: header, which contains the identification number. For example, the header line might be: Set-cookie: 1678
- When the browser receives the HTTP response message, it sees the Set-cookie: header. The browser then appends a line to the special cookie file that it manages. This line includes the hostname of the server and the identification number in the Set-cookie: header.
- each time the browser requests a Web page, it consults the cookie file, extracts the identification number for this site, and puts a cookie header line that includes the identification number in the HTTP request. in the example, each of the HTTP requests to the server includes the header line: Cookie: 1678



Web Caching:

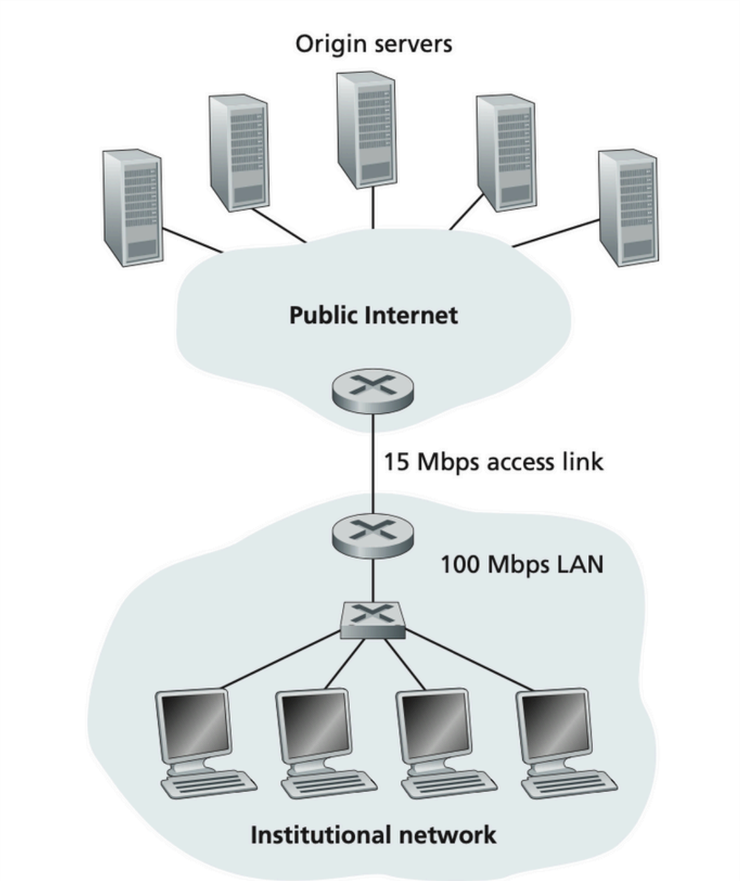
- A Web cache (also called a proxy server) is a network entity that satisfies HTTP requests on the behalf of an origin Web server.
- A web cache has its own disk storage and keeps copies of recently requested objects in this storage.
- A user's browser can be configured so that all of the user's HTTP requests are first directed to the Web cache. Once a browser is configured, each browser request for an object is first directed to the Web cache.

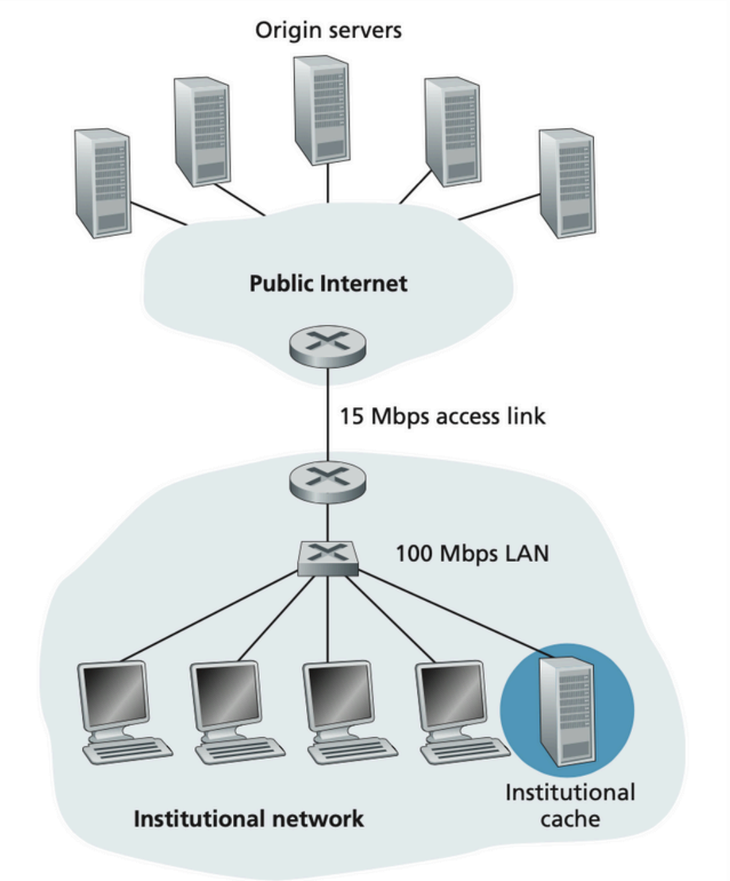


1. The browser establishes a TCP connection to the Web cache and sends an HTTP request for the object to the Web cache.
2. The Web cache checks to see if it has a copy of the object stored locally. If it does, the Web cache returns the object within an HTTP response message to the client browser.
3. If the Web cache does not have the object, the Web cache opens a TCP connection to the origin server. The Web cache then sends an HTTP request for the object into the cache-to-server TCP connection. After receiving this request, the origin server sends the object within an HTTP response to the Web cache.
4. When the Web cache receives the object, it stores a copy in its local storage and sends a copy, within an HTTP response message, to the client browser (over the existing TCP connection between the client browser and the Web cache).

- a cache is both a server and a client at the same time. When it receives requests from and sends responses to a browser, it is a server. When it sends requests to and receives responses from an origin server, it is a client.
- Typically a Web cache is purchased and installed by an ISP. For example, a university might install a cache on its campus network and configure all of the campus browsers to point to the cache.

- Web caching has seen deployment in the Internet for two reasons.
  a. First, a Web cache can substantially reduce the response time for a client request, particularly if the bottleneck bandwidth between the client and the origin server is much less than the bottleneck bandwidth between the client and the cache.
  b. Web caches can reduce traffic on an institution's access link to the Internet. By reducing traffic, the institution does not have to upgrade bandwidth as quickly, thereby reducing costs.





The Conditional GET

- the object housed in the Web server may have been modified since the copy was cached at the client.
- HTTP has a mechanism that allows a cache to verify that its objects are up to date. This mechanism is called the conditional GET.
- An HTTP request message is called conditional GET message if
  - The request message uses the GET method
  - The request message includes an If-Modified-Since: header line.
- Example:
  - on the behalf of a requesting browser, a proxy cache sends a request message to a Web server:

```
GET /fruit/kiwi.gif HTTP/1.1
Host: www.exotiquecuisine.com
```

- the Web server sends a response message with the requested object to the cache:

```
HTTP/1.1 200 OK
Date: Sat, 3 Oct 2015 15:39:29
Server: Apache/1.3.0 (Unix)
Last-Modified: Wed, 9 Sep 2015 09:23:24
Content-Type: image/gif

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

- The cache forwards the object to the requesting browser and also caches the object locally.The cache also stores the last-modified date along with the object.
- One week later, another browser requests the same object via the cache, and the object is still in the cache. Since this object may have been modified at the Web server in the past week, the cache performs an up-to-date check by issuing a conditional GET. Specifically, the cache sends:

```
GET /fruit/kiwi.gif HTTP/1.1
Host: www.exotiquecuisine.com
If-modified-since: Wed, 9 Sep 2015 09:23:24
```

- The value of the If-modified-since: header line is exactly equal to the value of the Last-Modified: header line that was sent by the server one week ago.
- This conditional GET tels the server to send the object only if the object has been modified since the specified date.
- Suppose the object has not been modified since 9 Sep 2015 09:23:24. Then, the Web server sends a response message to the cache:

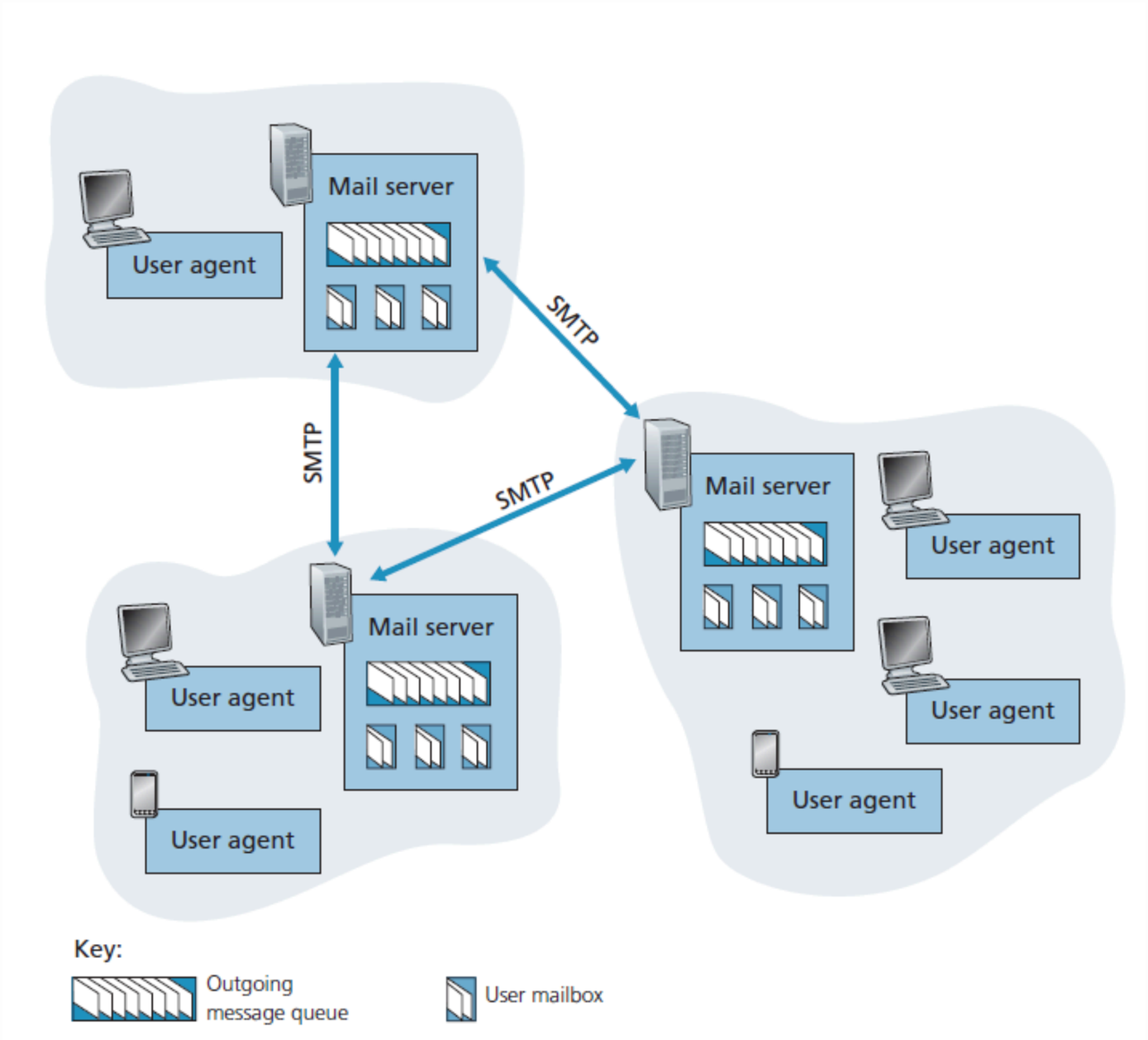
```
HTTP/1.1 304 Not Modified
Date: Sat, 10 Oct 2015 15:39:29
Server: Apache/1.3.0 (Unix)

(empty entity body)
```

- The Web server still sends a response message but does not include the requested object in the response message.
- Including the requested object would only waste bandwidth and increase userperceived response time.
- The response message has 304 Not Modified in the status line. This tells the cache that it can go ahead and forward its cached copy of the object to the requesting browser.

Electronic Mail in the Internet

- Electronic mail has been around since the beginning of the Internet. And has become more elaborate and powerful over the years.
- E-mail is an asynchronous communication medium, people send and read messages when it is convenient for them, without having to coordinate with other people's schedules.
- We see from the diagram that E-mail has three major components:
  - User agents.M
  - Mail servers.
  - And the Simple Mail Transfer Protocol (SMTP).

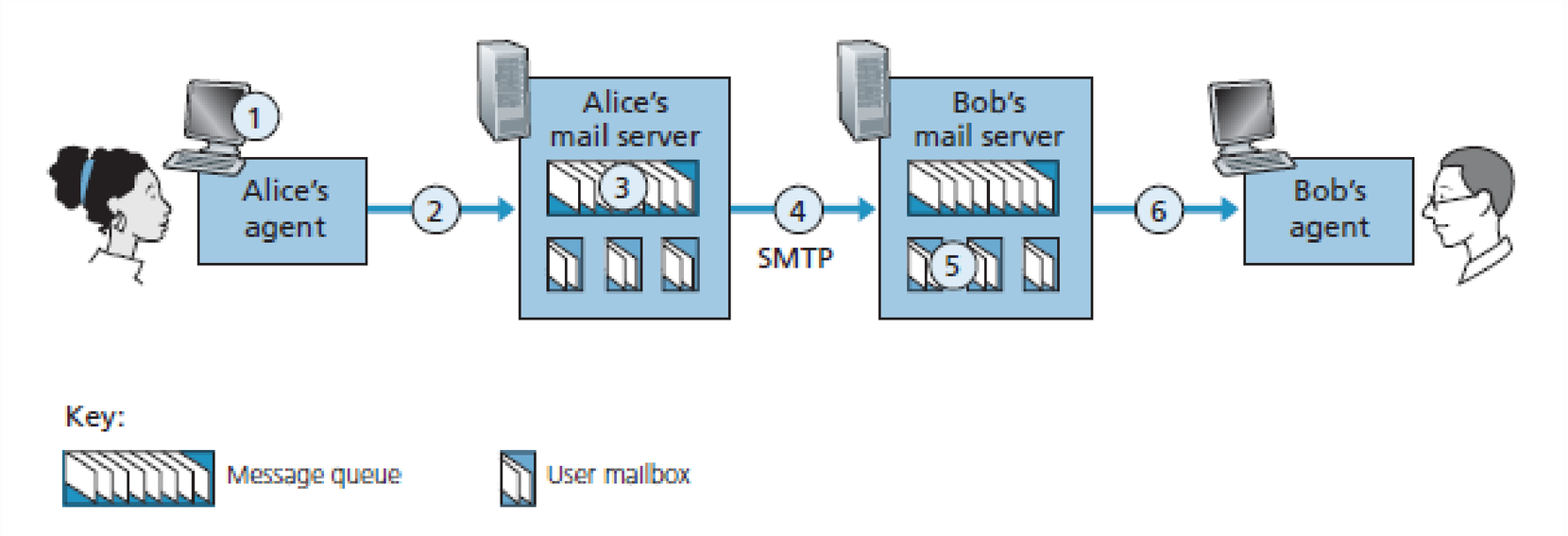


- The sender sends an e-mail message to a recipient.
- User agents allow users to read, reply to, forward, save, and compose messages. Examples of user agents for e-mail include Microsoft Outlook, Apple Mail, Webbased Gmail, the Gmail App running in a smartphone, and so on.
- When the sender is finished composing her message, her user agent sends the message to her mail server, where the message is placed in the mail server's outgoing message queue.
- When the recipient wants to read a message, his user agent retrieves the message from his mailbox in his mail server.
- Mail servers form the core of the e-mail infrastructure.
- Each recipient, has a mailbox located in one of the mail servers.
- recipient's mailbox manages and maintains the messages that have been sent to him.
- A typical message starts its journey in the sender's user agent, then travels to the sender's mail server, and then travels to the recipient's mail server, where it is deposited in the recipient's mailbox.
- When a recipient wants to access the messages in his mailbox, the mail server containing his mailbox authenticates him(with his username and password).
- Sender's mail server must also deal with failures in recipient's mail server. If sender's server cannot deliver mail to recipient's server, sender's server holds the message in a message queue and attempts to transfer the message later.
- If there is no success after several days, the server removes the message and notifies the sender with an e-mail message.

SMTP

- SMTP is at the heart of Internet electronic mail.
- SMTP transfers messages from senders' mail servers to the recipients' mail servers
- SMTP is much older than HTTP.
- It restricts the body (not just the headers) of all mail messages to simple 7-bit ASCII.
- It requires binary multimedia data to be encoded to ASCII before being sent over SMTP; and it requires the corresponding ASCII message to be decoded back to binary after SMTP transport. Recall that HTTP does not require multimedia data to be ASCII encoded before transfer.
- The client SMTP (running on the sending mail server host) has TCP establish a connection to port 25 at the server SMTP (running on the receiving mail server host).
- If the server is down, the client tries again later. Once this connection is established, the server and client perform some applicationlayer handshaking.
- During the SMTP handshaking phase, the SMTP client indicates the e-mail address of the sender (the person who generated the message) and the e-mail address of the recipient. Once the SMTP client and server have introduced themselves to each other, the client sends the message.
- SMTP can count on the reliable data transfer service of TCP to get the message to the server without errors.

- Example in the Figure:
  - 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.
  - 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 messag into the TCP connection.
  - 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.



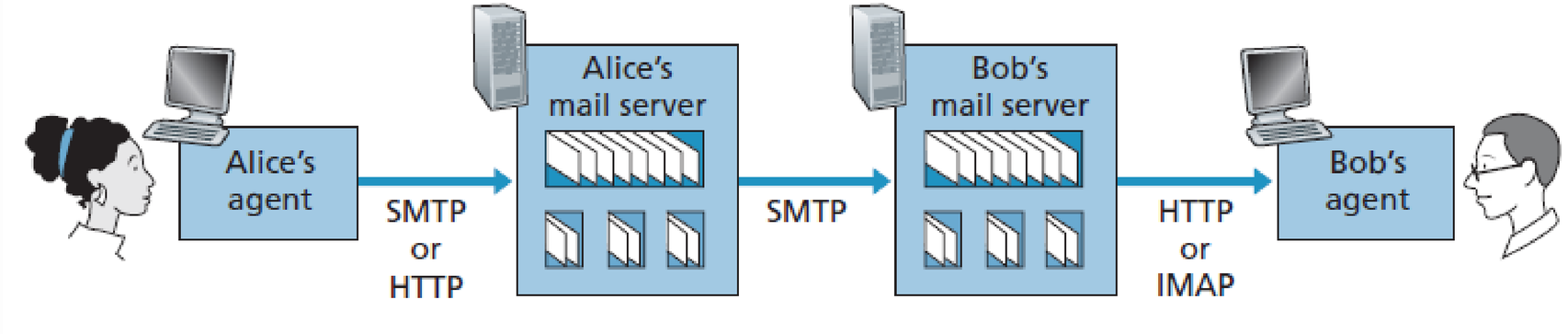
example transcript of messages exchanged between an SMTP client (C) and an SMTP server (S):

- The hostname of the client is crepes.fr and the hostname of the server is hamburger.edu.
- C: are exactly the lines the client sends into its TCP socket, and S: the lines the server sends into its TCP socket.
- The transcript on the right begins as soon as the TCP connection is established.
- the client sends a message (“Do you like ketchup? How about pickles?”) from mail server crepes.fr to mail server hamburger.edu.
- The client sends a line consisting of a single period, which indicates the end of the message to the server.
- The server issues replies to each command, with each reply having a reply code and some (optional) English-language explanation.
- SMTP uses persistent connections: If the sending mail server has several messages to send to the same receiving mail server, it can send all of the messages over the same TCP connection.

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

Mail Access Protocols

- Once SMTP delivers the message from sender's mail server to recipient's mail server, the message is placed in Bob's mailbox.
- In the example:
  - Alice's user agent uses SMTP or HTTP to deliver the e-mail message into her mail server.
  - Alice's mail server uses SMTP (as an SMTP client) to relay the e-mail message to Bob's mail server.
  - By having Alice first deposit the e-mail in her own mail server, Alice's mail server can repeatedly try to send the message to Bob's mail server, say every 30 minutes, until Bob's mail server becomes operational.
  - Bob's user agent can't use SMTP to obtain the messages because obtaining the messages is a pull operation, whereas SMTP is a push protocol.



- Today, there are two common ways for Bob to retrieve his e-mail from a mail server:

    - If Bob is using Web-based e-mail or a smartphone app (such as Gmail), then the user agent will use HTTP to retrieve Bob's e-mail. This case requires Bob's mail server to have an HTTP interface as well as an SMTP interface (to communicate with Alice's mail server).
    - The alternative method, typically used with mail clients such as Microsoft Outlook, is to use the Internet Mail Access Protocol (IMAP).

- Both the HTTP and IMAP approaches allow Bob to manage folders, maintained in Bob's mail server. Bob can move messages into the folders he creates, delete messages, mark messages as important, and so on.

DNS—The Internet's Directory Service

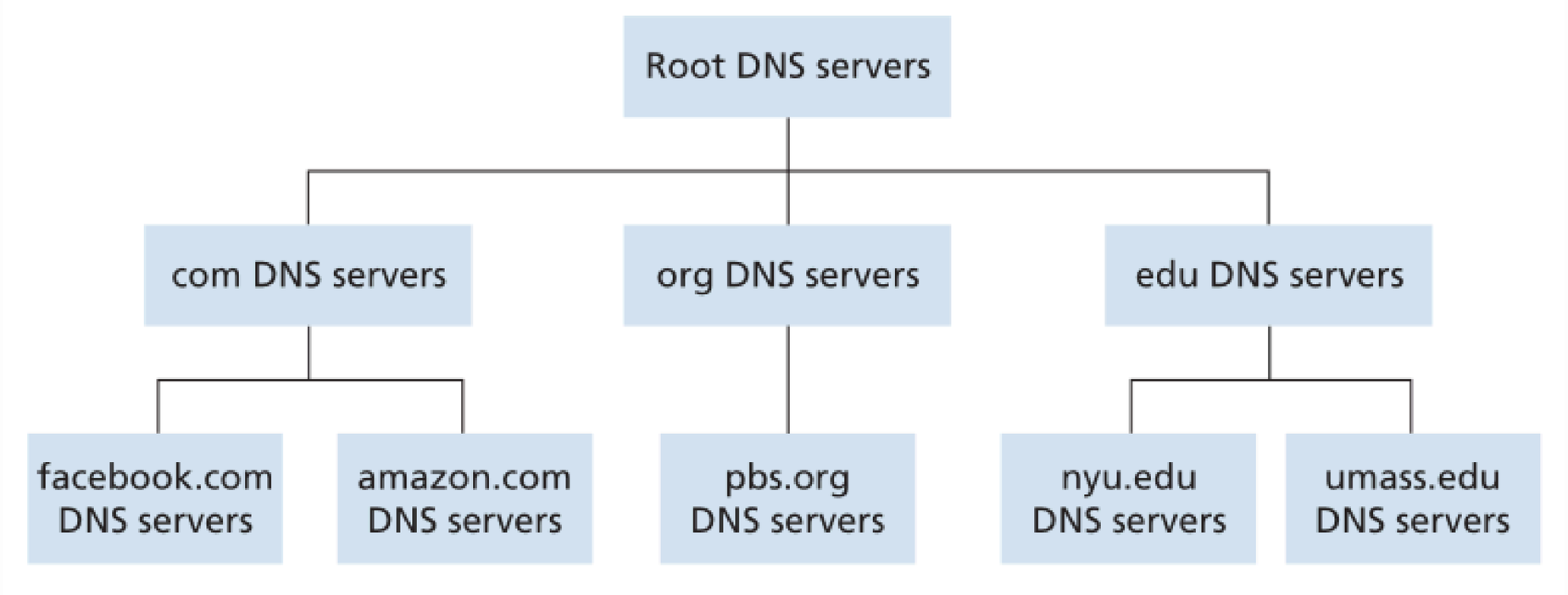
- One identifier for a host is its hostname. Hostnames—such as www.facebook.com, www.google.com, gaia.cs.umass.edu—are mnemonic and are therefore appreciated by humans. However, hostnames provide little, if any, information about the location within the Internet of the host.
- Because hostnames can consist of variable-length alphanumeric characters, they would be difficult to process by routers. For these reasons, hosts are also identified by so-called IP addresses.
- An IP (Version4) address consists of four bytes and has a rigid hierarchical structure. An IP address looks like 121.7.106.83, where each period separates one of the bytes expressed in decimal notation from 0 to 255. An IP address is hierarchical because as we scan the address from left to right, we obtain more and more specific information about where the host is located in the Internet (that is, within which network, in the network of networks).
- We need a directory service that translates hostnames to IP addresses. This is the main task of the Internet's domain name system (DNS).
- The DNS is :
  - A distributed database implemented in a hierarchy of DNS servers.
  - And an application-layer protocol that allows hosts to query the distributed database.
- DNS is commonly employed by other application-layer protocols, including HTTP and SMTP, to translate user-supplied hostnames to IP addresses.

How DNS Works?

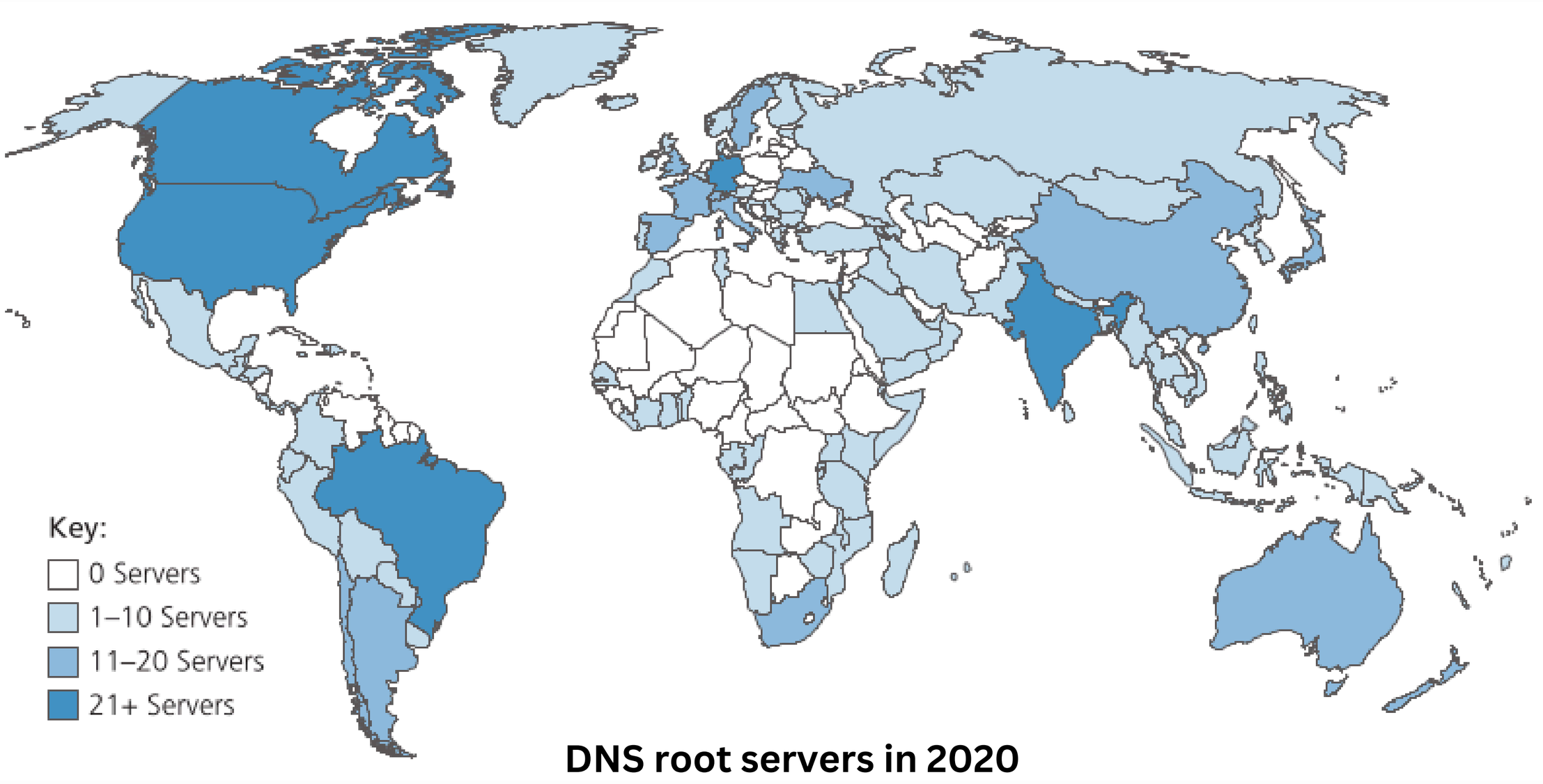
- Suppose that some application (such as a Web browser or a mail client) running in a user's host needs to translate a hostname to an IP address:
  - The application will invoke the client side of DNS, specifying the hostname that needs to be translated.
  - DNS in the user's host then takes over, sending a query message into the network. All DNS query and reply messages are sent within UDP datagrams to port 53.
  - After a delay, ranging from milliseconds to seconds, DNS in the user's host receives a DNS reply message that provides the desired mapping.
  - This mapping is then passed to the invoking application.
  - From the perspective of the invoking application in the user's host, DNS is a black box providing a simple, straightforward translation service.
  - In fact, the black box that implements the service is complex, consisting of a large number of DNS servers distributed around the globe, as well as an application-layer protocol that specifies how the DNS servers and querying hosts communicate.

A Distributed, Hierarchical Database

- DNS uses a large number of servers, organized in a hierarchical fashion and distributed around the world.
- No single DNS server has all of the mappings for all of the hosts in the Internet. Instead, the mappings are distributed across the DNS servers.
- There are three classes of DNS servers, root DNS servers, top-level domain (TLD) DNS servers, and authoritative DNS servers, organized in a hierarchy as shown in the figure.



- Example: suppose a DNS client wants to determine the IP address for the hostname www.amazon.com
  - The client first contacts one of the root servers, which returns IP addresses for TLD servers for the top-level domain .com
  - The client then contacts one of these TLD servers, which returns the IP address of an authoritative server for amazon.com
  - Finally, the client contacts one of the authoritative servers for amazon.com, which returns the IP address for the hostname www.amazon.com
- Root DNS servers: There are more than 1000 root servers instances scattered allover the world. These root servers are copies of 13 different root servers, managed by 12 different organizations. Root name servers provide the IP addresses of the TLD servers.
- Top-level domain (TLD) servers: For each of the top-level domains (top-level domains such as com, org, net, edu, and gov) and all of the country top-level domains such as (uk, fr, ca, and jp) there is TLD server (or server cluster). TLD servers provide the IP addresses for authoritative DNS servers.
- Authoritative DNS servers: Every organization with publicly accessible hosts (such as Web servers and mail servers) on the Internet must provide publicly accessible DNS records that map the names of those hosts to IP addresses. An organization's authoritative DNS server houses these DNS records. An organization can choose to implement its own authoritative DNS server to hold these records; alternatively, the organization can pay to have these records stored in an authoritative DNS server of some service provider. Most universities and large companies implement and maintain their own primary and secondary (backup) authoritative DNS server.



DNS root servers in 2020

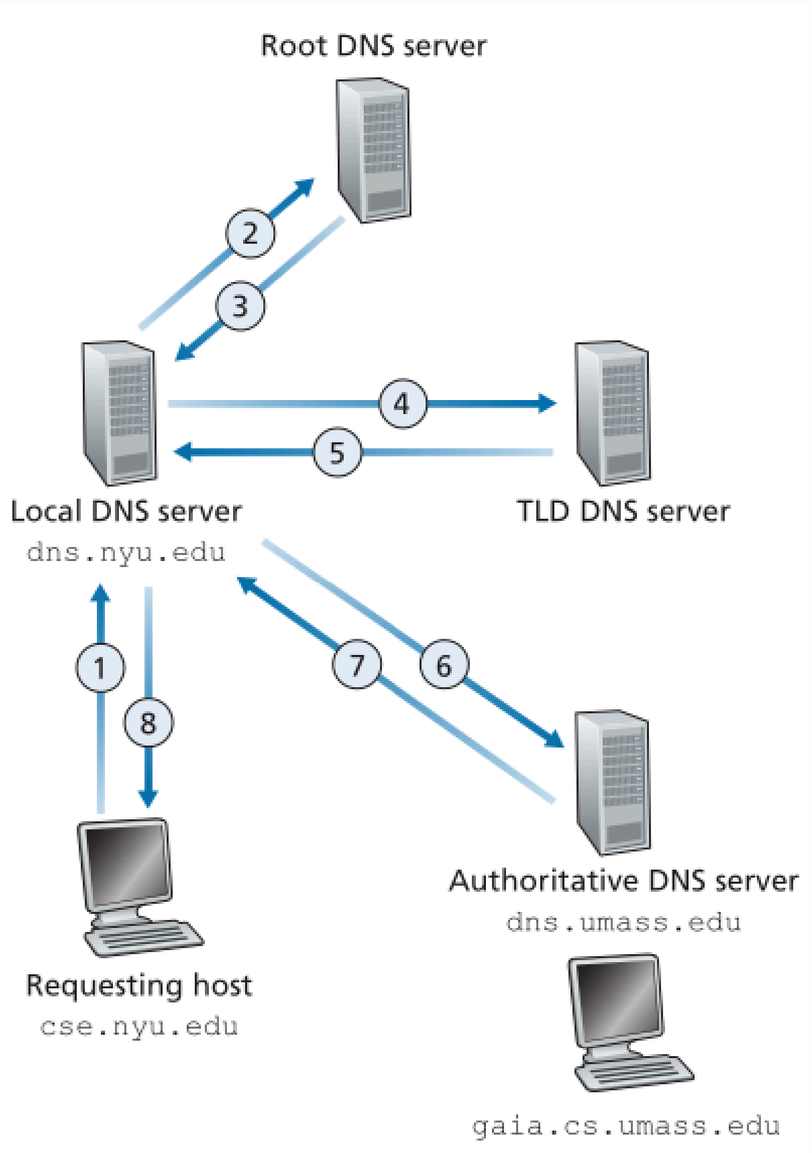
local DNS servers

- A local DNS server does not strictly belong to the hierarchy of servers but is nevertheless central to the DNS architecture.
- Each ISP such as a residential ISP or an institutional ISP has a local DNS server (default name server).
- When a host connects to an ISP, the ISP provides the host with the IP addresses of one or more of its local DNS servers.
- A host’s local DNS server is typically “close to” the host. For an institutional ISP, the local DNS server may be on the same LAN as the host; for a residential ISP, it is typically separated from the host by no more than a few routers.
- When a host makes a DNS query, the query is sent to the local DNS server, which acts a proxy, forwarding the query into the DNS server hierarchy.

recursive queries and iterative queries

Suppose the host cse.nyu.edu desiresthe IP address of gaia.cs.umass.edu. Also suppose that NYU's local DNS server for cse.nyu.edu is called dns.nyu.edu and that an authoritative DNS server for gaia.cs.umass.edu is called dns.umass.edu.

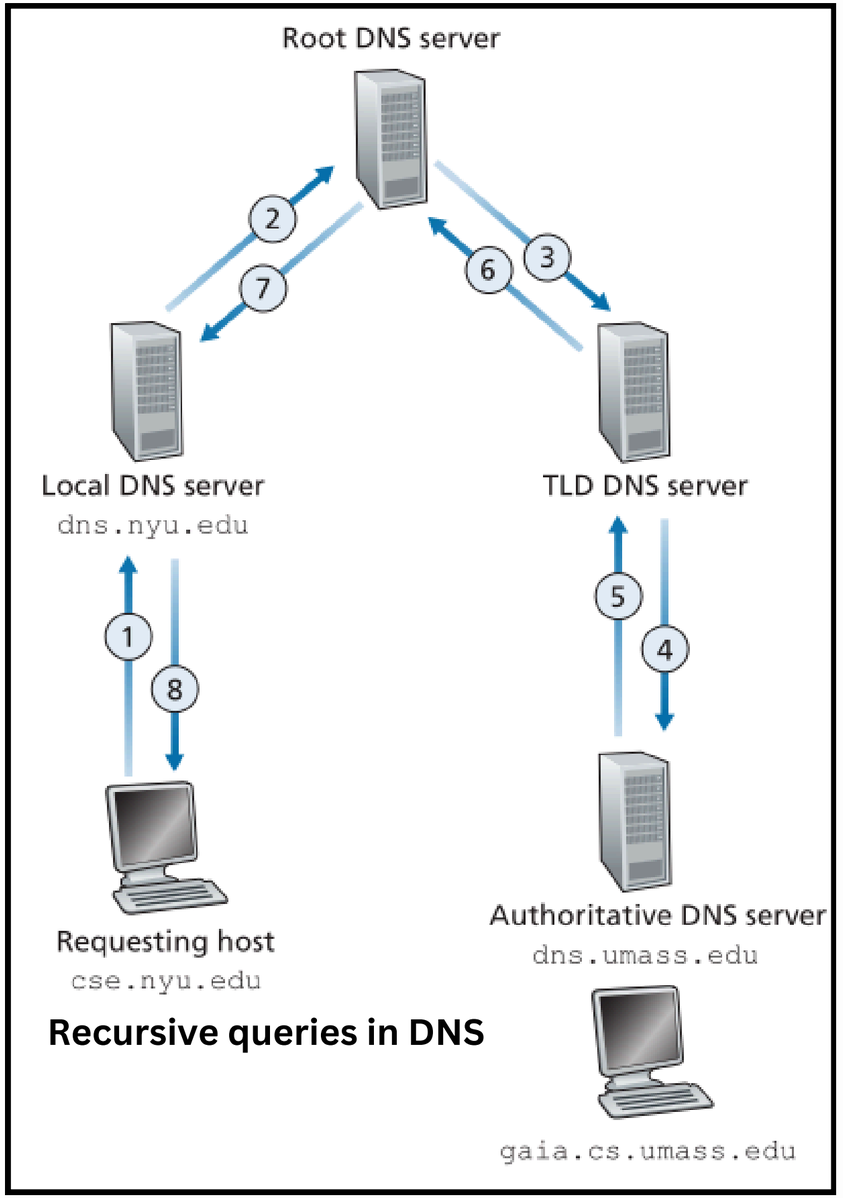
- The host cse.nyu.edu first sends a DNS query message to its local DNS server, dns.nyu.edu. The query message contains the hostname to be translated, namely, gaia.cs.umass.edu
- The local DNS server forwards the query message to a root DNS server. The root DNS server takes note of the edu suffix and returns to the local DNS server a list of IP addresses for TLD servers responsible for edu
- The local DNS server then resends the query message to one of these TLD servers. The TLD server takes note of the umass.edu suffix and responds with the IP address of the authoritative DNS server for the University of Massachusetts,namely, dns.umass.edu.
- Finally, the local DNS server resends the query message directly to dns.umass.edu, which responds with the IP address of gaia.cs.umass.edu



DNS Caching

DNS extensively exploits DNS caching in order to improve the delay performance and to reduce the number of DNS messages ricocheting around the Internet.

- in a query chain, when a DNS server receives a DNS reply, it can cache the mapping in its local memory.
- In our previous example, each time the local DNS server dns.nyu.edu receives a reply from some DNS server, it can cache any of the information contained in the reply.
- 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 from its cache.
- DNS servers discard cached information after a period of time (often set to two days).
- A local DNS server can also cache the IP addresses of TLD servers, thereby allowing the local DNS server to bypass the root DNS servers in a query chain. In fact, because of caching, root servers are bypassed for all but a very small fraction of DNS queries.

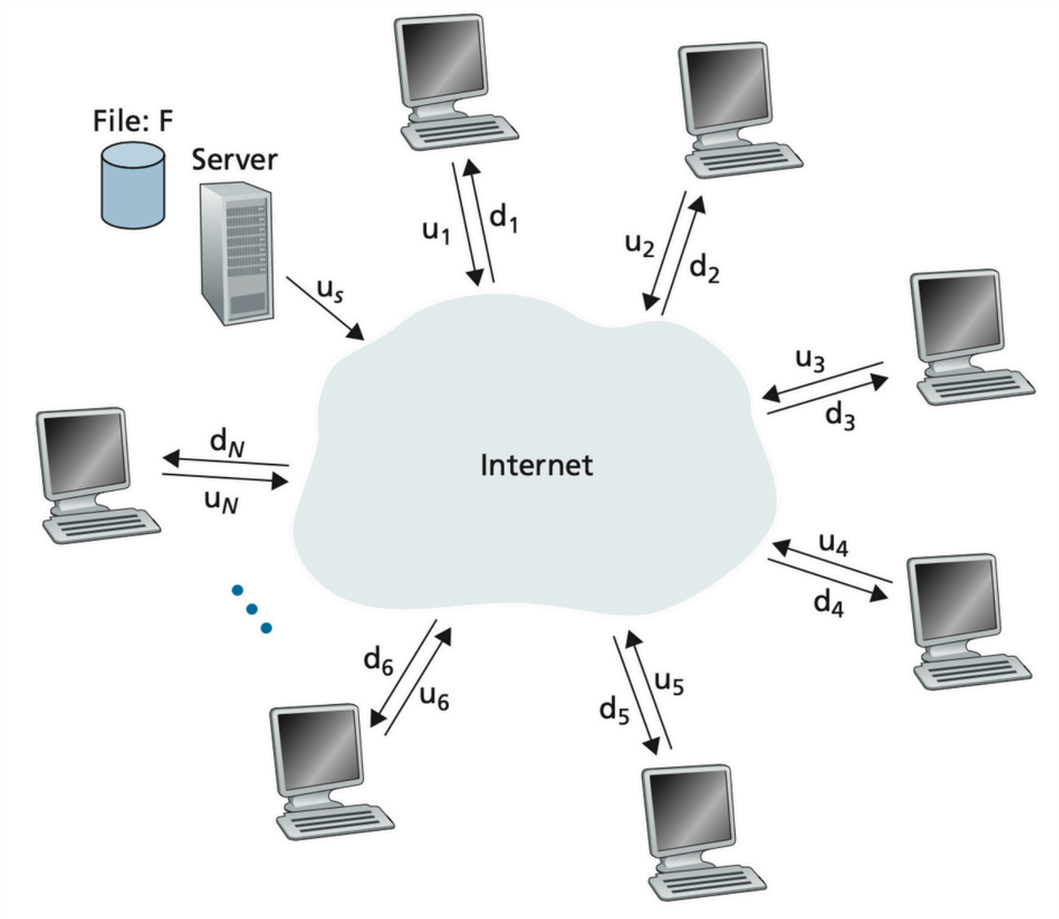


Recursive queries in DNS

Scalability of P2P Architectures

To compare client-server architectures with peer-to-peer architectures, and illustrate the inherent self-scalability of P2P, we now consider a simple quantitative model for distributing a file to a fixed set of peers for both architecture types.

- Denote the upload rate of the server's access link by *us*, the upload rate of the *i*th peer's access link by *ui*, and the download rate of the *i*th peer's access link by *di,* the size of the file to be distributed (in bits) by *F* and the number of peers that want to obtain a copy of the file by *N*.
- The distribution time is the time it takes to get a copy of the file to all N peers.



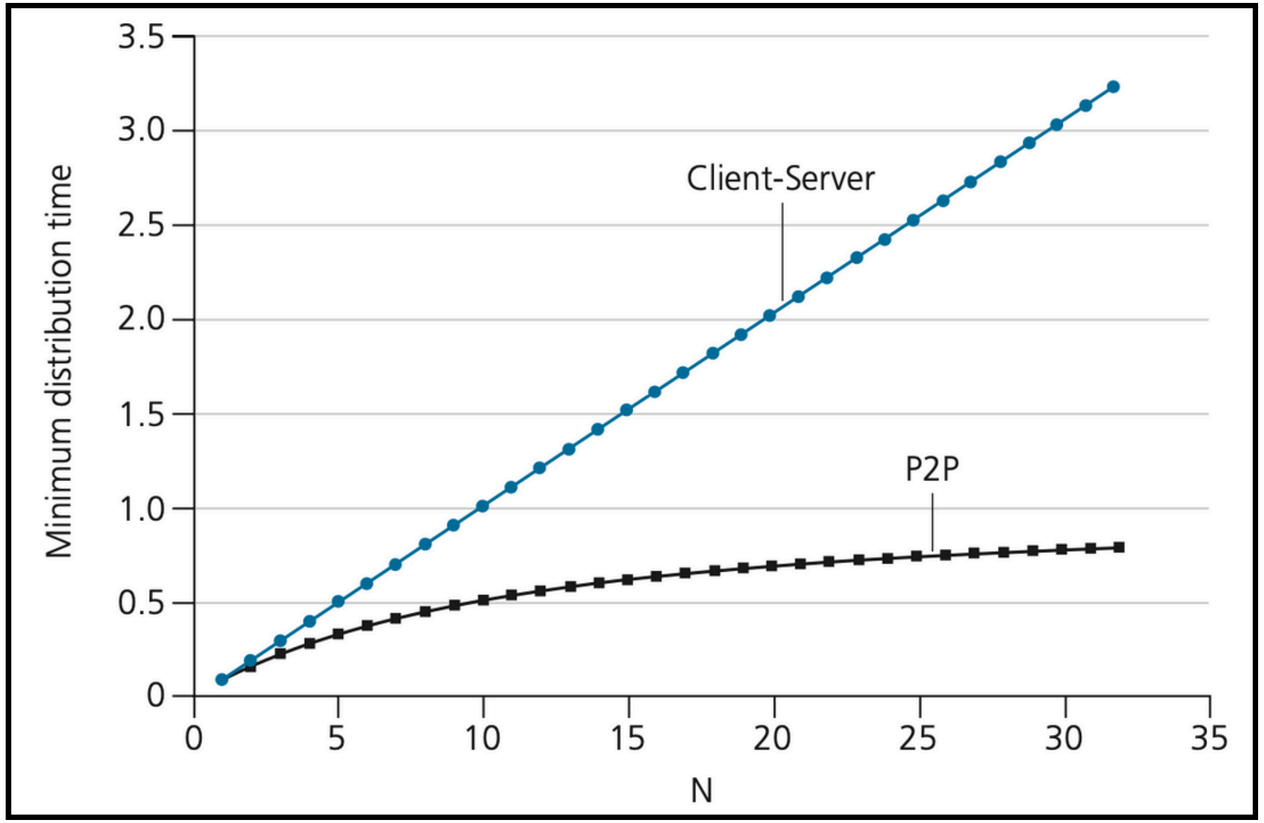
we make the simplifying assumption that the Internet core has abundant bandwidth, implying that all of the bottlenecks are in access networks.

- Let's first determine the distribution time for the client-server architecture, which we denote by *Dcs*
  - The server must transmit one copy of the file to each of the *N* peers. Thus, the server must transmit *NF* bits. The time to distribute the file must be at least *NF*/*us*.
  - Let *d*min denote the download rate of the peer with the lowest download rate, that is, *dmin* = min{*d1* , *d2* , . . . , *d N}*. The peer with the lowest download rate cannot obtain all *F* bits of the file in less than *F*/*d*min seconds. Thus, the minimum distribution time is at least *F*/*d*min.

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

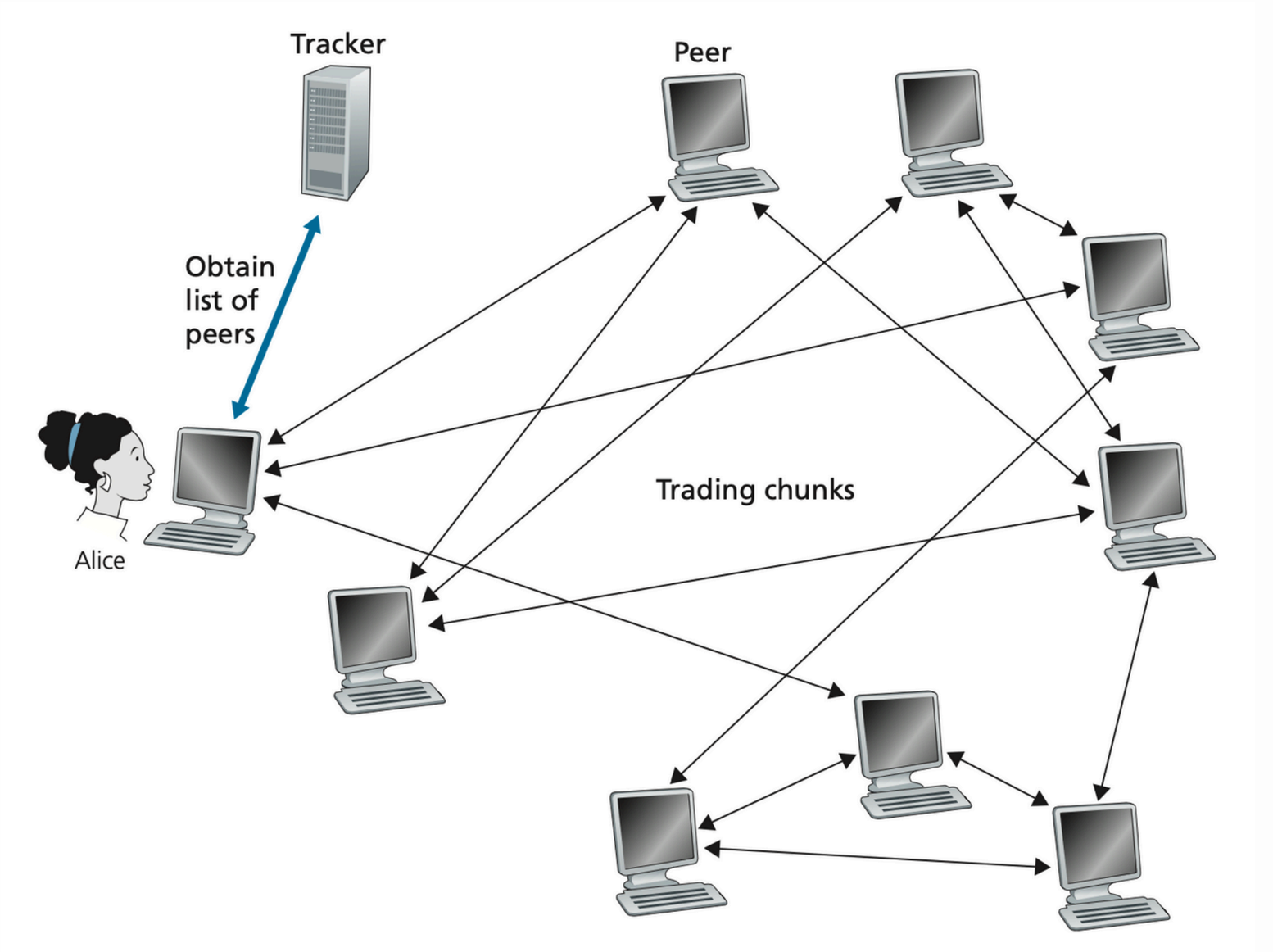
Let’s now go through a similar analysis for the P2P architecture, where each peer can assist the server in distributing the file:

- At the beginning of the distribution, only the server has the file. To get this file into the community of peers, the server must send each bit of the file at least once into its access link. Thus, the minimum distribution time is at least F/us.
- The peer with the lowest download rate cannot obtain all *F* bits of the file in less than *F*/*d*min seconds.
- *u*total = *us* + *u*1 +…+ *uN*. The system must deliver (upload) *F* bits to each of the *N* peers, thus delivering a total of *NF* bits. This cannot be done at a rate faster than *u*total. Thus, the minimum distribution time is also at least *NF*/(*us* + *u*1 + …+ *uN*).



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

BitTorrent



Summery:

- We’ve learned about the ubiquitous client-server architecture adopted by many Internet applications and seen its use in the :
  - HTTP
  - SMTP, IMAP
  - And DNS protocols.
- We’ve learned about Caching and Cookies.
- We’ve learned about the P2P architecture and contrasted it with the client-server architecture.
- The material in this chapter, and in particular our detailed study of the HTTP, SMTP, and DNS protocols, has now added considerable substance to this fact:
  - Protocols are a key concept in networking.
- We described the service models that TCP and UDP offer to applications that invoke them.