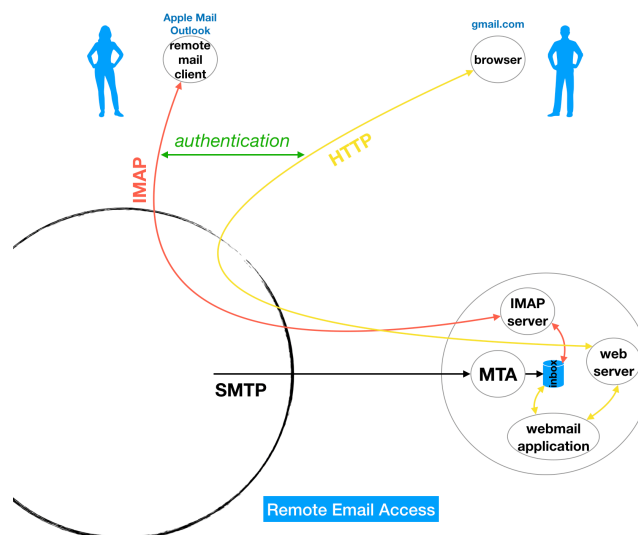


Application Layer (IV)

MUA to MTA Communication

- MUA and MTA on the same host in the past
 - MUA and MTA communicate using files
 - use of host's authentication methods
- MUA and MTA communicate over a network in nowadays
 - SMTP was not designed for retrieving messages
 - it can push message to the server, but cannot pull the message from the server
 - retrieving mail: POP-3 and IMAP (include authentication)
 - POP-3 (post office protocol - version 3): port 110
 - manage messages locally, cannot access messages/folders from different machines
 - IMAP (Internet Mail Access Protocol):
 - significantly more complex
 - enable manage and organize message on the MTA, so can access folders across machines
 - obtain components of messages, useful for low bandwidth



MIME

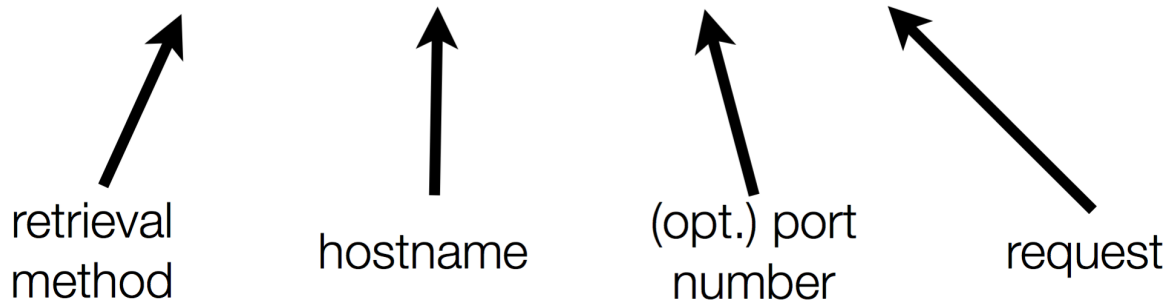
- **Problem:** SMTP was designed to deliver limited length, English text only
- **Solution:** MIME (Multipurpose Internet Mail Extensions) (but can be use for HTML)
 - make everything look like text
 - pack the message (including text, characters, images, icons, audio, video, etc.) and mark it with content type so it can be unpacked and rendered on the receiving end

- Emails are **transmitted via SMTP in MIME format.**

HTTP Overview

- URL (Universal Resource Locator)

http://www.cs.colby.edu:80/dir/file.html



- Web page
 - A html file references to various objects such as images, movies, etc.
- Hypertext Transfer Protocol (HTTP)
 - defines how web clients request web pages from Web servers and how servers transfer Web pages to clients.
 - Client and Server architecture
 - Client side - web browser
 - Server side - web server (Apache, Microsoft Internet Information Server)
 - TCP underneath
 - stateless protocol
 - servers maintain no information about the client
 - Non-persistent and persistent connections
 - Non-persistent (HTTP/1.0)
 - set up a TCP connection for each object
 - those connections may be sequential or parallel. The browser controls the degree of parallelism. (Default 5-10 parallel connections)
 - Persistent (default HTTP/1.1)
 - HTTP/1.1 persistent connection, the server leaves the TCP connection open after sending a response, and close a connection when it isn't used for a certain time (configurable). Allow back-to-back objects/requests transmission.
 - HTTP/2 allows multiple requests and replies to be interleaved in the same connection, and has mechanism for prioritizing HTTP message requests and replies within this connection.
- Cookies:
 - User-Server interaction
 - most major commercial web sites use cookies
 - a cookie header line in the HTTP response to let the client's browser store an identification number for the user; when the user visit the same site again, the client request contains a cookie header line in the HTTP request with the ID number; the ID number is stored by the

serve in a database to keep track of the user's status (e.g., items in the Amazon cart); the browser also maintains a cookie file for the user (e.g., the ID number)

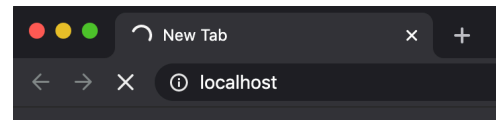
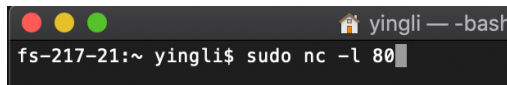
- Web caching
 - proxy server: satisfies HTTP requests on behalf of an origin web server; has its own disks storage and keeps copies of recently requested objects
 - users can config their computer or browsers so that all the HTTP requests direct to the proxy server first; if the proxy server doesn't have the requested object, the proxy server send a HTTP request into the cache-to-server connection; the proxy server stores a copy in its local storage and send a copy to the client browser with a HTTP response.
 - typically purchased and installed by an ISP (e.g., a college may install a cache for its campus network and config all of the campus browsers to point to the cache)
 - benefits: reduce the response time for a client request; reduce traffic on an institution's access link to the Internet (so they don't have to upgrade bandwidth quickly, therefore reduce cost); reduce traffic in the Internet as a whole, thereby improving performance for all application — Content Distribution Networks (CDNs)

HTTP GET Method

- GET
 - let a client requests data from the server
 - e.g., open a website

GET Examples

- Let's take a look at the details of a http request. In this example, we use localhost as a server and send a http request to itself. You may get different information in the request when using different web browsers.

A terminal window titled 'yingli — nc - sudo — 80x24' showing the output of the netcat listener. The output is an HTTP GET request from a Google Chrome browser on a Macintosh.

```
[fs-217-21:~ yingli$ sudo nc -l 80
GET / HTTP/1.1
Host: localhost
Connection: keep-alive
Cache-Control: max-age=0
sec-ch-ua: "Google Chrome";v="89", "Chromium";v="89", ";Not A Brand";v="99"
sec-ch-ua-mobile: ?0
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_6) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/89.0.4389.82 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
```