CSC 4350 – Computer Networks

Fall 2024 Semester

Lecture 7 – Cookies, Caching, Speed Enhancements,

Intro to SMTP

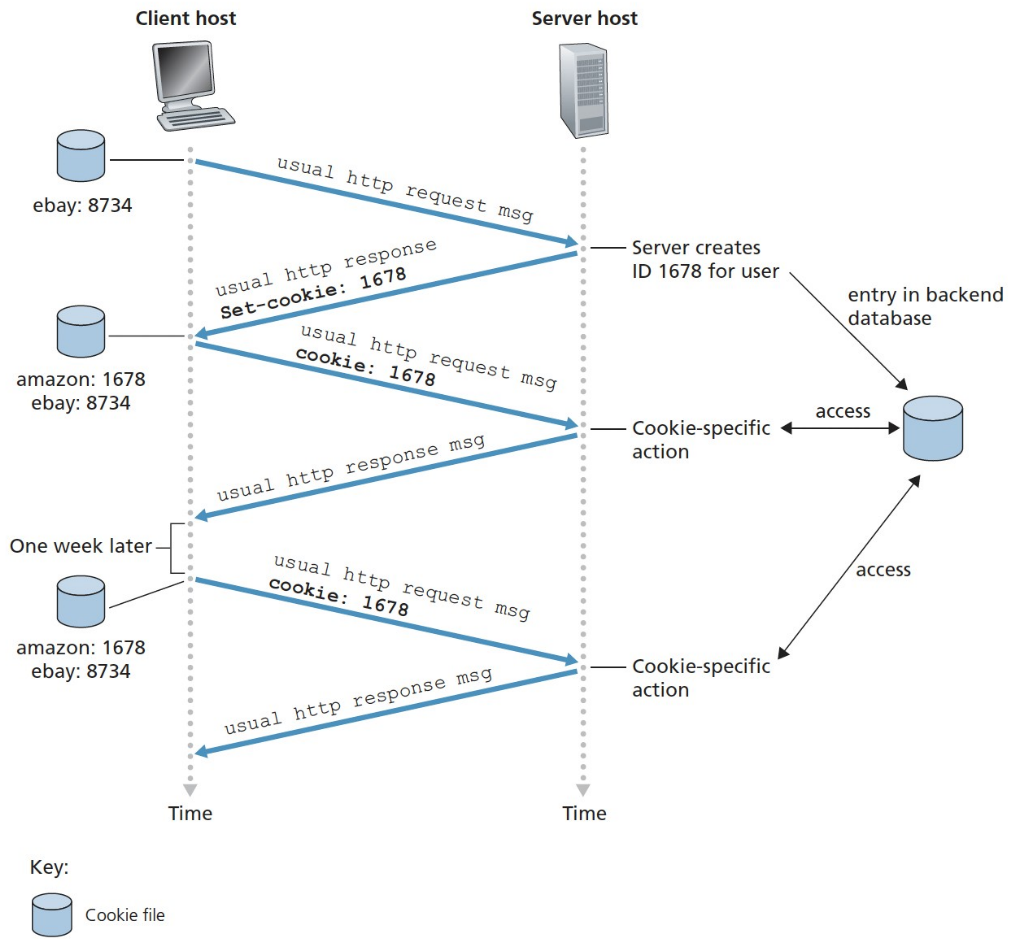
Selection of Common Status Codes and Associated Phrases

- 200 OK – request succeeded and the information is returned in the response
- 301 Moved Permanently – requested object has been permanently moved; the new URL is specified in Location: header of the response message
  - Client software will automatically retrieve the new URL
- 400 Bad Request – generic error code indicating the request could not be found
- 404 Not Found – requested document does not exist on this server
- 505 HTTP Version Not Supported – requested HTTP protocol version is not supported by the server

User-Server Interaction – Cookies

- HTTP server is stateless
- It can be desirable for a web site to identify users, either:
  - The server wishes to restrict user access
  - It wants to serve content as a function of the user identity
- For these purposes, HTTP uses cookies
  - Allow sites to keep track of users
  - Most (major) commercial sites use cookies today

Figure 2.10 – Keeping User State with Cookies



Cookies

- Cookie technology has four components
  - Cookie header line in the HTTP response message
  - Cookie header line in the HTTP request message
  - Cookie file kept on the user’s end system and managed by the user’s browser
  - A back-end database at the web site
- Example 2.10 – User visiting Amazon with Edge
  - Visits Amazon for the first time
    - When the request comes into the web server, the server creates a unique ID and creates an entry in its back-end database that is indexed by the identification number
    - Server responds to browser, including the HTTP response to Set-cookie header, which contains the identification number
    - The server can keep track of where you’ve been
      - Recommends similar products
      - Come back a week later – ”remembers” what was looked at
- Cookies can be used to create a user session layer on top of stateless HTTP

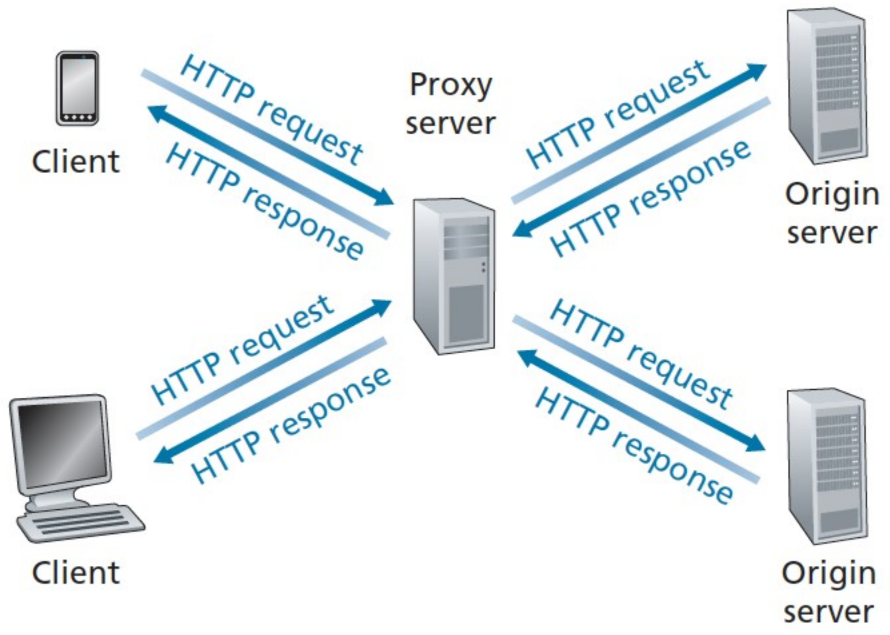
Web Caching

- Web cache – proxy server – network entity that satisfies HTTP requests on behalf of the origin server
- 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 configured, each request for an object is first directed to the web cache

Web Caching - Steps

- Browser establishes a TCP connection to the web cache and sends an HTTP request for the object to the web cache
- Web cache checks to see if it has a copy of the object locally stored
  - If so, the web cache returns the object within an HTTP response message to the client browser
- If the web cache does not have the object, the cache opens a TCP connection to the origin server
  - Sends an HTTP request for the object into the cache-to-server TCP connection
  - After receiving the request, the origin server sends the object within an HTTP response to the web cache
- 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 existing TCP connection between browser and web cache

Figure 2.11 – Clients Requesting Objects Through a Web Cache



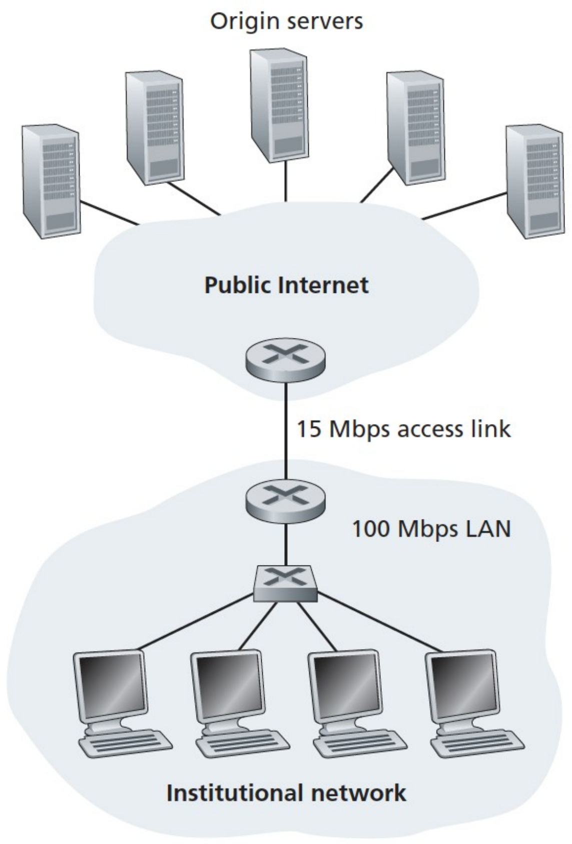
Caching

- Note that a cache is both a server and 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 purchased and installed by an ISP
    - College – install a cache on its campus network and configure all of the campus browsers to point to the cache
    - Major residential ISP might install one or more caches in its network and preconfigure its shipped browsers to point to the installed caches
- Also – local web cache folder might reside on your machine

Caching – A Necessity

- Deployment for two reasons
  - Can substantially reduce response time for a client request, particularly if bottleneck bandwidth between client and origin is much less than the bottleneck bandwidth between client and cache
    - Cache can rapidly retrieve object for client
  - Can substantially reduce traffic on an access link to the internet
    - Result: don't have to scale up bandwidth as frequently

Figure 2.12 – Bottleneck Between Institutional Network and the Internet



Deeper Understanding of Cache Benefits

- Two networks – institutional network and the rest of the public internet
- Institutional network is a high-speed LAN
- Router in network and a router in the internet are connected by a 15 Mbps link
- Origin servers are attached to the internet but are located all over the globe
- Assume: average object size is 1 Mbits, average request rate is 15 requests per second
- Assume: HTTP request messages are small and create no traffic in the networks, time it takes from when the router on the internet side forwards an HTTP request is 2s on average
    - Last delay – “Internet delay”

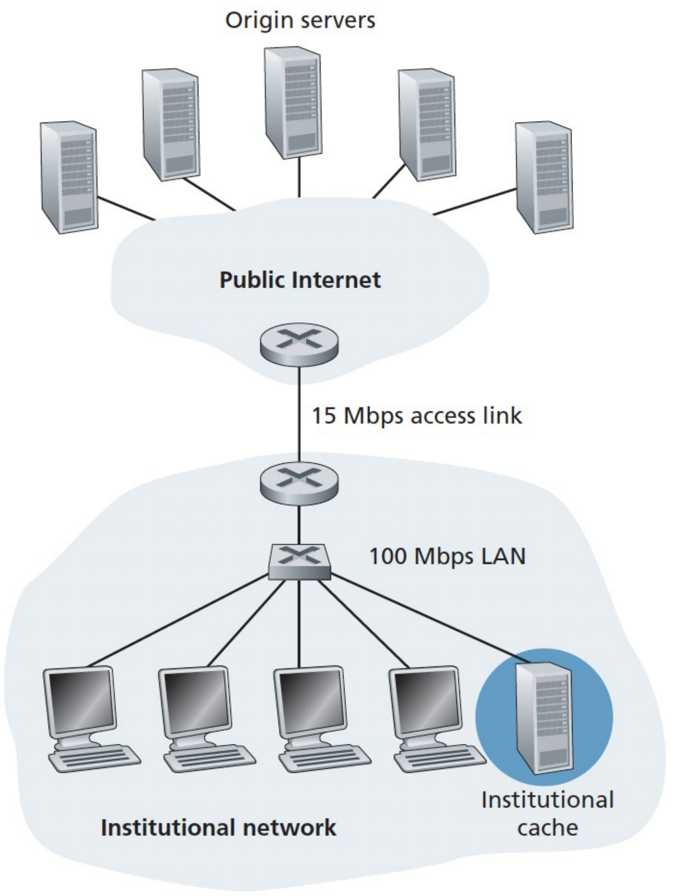
Caches and Better Understanding

- Total response time is the sum of
  - LAN delay
  - Access delay (between the two routers)
  - And Internet delay
- Estimate the delay:
  - (15 requests/sec)*(1 Mbits/request)/(100 Mbps) = .15
- Traffic intensity on the access link:
  - (15 requests/sec)*(1 Mbits/request)/(15 Mbps) = 1
- A traffic intensity of .15 on a LAN – tens of ms of delay
- But – as traffic intensity approaches 1, the delay on a link becomes very large and grows without bound
  - Average response time to satisfy requests could be minutes, if not longer

Possible Solutions?

- Increase access rate from 15Mbps to maybe 100Mbps
  - Lowers traffic intensity on the access link to .15
  - Leads to negligible delays between the two routers
  - Response time will be roughly 2s.
  - But – institution has to upgrade its access rate – costly
- Utilizing a web cache
  - Hit rates (fraction of requests satisfied by cache) range from .2 to .7
  - In our hypothetical example – .4
  - Clients and cache are connected on the same high-speed LAN, 40 percent of requests will be satisfied almost immediately, within maybe 10ms
  - Remaining 60 percent must pass through the access link
    - But, intensity on the access link then drops from 1 to .6
  - Typically – traffic intensity less than .8 means a small delay, in the order of 10s of ms
  - Average delay: .4*(0.01s) + .6*(2.01s) ☾ 1.2s
  - Lower response time than the first approach

Figure 2.13 – Adding a Cache to the Institutional Network



Conditional GET

- Although caching can reduce response times, it introduces a new problem
  - Copy of an object residing in the cache may be stale
- HTTP has a mechanism that allows a cache to verify objects are up to date – Conditional GET
- Considered as such if
  - The request messages uses the GET method
  - The request message includes an If-Modified-Since: header line
- Usually cache sends a GET, with host, and then If-modified-since
  - Server responds Modified/Not, as well as when, and the server

HTTP/2.0

- Standardized in 2015
- Over 40% of the top 10 million sites were supporting HTTP/2 in 2020
    - Like, what makes a site a top site, though?
- Most browsers also support HTTP/2
- Goals
    - Reduce perceived latency by enabling request and response multiplexing over a single TCP connection
    - Provide request prioritization and server push
    - Provide efficient compression of HTTP header fields
- Changes how data is formatted and transported between client and server

See the Need…

- HTTP/1.1 uses persistent TCP connection
  - Web page is sent from server to client over a single TCP connection
  - Having only one TCP connection per web page – number of sockets available at the server is reduced and each transported page gets a “fair share” of network bandwidth
  - Developers discover that sending all objects over a single TCP connection has a Head of Line blocking problem
    - Example – web page with video clip near the top, small objects below the video
      - Also suppose low-to-medium speed bottleneck link
    - The video clip will take a long time to pass through the bottleneck link, while the small objects are delayed as they wait behind the video clip
    - HTTP/1.1 browsers work around this by opening multiple parallel TCP connections
      - Small objects can be received and posted at a much faster speed
- TCP congestion control provides browsers with an incentive to use multiple parallel TCP connections rather than just one connection
  - Aims to give each TCP connection sharing a bottleneck link an equal share of the available bandwidth of the link
- Try to get rid of/reduce the number of parallel TCP connections for transporting a single page

HTTP/2 Framing

- Solution for HOL blocking – break each message into small frames and interleave request/response messages on the same TCP connection
- Example: web page, with 1 video, 8 small objects
  - Server will receive 9 concurrent requests from any browser wanting to see this page
  - Each request – server needs to send 9 competing HTTP response messages to the browser
  - Frames are of fixed length
    - Video – 1000 frames
    - Each of the smaller objects – 2 frames
  - Frame interleaving – one frame from video clip, first frames of each of the small objects are sent
    - Second frame of video clip – last frames of each of the small objects are sent
    - All smaller objects are sent after sending a total of 18 frames
      - 16 from smaller objects
      - 2 from video
  - No interleaving – smaller objects would be sent only after sending 1016 frames

Response Message Prioritization and Server Pushing

- Message prioritization – allows developers to customize the relative priority of requests to better optimize application performance
- When a client sends concurrent requests to a server, it can prioritize the responses by assigning a weight between 1 and 256 to each message
  - Higher number ☾ higher priority
  - Server can send first the frames for the responses with the highest priority
    - Client also states each message's dependency on other message via specifying ID of message
- Server to send multiple responses for a single client request
  - Respond to original request
  - Server pushes additional objects to the client without the client having to request each one
  - Server can analyze the HTML page, identify the objects needed and send them to the client before receiving explicit requests

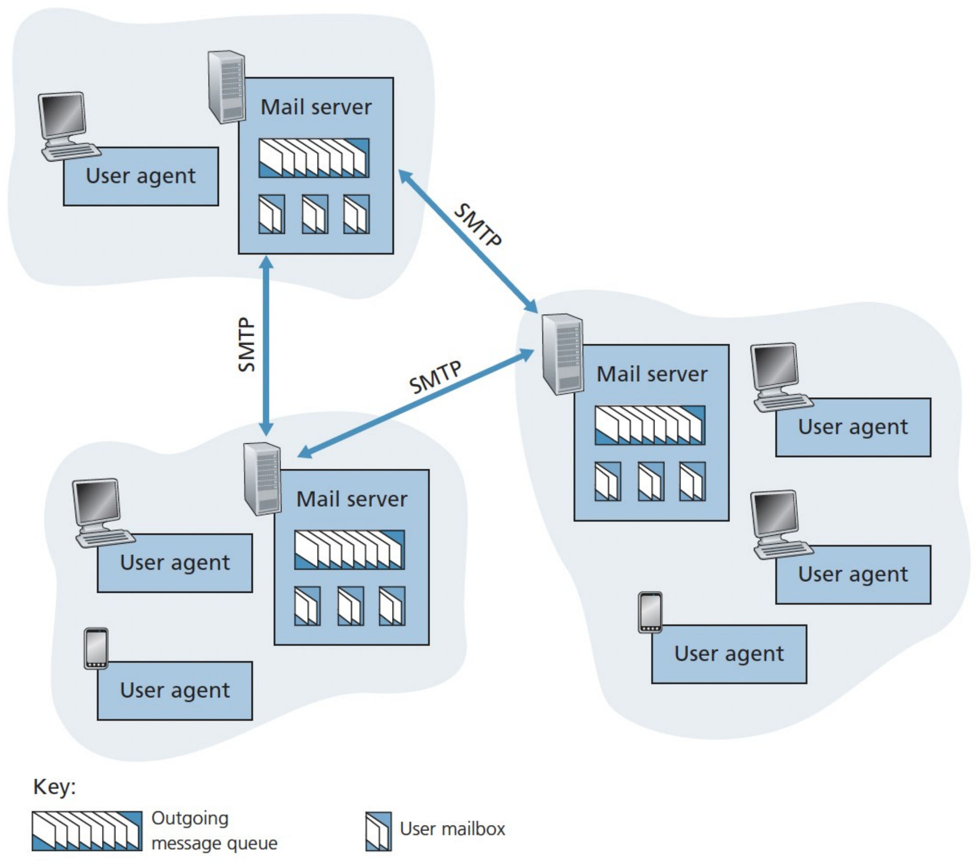
HTTP/3

- Referred to as QUIC (see Ch. 3)
- Implemented in application layer over UDP protocol
- Several interesting features
  - Message multiplexing
  - Per-stream flow control
  - Low latency connection establishment
- Has not yet been fully standardized

Email in the Internet

- Asynchronous communication medium
  - People send and read messages when it is convenient for them, without having to coordinate with someone's schedule
- Modern email has many "powerful" features, including messages with attachments, hyperlinks, HTML-formatted information, and embedded photos
- Figure 2.14 – high-level view of the Internet mail system
  - User agents
  - Mail servers
  - Simple Mail Transfer Protocol (SMTP)
- When someone is finished writing their message to a recipient, it is sent to the mail server where the message is placed in the mail server's outgoing message queue
- Each recipient has a mailbox located in one of the mail servers
- If a message cannot be sent by the server, the server holds the message in a message queue and attempts to move the message later
  - Reattempts – every 30-ish minutes?

Figure 2.14 – A High-Level View of the Internet E-Mail System



Email Subtopics

- SMTP
- Mail Message Formats
- Mail Access Protocols

SMTP

- Principal application-layer protocol for email
- Uses TCP to transfer mail from sender’s mail server to recipient’s mail server
- Two sides
  - Client – executes on the sender’s mail server
  - Server – executes on the recipient’s mail server
  - Both run on every mail server
- When a mail server sends mail to other mail servers – client
- When a mail server receives mail from other mail servers – server
- Legacy technology with some archaic features
  - Restricts the body of all mail messages to simple 7-bit ASCII
  - Made sense in the 1980s when transmission capacity was scarce, no one sending large attachments
  - Today – requires binary multimedia data to be encoded to ASCII before being sent over SMTP
    - Requires corresponding ASCII message to be decoded back to binary after SMTP transport

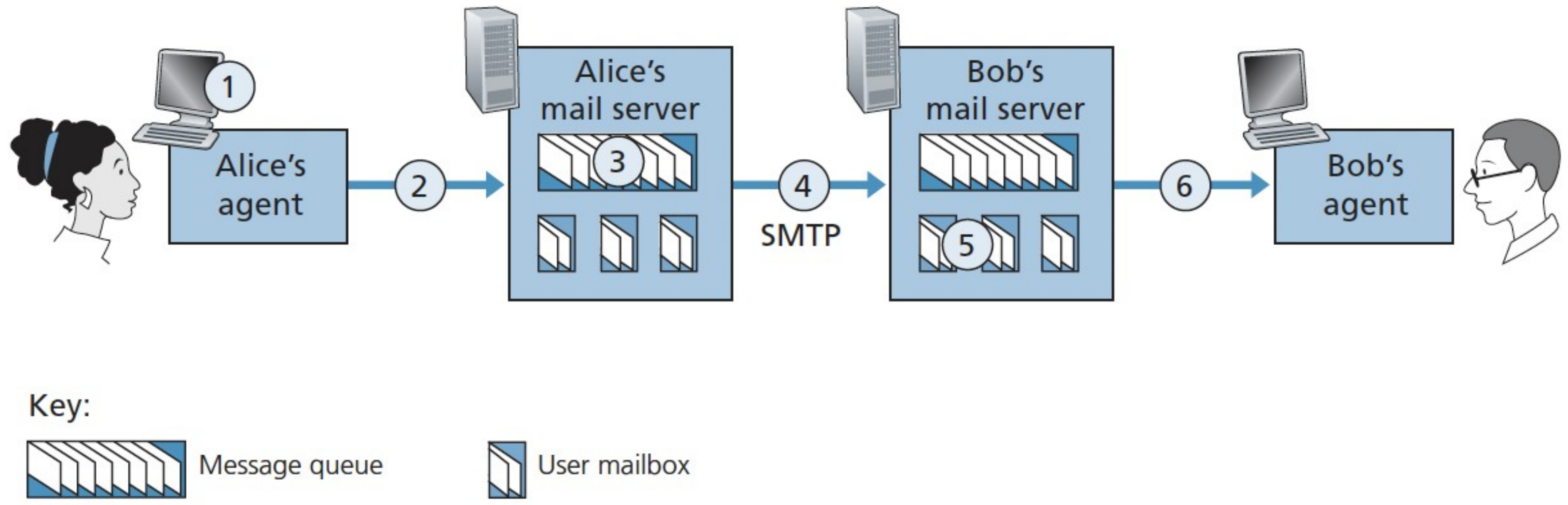
Basic Operation of SMTP

- Alice users her user agent for email, provides Bob’s email address (for instance, bob@someschool.edu), composes a message, and instructs the user agent to send the message
  - User agent – like Outlook, Thunderbird, Mail client on Mac
- Alice’s user agent sends the message to the mail server, where it is placed in a message queue
- Client side of SMTP, running on Alice’s mail server, sees the message in the message queue; opens TCP connect to the SMTP server running on Bob’s mail server
- After some SMTP handshaking, the client sends Alice’s message 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
- Does not normally use intermediate mail servers for sending mail; TCP provides direct connection between servers

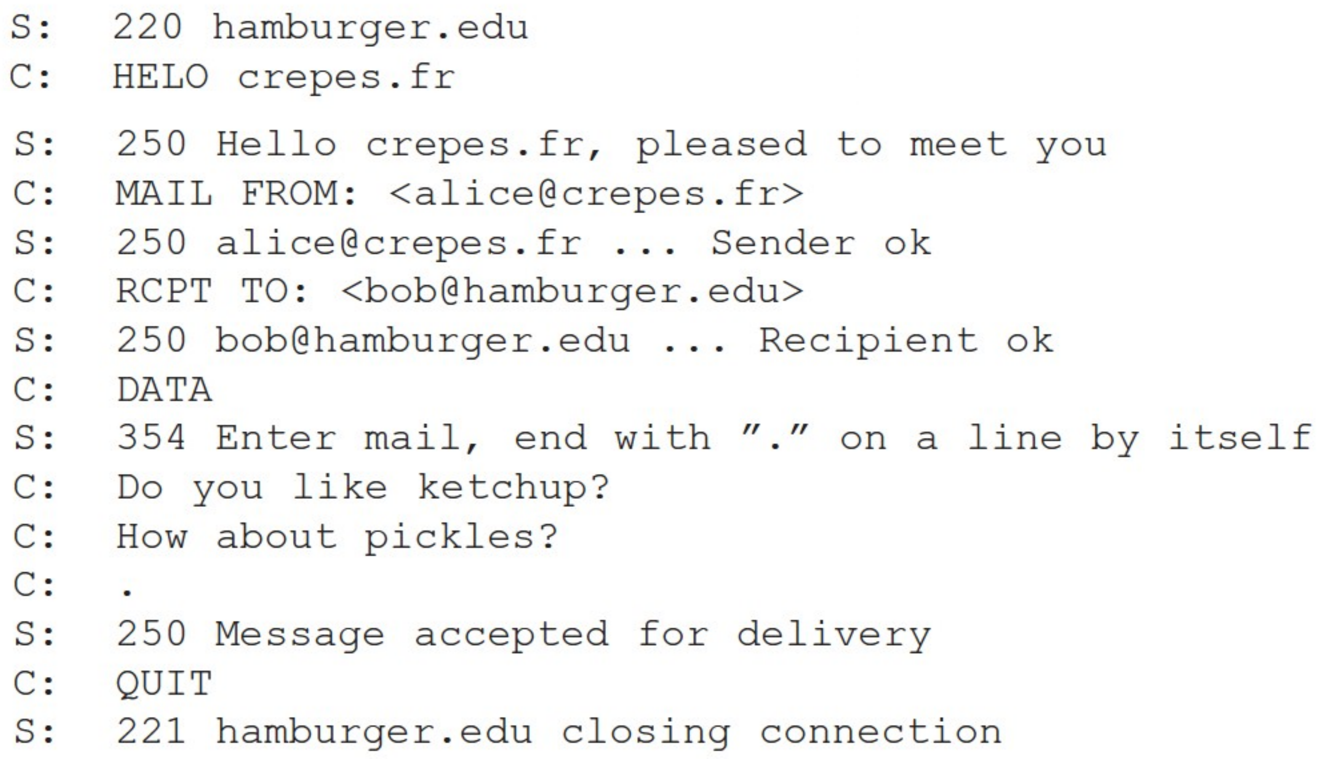
Operation of SMTP

- Transfers a message from a sending mail server to receiving mail server
- Has many similarities with face-to-face human communication
- Client has TCP establish a connection to port 25 at the server SMTP
- Server down? Client tries again later
- Once established, handshaking (introducing one server to the other)
- SMTP client indicates the mail address of the recipient
- Message sent via TCP
- Client repeats process over the same connection if it has other messages to send to the server; otherwise, TCP is told to close the connection

Figure 2.15 – Alice Sends Bob a Message



Example Transcript of Messages



```
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
```

Operation of SMTP

- Recommended that you use Telnet (ssh?) to carry out a direct dialogue with an SMTP server
  - Command: telnet serverName 25
    - Establishing a connection between your local host and the mail server
    - Should immediately receive the 220 reply from server
    - Issue SMTP commands at appropriate times
      - HELO
      - MAIL FROM
      - RCPT TO
      - DATA
      - CRLF

Mail Message Formats

- When an email message is sent from one person to another, a header containing peripheral information precedes the body
    - See RFC 5322
    - Header and body are separated by a blank line (CRLF)
- Each header line contains readable text
    - Keyword, followed by a colon followed by a value