CSE 3231
Computer Networks

Chapter 7
Applications
part 2

William Allen, PhD

Spring 2022

Static Web Pages

- *Static* web pages are pre-built files
  - Their contents don't change from one viewing to the next (unless edited by admin)
  - Often written in HTML or similar languages
- *Dynamic* web pages are generated by programs running on the client or server
  - Pages are created on demand by software
  - They can create pages that depend on user input or from data pulled from a database

Static Web Pages

Although static web pages are pre-built files, they can still be interactive and can contain a mix of text and images

- The HyperText Markup Language (HTML) can be used to create static web pages
- Forms can be used to gather user input
- Style sheets can customize presentation
- They can include embedded or linked images and media

Static Web Pages

Progression of features through HTML 5.0

Item	HTML 1.0	HTML 2.0	HTML 3.0	HTML 4.0	HTML 5.0
Hyperlinks	x	x	x	x	x
Images	x	x	x	x	x
Lists	x	x	x	x	x
Active maps & images		x	x	x	x
Forms		x	x	x	x
Equations			x	x	x
Toolbars			x	x	x
Tables			x	x	x
Accessibility features				x	x
Object embedding				x	x
Style sheets				x	x
Scripting				x	x
Video and audio					x
Inline vector graphics					x
XML representation					x
Background threads					x
Browser storage					x
Drawing canvas					x

Dynamic Pages & Web Applications

Dynamic pages can be generated by:

- software running on the server
- a program running on the client
- interactions between programs on both the client and the server

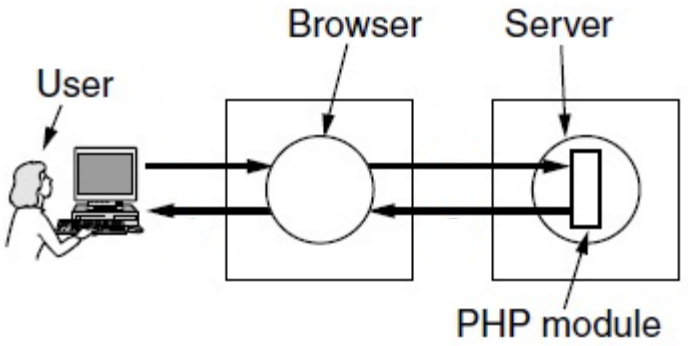
A number of different programming languages can be used to make pages

- e.g., PHP at server, JavaScript at client
- Java or PHP on either endpoint, etc.

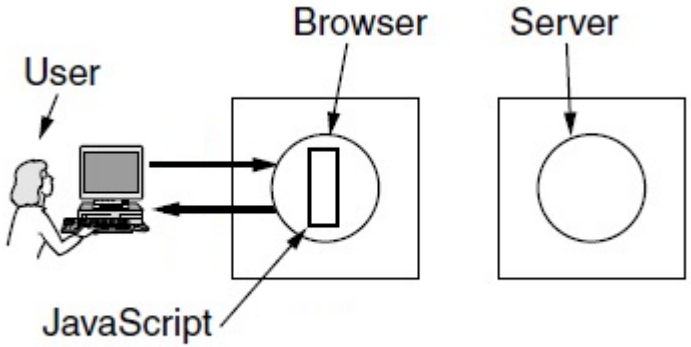
Dynamic Pages & Web Applications

These diagrams show the difference between creating dynamic pages on the server or in a client program

- The browser can interact directly with a program on the server to create web pages
- Or the client can create the pages in the browser

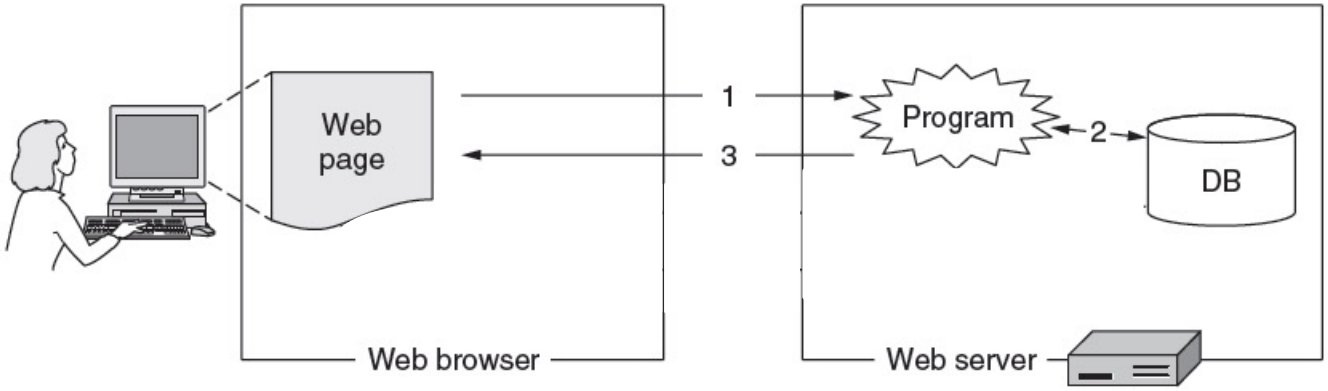


Server-side scripting with PHP



Client-side scripting with JavaScript

Dynamic Pages & Web Applications



Dynamic pages can be created on the server

1. Browser sends request to server
2. Server executes program to manage reply
3. Program extracts data from database, builds web page and sends it to the browser

Dynamic Pages & Web Applications

Web page that gets form input and calls a server program

Form for user input

```
<html>
<body>
<form action="action.php" method="post">
<p> Please enter your name: <input type="text" name="name"> </p>
<p> Please enter your age: <input type="text" name="age"> </p>
<input type="submit">
</form>
</body>
</html>
```

PHP server program that creates a custom Web page using the input data

PHP calls get data from the user's input

```
<html>
<body>
<h1> Reply: </h1>
Hello <?php echo $name; ?>
Prediction: next year you will be <?php echo $age + 1; ?>
</body>
</html>
```

Resulting Web page (includes user inputs "Barbara" and "32")

```
<html>
<body>
<h1> Reply: </h1>
Hello Barbara.
Prediction: next year you will be 33
</body>
</html>
```

Dynamic Pages & Web Applications

JavaScript program produces the result page in the browser (this code will not be visible on the screen)

```
<html>
<head>
<script language="javascript" type="text/javascript">
function response(test_form) {
    var person = test_form.name.value;
    var years = eval(test_form.age.value) + 1;
    document.open();
    document.writeln("<html> <body>");
    document.writeln("Hello " + person + ".<br>");
    document.writeln("Prediction: next year you will be " + years + ".");
    document.writeln("</body> </html>");
    document.close();
}
</script>
</head>
```

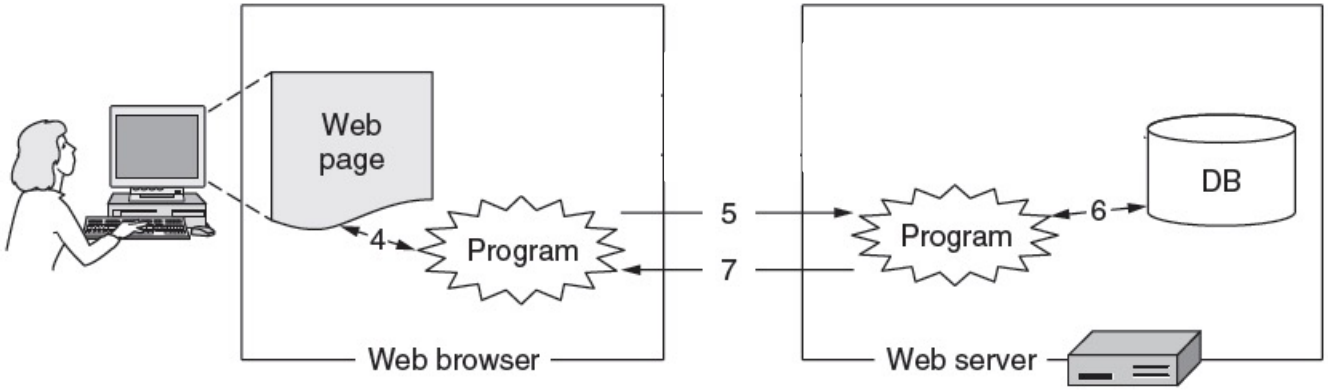
Page with form is displayed on screen and gets user input before calling the above program

```
<body>
<form>
Please enter your name: <input type="text" name="name">
<p>
Please enter your age: <input type="text" name="age">
<p>
<input type="button" value="submit" onclick="response(this.form)">
</form>
</body>
</html>
```

Input form shown on previous page

"*submit*" button added to page, including a call to the "response" program

Dynamic Pages & Web Applications



Dynamic pages can be created by software on the client interacting with software on the server

4. Browser interacts with a program on the client
5. Client sends requests to a program on the server
6. Server program extracts data from the database
7. Server program sends data to client program which completes the webpage and delivers it to browser

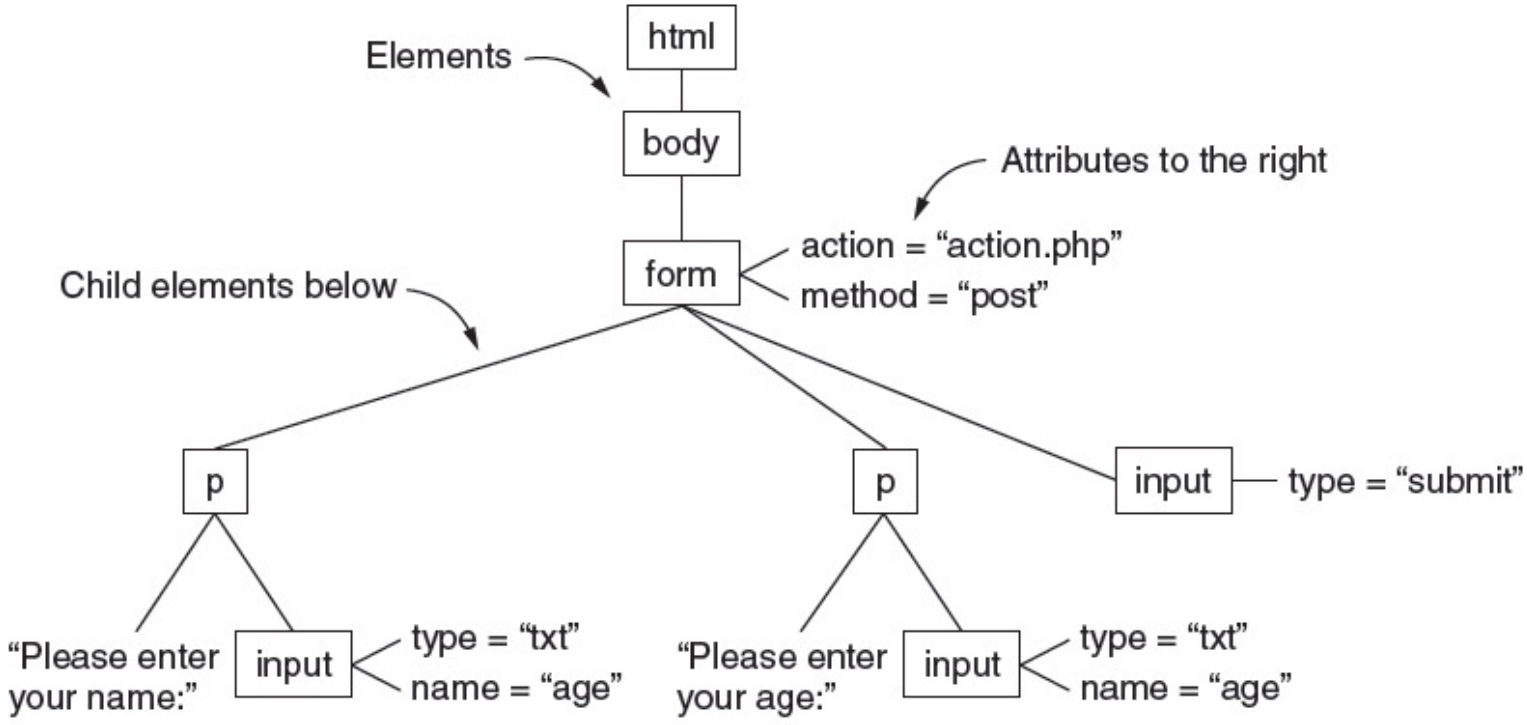
Dynamic Pages & Web Applications

Web applications use a set of technologies that are designed to work together

- HTML: format information as web pages
- DOM: parts of pages can be updated while they are already displayed
- XML: document format used by programs to exchange data with the server
  - Programs can send and retrieve XML data in an asynchronous manner
- JavaScript can bind all this together

Dynamic Pages & Web Applications

The DOM (Document Object Model) tree represents web pages as a structure that programs can alter



Dynamic Pages & Web Applications

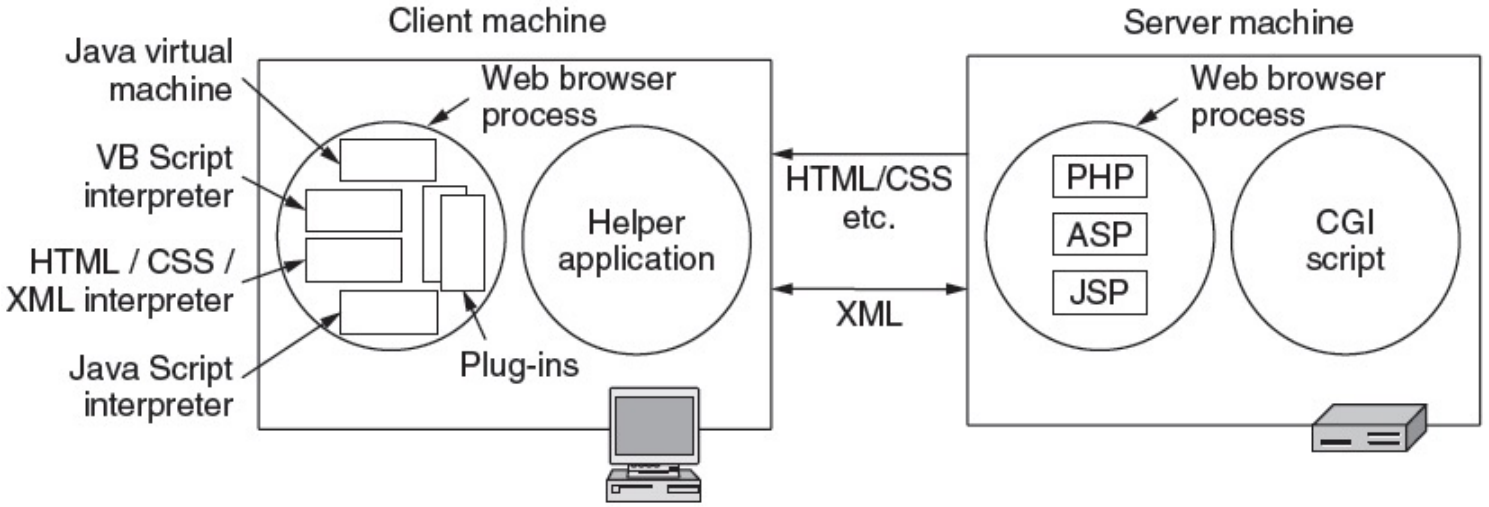
XML is a markup language like HTML, but focuses on document structure, not on page presentation

```
<?xml version="1.0" ?>

<book_list>

<book>
   <title> Human Behavior and the Principle of Least Effort </title>
   <author> George Zipf </author>
   <year> 1949 </year>
</book>

<book>
   <title> The Mathematical Theory of Communication </title>
   <author> Claude E. Shannon </author>
   <author> Warren Weaver </author>
   <year> 1949 </year>
</book>

<book>
   <title> Nineteen Eighty-Four </title>
   <author> George Orwell </author>
   <year> 1949 </year>
</book>

</book_list>
```

Dynamic Pages & Web Applications

Web applications use a range of technologies and programming languages

- HTML, CSS, XML are used for exchanging data
- Browsers can execute VB, PHP, ASP, JSP, etc.
- CGI scripts run on servers to access databases, etc.



HTTP Caching

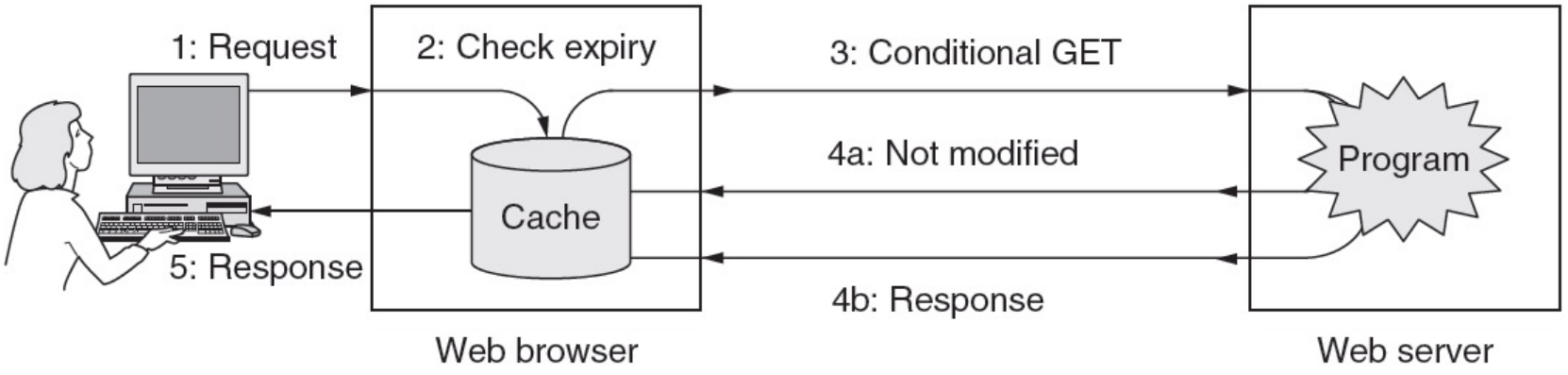
Improvements to web caching result from significant research, motivated by the overall benefits of efficient caching

- Clients benefit from being able to retrieve a page from a nearby cache because it can be displayed much more quickly than if it has to be fetched from a remote web server
- Servers benefit from having a cache that can successfully process a client's request because it reduces the load on the server

HTTP Caching

Web caching checks to see if the browser has a recent copy of the page. If not, it checks whether the server has updated the page before downloading it again

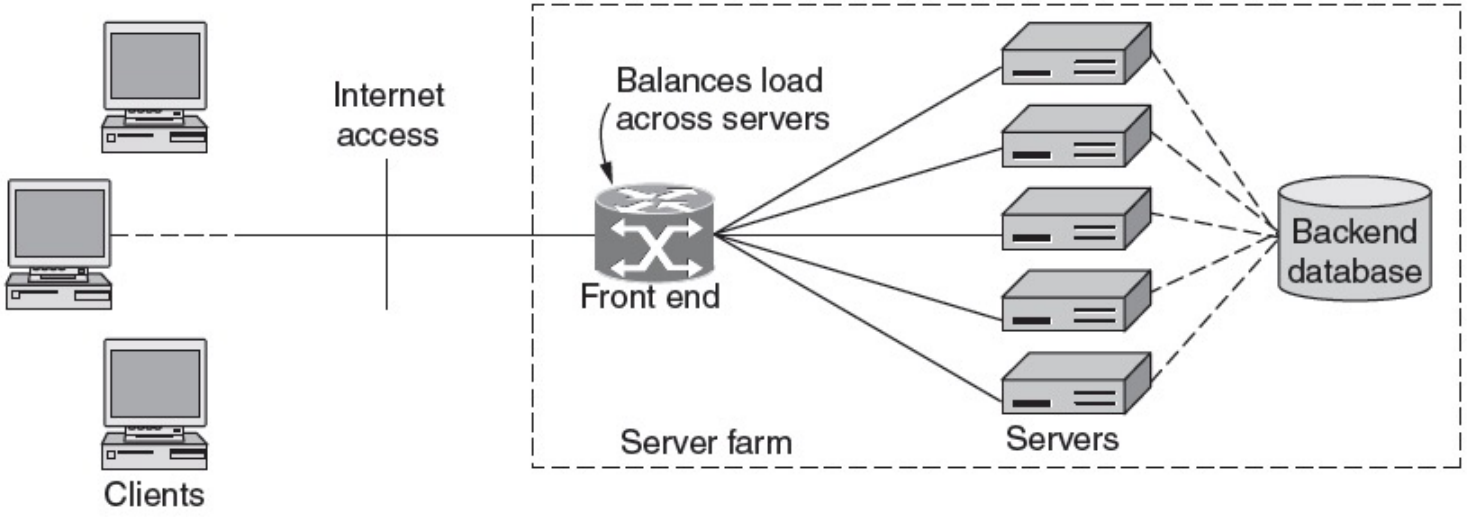
- Information in the headers is used for the status
- Multi-level caching is common (e.g., web proxy)



Server Farms

Server farms enable large-scale Web servers:

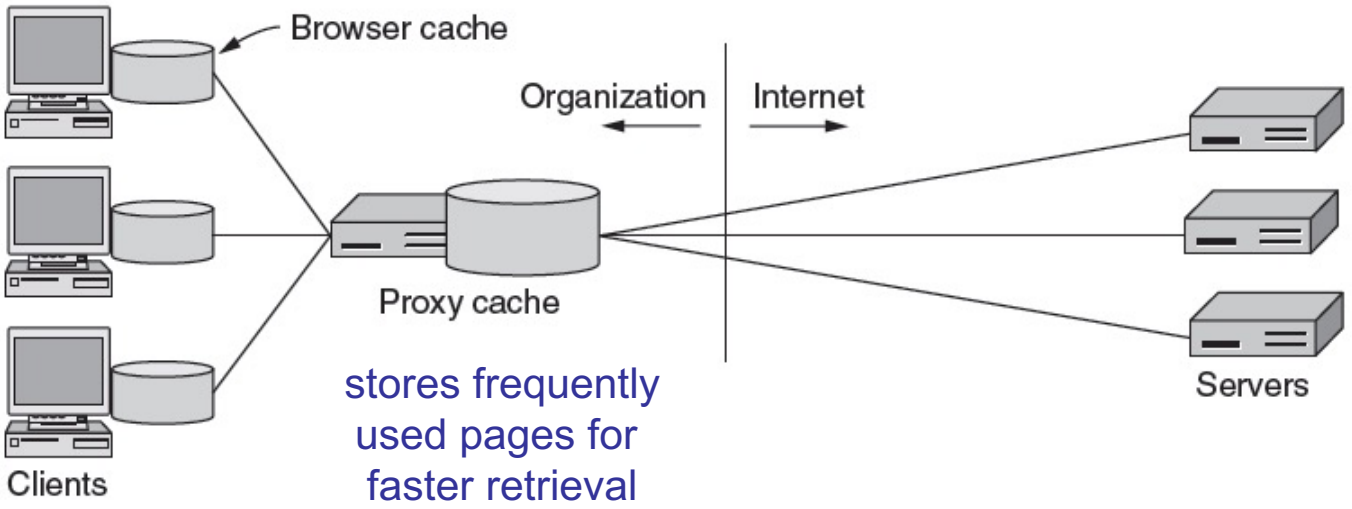
- Front-end processors can load-balance requests over a number of servers
- All servers access the same backend database to provide consistency in data retrieval/update



Proxy Caches

Proxy caches help organizations scale the Web

- Caches server content for a range of clients to lower response time and improve performance
- They can also implement organization policies by enforcing access control



Mobile Devices and the Web

Mobile devices are challenging clients:

- Relatively small screens
- Limited input capabilities, often lengthy input
- Network bandwidth is limited
- Connectivity may be intermittent
- Computing power is limited

Strategies to handle them:

- Content: servers provide mobile-friendly versions; transcoding can also be used
- Protocols: no real need for specialized protocols; HTTP with header compression is usually sufficient

Streaming Audio and Video

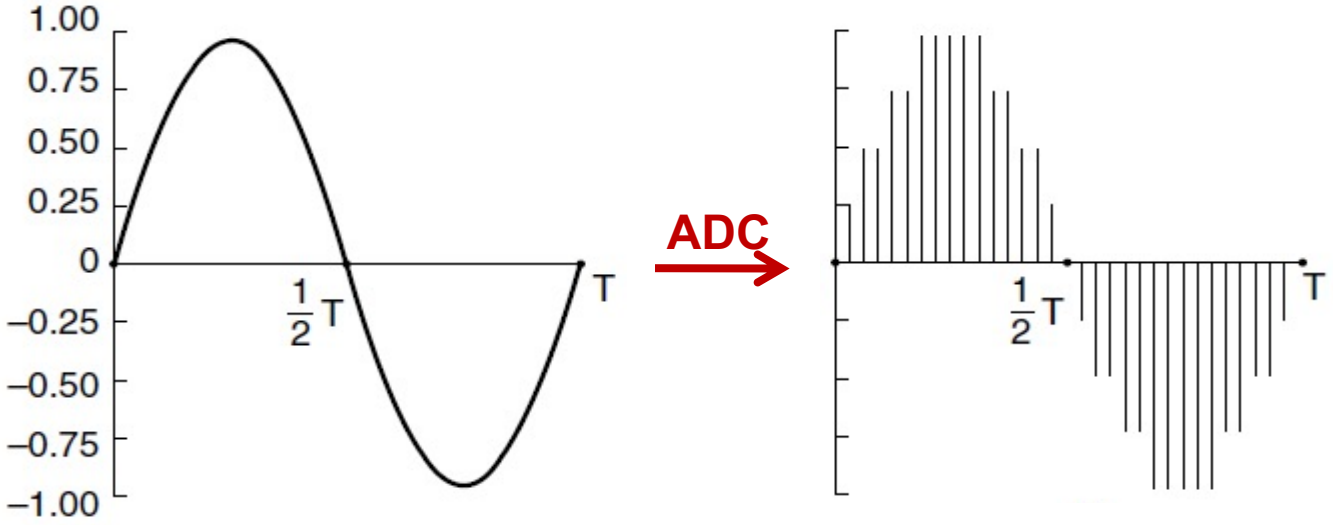
Audio and video have become common types of traffic for entertainment and business, particularly over mobile devices

- Digital audio
- Digital video
- Streaming stored media
- Streaming live media

Digital Audio

ADC (Analog-to-Digital Converter) produces digital audio from an analog input (microphone)

- Telephone: 8000 8-bit samples/second (64 Kbps)
- Computer audio is usually better quality (16 bits)



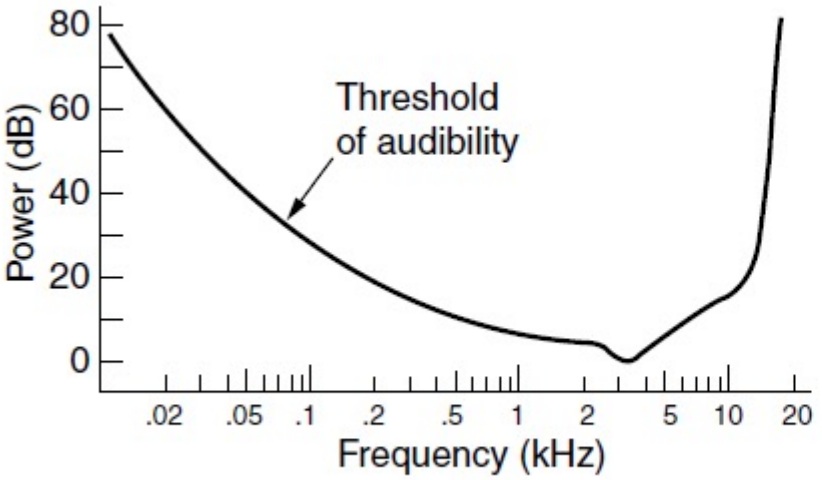
Continuous audio (sine wave)

Digital audio (sampled, 4-bit quantized)

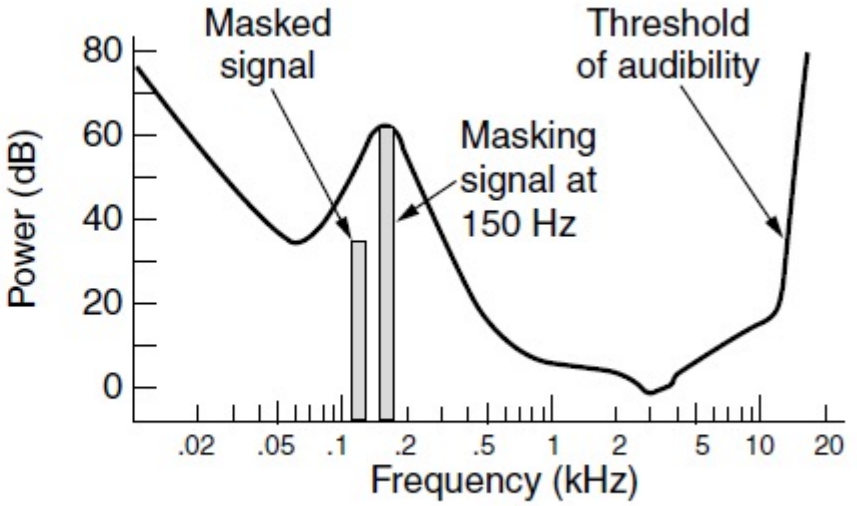
Digital Audio

Digital audio is typically compressed

- Lossy encoders (like AAC) exploit human perception
- Large compression ratios (can be >10X)



Sensitivity of the ear varies with frequency



Masking: a loud tone can mask nearby tones

Digital Video

Video is digitized as pixels (sampled, quantized)

- SD TV: 640x480 pixels, 24-bit color, 30 samples/sec
- HD (1080): 1920x1080, color depth and rates vary

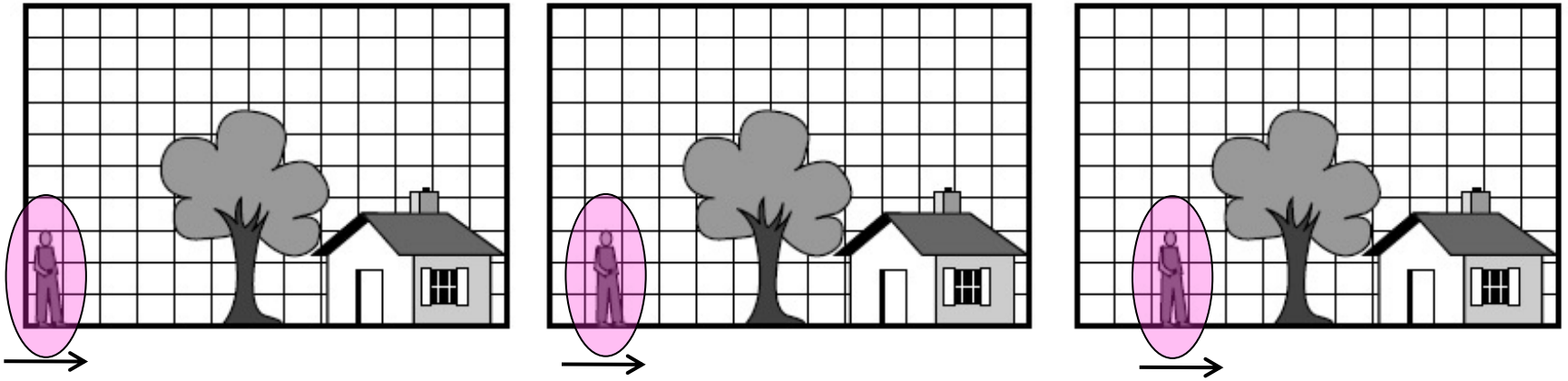
Video is compressed to reduce bandwidth

- Uncompressed SD video needs 27 MBytes/second
- Lossy compression exploits human perception
  - E.g., JPEG for still images, MPEG, H.264 for video
- Large compression ratios (often 50X for video)
- Video is normally > 1 Mbps, versus >10 kbps for speech and >100 kbps for music

Digital Video

MPEG compresses over a sequence of frames

- **I** (Intra-coded) frames: self-contained still pictures (jpeg)
- **P** (Predictive) frames: track motion in the video and record frame-by-frame *differences* to reduce repetition
- **B** (Bidirectional) frames: track motion in both directions

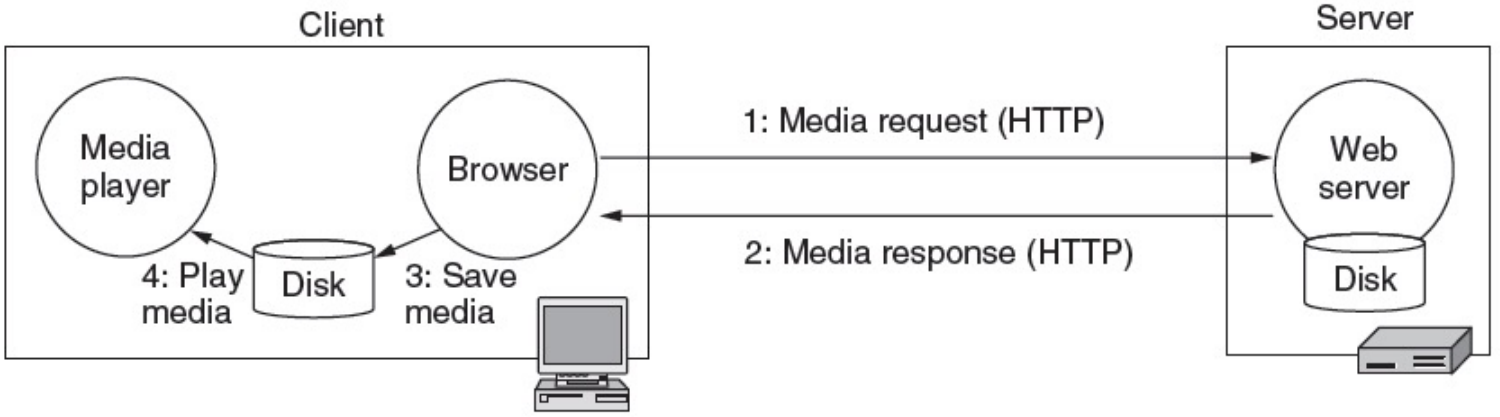


Three consecutive frames with stationary and moving components
Areas where no change occurs require storage of minimal data

Streaming Stored Media

A simple method to stream stored media, e.g., for video on demand, is to fetch the entire video as a file download

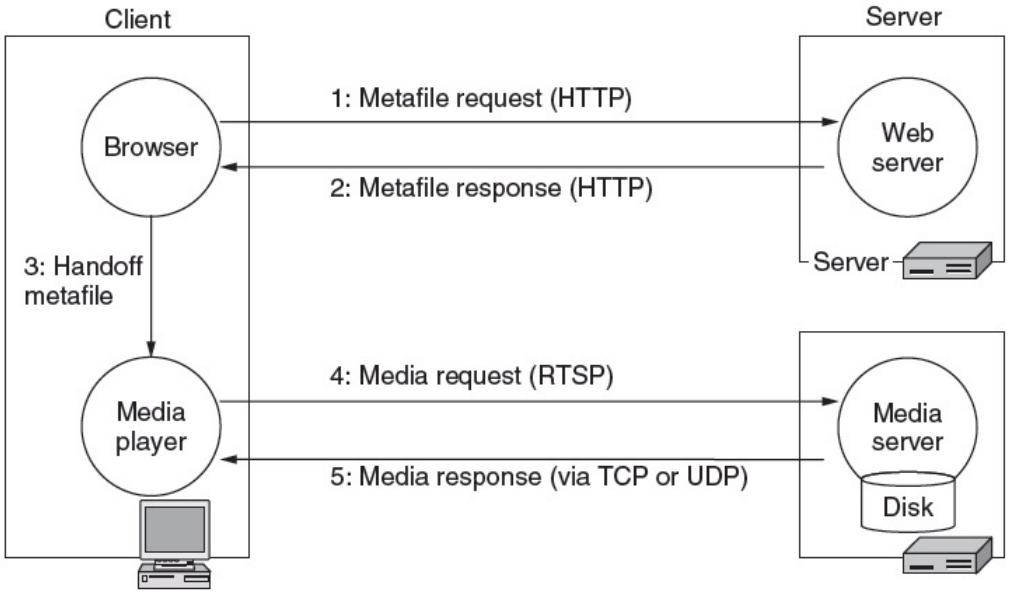
- Has large startup delay because it does not start playing until the file has been downloaded
- But, no buffering is needed, no delays or jitter



Streaming Stored Media

A more effective streaming method starts the playout right after streaming begins

- Example: RTSP (Real-Time Streaming Protocol)
- Browser makes connection and gets metadata
- Then hands off playback to the media player



Streaming Stored Media

Key problem: how to handle transmission errors

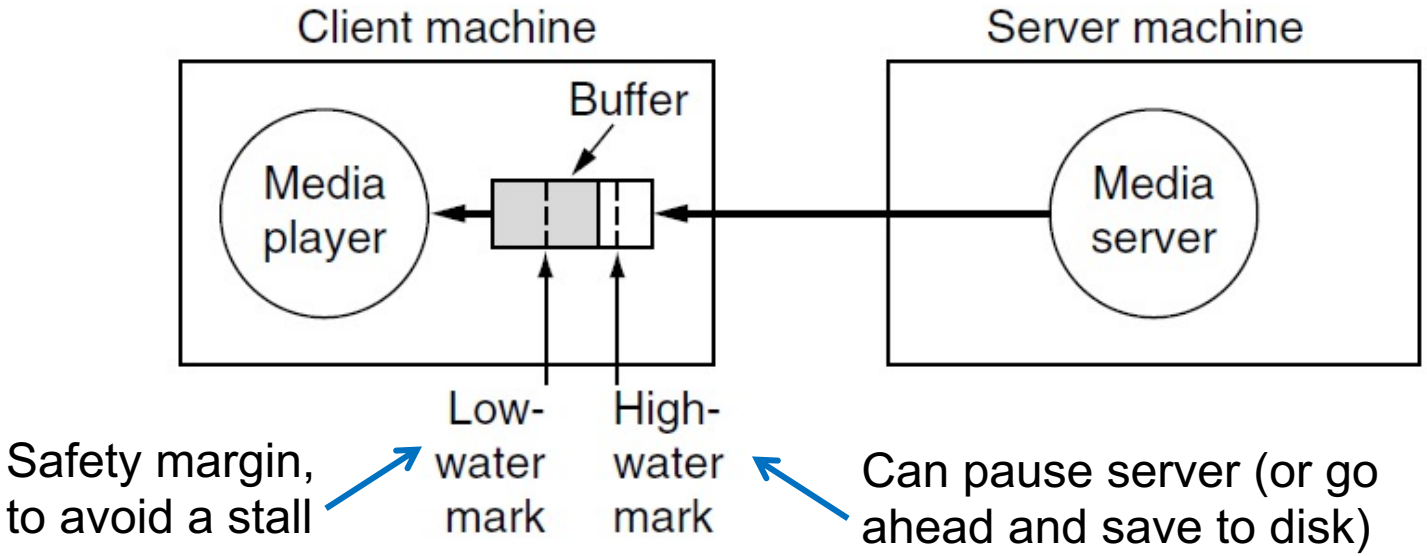
- We could use error detection/correction, but has its own issues (increased data size & processing time)

Strategy	**Advantage**	**Disadvantage**
Use reliable transport (TCP)	Repairs all errors	Increases jitter significantly
Add Forward Error Correction (FEC)	Repairs most errors	Increases overhead, decoding complexity and jitter
Interleave media	Masks most errors	Slightly increases decoding complexity and jitter

Streaming Stored Media

Key problem: media *may not arrive in time* for playout due to variable bandwidth and packet losses/retransmissions

- Client can buffer media to absorb jitter; we still need to pick an achievable media rate



Streaming Stored Media

RTSP commands

– Player sends messages to server to adjust rate of streaming to improve performance

Command	Server action
DESCRIBE	List media parameters
SETUP	Establish a logical channel between the player and the server
PLAY	Start sending data to the client
RECORD	Start accepting data from the client
PAUSE	Temporarily stop sending data
TEARDOWN	Release the logical channel

Streaming Live Media

Streaming live media is similar to the stored version, but adds several problems

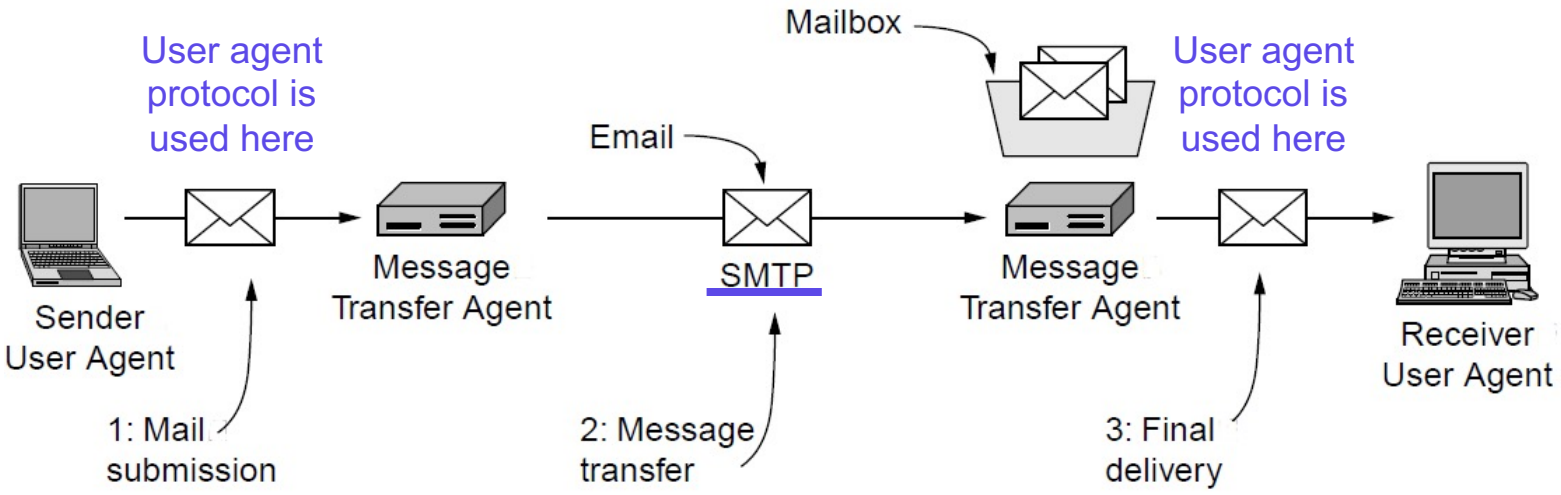
- Can't stream media faster than the "live rate" to get ahead and avoid jitter
  - Will have to use a larger buffer to absorb jitter
- Often have many simultaneous viewers
  - UDP with multicast is more efficient, but is rarely available, so many TCP connections are used
  - For very many users, *content distribution* methods are used (caching, proxies, distributed storage, etc.)

Email

- Email requires several different protocols to provide the full range of capabilities for users
- The architecture of the email system includes both local clients to send and read email and a collection of servers that exchange emails between networks and also provide users with local access to their messages
- Originally, email was a text-based system and protocols had to be added to support the exchange of attachments, images and media
  - All of these parts are specified in RFCs

Email

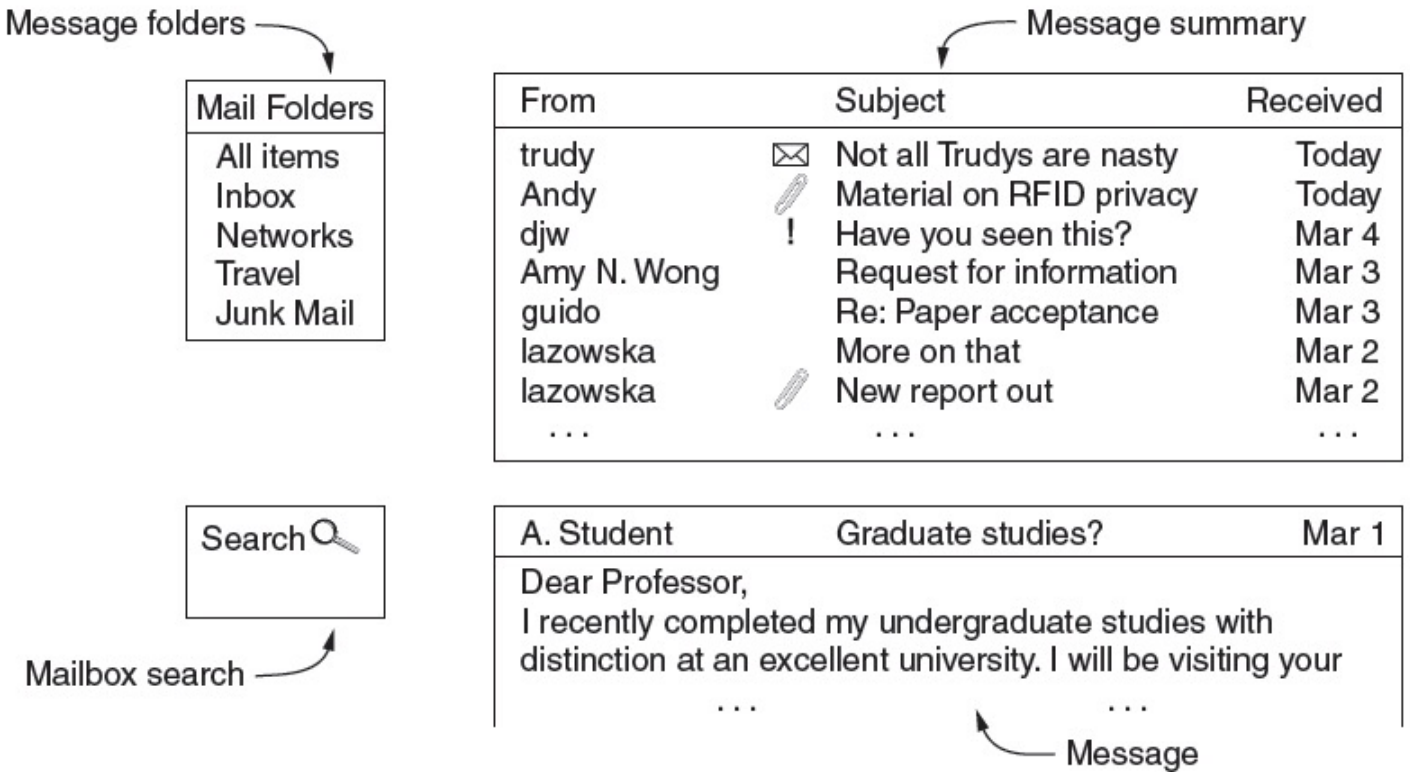
- The term "*user agent*" refers to the client software that sends and retrieves email
- The term "*transfer agent*" refers to the software that collects the sender's email and delivers it to another transfer agent where the recipient can read it



Architecture of the email system

User Agents

- Many different user agents are available, but all have a similar set of basic features



Email

- *RFC 822* defined email messages to have two parts: a *header* and a *body*
  - Email was originally expected to be text messages and both the header and body are ASCII text
  - This is still the case, although RFC 822 has been augmented by the *Multipurpose Internet Mail Extensions* (MIME) protocol to allow the message body to carry all types of data
  - MIME encodes binary data into the range of ASCII characters so that attachments and images can be included in an email message without changing the text-based nature of email

Email Message Formats

The message header consists of lines of text terminated by *CarriageReturnLineFeed* (CRLF)

- Each header line contains a *type* and *value* separated by a colon "**:**"
- The header is separated from the message body by a blank line
- RFC 822 was extended by MIME to allow email messages to carry many different types of data: audio, video, images, PDF documents, and so on

Email Message Formats

- Header fields are readable ASCII text
- The fields below are for *message transport*

Header	Meaning
To:	Email address(es) of primary recipient(s)
Cc:	Email address(es) of secondary recipient(s)
Bcc:	Email address(es) for blind carbon copies
From:	Person or people who created the message
Sender:	Email address of the actual sender
Received:	Line added by each transfer agent along the route
Return-Path:	Can be used to identify a path back to the sender

Email Message Formats

- These fields are used by *user agents*
  - Consequentially, they are not used to deliver messages and may not match the previous fields

Header	Meaning
Date:	The date and time the message was sent
Reply-To:	Email address to which replies should be sent
Message-Id:	Unique number for referencing this message later
In-Reply-To:	Message-Id of the message to which this is a reply
References:	Other relevant Message-Ids
Keywords:	User-chosen keywords
Subject:	Short summary of the message for the one-line display

After the TCP connection is established

```
220 mail0.company.com0 ESMTP Postfix (Enterprise Network Mail Server)
EHLO worker0-0
250-mail0.company.com0
MAIL FROM:<worker0-0@mail0.company.com0>
250 2.1.0 Ok
RCPT TO:<worker2-2@mail0.company.com0>
250 2.1.5 Ok
DATA
354 End data with <CR><LF>.<CR><LF>
Date: Tue, 19 Jan 2016 18:00:38 -0500 (EST)
From: worker0-0@mail0.company.com0
To: worker2-2@mail0.company.com0
Message-ID: <1338576172.94.1453244438408@worker0-0>
Subject: Requirements
MIME-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit

Here is an insight to our current requirements.
.
250 2.0.0 Ok: queued as 5BE21209C5
QUIT
221 2.0.0 Bye
```

Messages from the server

Messages from the user agent (client)

ESMTP is Extended SMTP

EHLO is the HELLO message sent by the client to the server to begin the exchange

Period on a line by itself indicates end of text

MIME

MIME consists of three basic components:

- **First**: a set of header lines that extend RFC 822
  - *MIME-Version*: (the version of MIME being used)
  - *Content-Description*: (what's in the message)
  - *Content-Type*: (the type of data contained in the message)
  - *Content-Id:* a unique identifier for the content
  - *Content-Transfer-Encoding* (how the data is encoded)
- **Second**: definitions for a set of *content types*
  - For example, MIME defines two different still image types: image/gif and image/jpeg to handle both kinds of image
- **Third**: a way to *encode* the various data types so they can be included in an ASCII email message

MIME

MIME supports a variety of common media and data types

Type	Example subtypes	Description
text	plain, html, xml, css	Text in various formats
image	gif, jpeg, tiff	Pictures
audio	basic, mpeg, mp4	Sounds
video	mpeg, mp4, quicktime	Movies
model	vrml	3D model
application	octet-stream, pdf, javascript, zip	Data produced by applications
message	http, rfc822	Encapsulated message
multipart	mixed, alternative, parallel, digest	Combination of multiple types

An Email with MIME Data

Putting it all together: a multipart message containing HTML and audio alternatives

```
From: alice@cs.washington.edu
To: bob@ee.uwa.edu.au
MIME-Version: 1.0
Message-Id: <0704760941.AA00747@cs.washington.edu>
Content-Type: multipart/alternative; boundary=qwertyuiopasdfghjklzxcvbnm
Subject: Earth orbits sun integral number of times

This is the preamble. The user agent ignores it. Have a nice day.

--qwertyuiopasdfghjklzxcvbnm
Content-Type: text/html

<p>Happy birthday to you<br>
Happy birthday to you<br>
Happy birthday dear <b> Bob </b><br>
Happy birthday to you</p>

--qwertyuiopasdfghjklzxcvbnm
Content-Type: message/external-body;
        access-type="anon-ftp";
        site="bicycle.cs.washington.edu";
        directory="pub";
        name="birthday.snd"

content-type: audio/basic
content-transfer-encoding: base64
--qwertyuiopasdfghjklzxcvbnm--
```

One part (HTML)

ASCII text (identified as HTML)

Another (audio)

MIME header fields

Email Message Transfer

Simple Mail Transfer Protocol (SMTP) moves email between servers (i.e., Transfer Agents) to deliver it to the user's local email server

- There is a program running (as a daemon) on the email server that handles these transfers, hence it is called the Message Transfer Agent (MTA)
- Email transfer is over TCP to provide reliable delivery
- SMTP is not normally used to deliver email to the user's computer, there are many user agents for that
- Depending on the location and topology, email may be transferred directly to the recipient's server or may be relayed across multiple servers before delivery

Email Message Transfer

- MTAs typically transfer email in bulk and can buffer email in cases where it is unable to move it to the next MTA along the path
- If delivery fails or cannot occur, MTAs will wait and attempt to resend the email later
- In this way, MTA are similar to routers except that MTAs will buffer undelivered email much longer and retry delivery for a matter of days
- Transfer between MTAs is TCP on port 25
  - The port that user agents employ depends on their user agent protocol

Email Message Transfer

- Modern MTAs can use SMTP extensions to deal with newer options, such as:
  - authentication, binary attachments in MIME format, secure connections (using TLS encryption) and addresses that include international character sets

Keyword	Description
AUTH	Client authentication
BINARYMIME	Server accepts binary messages
CHUNKING	Server accepts large messages in chunks
SIZE	Check message size before trying to send
STARTTLS	Switch to secure transport (TLS; see Chap. 8)
UTF8SMTP	Internationalized addresses

Email Readers (User Agents)

- There are a variety of email readers and they normally conform to a specific protocol
- POP (Post Office Protocol) (latest is POP3)
  - POP originally connected to the server and downloaded all emails to the client machine, deleting them from the server
  - Newer versions allow leaving email on the server
- IMAP (Internet Message Access Protocol)
  - IMAP uses a client-server model to access email on the server by exchanging commands
  - IMAP uses a number of commands to send and receive emails and manage email on the server

Email Readers (User Agents)

- Both POP and IMAP can use encryption to make a secure connection to the server and support authentication of users
  - POP uses port 110 for unencrypted connections and port 995 for TLS or SSL connections
  - IMAP uses port 143 for unencrypted connections and port 993 for TLS or SSL connections
- IMAP is more flexible than POP, allowing multiple connections to the same server, supports organizing email in folders and has complex search features

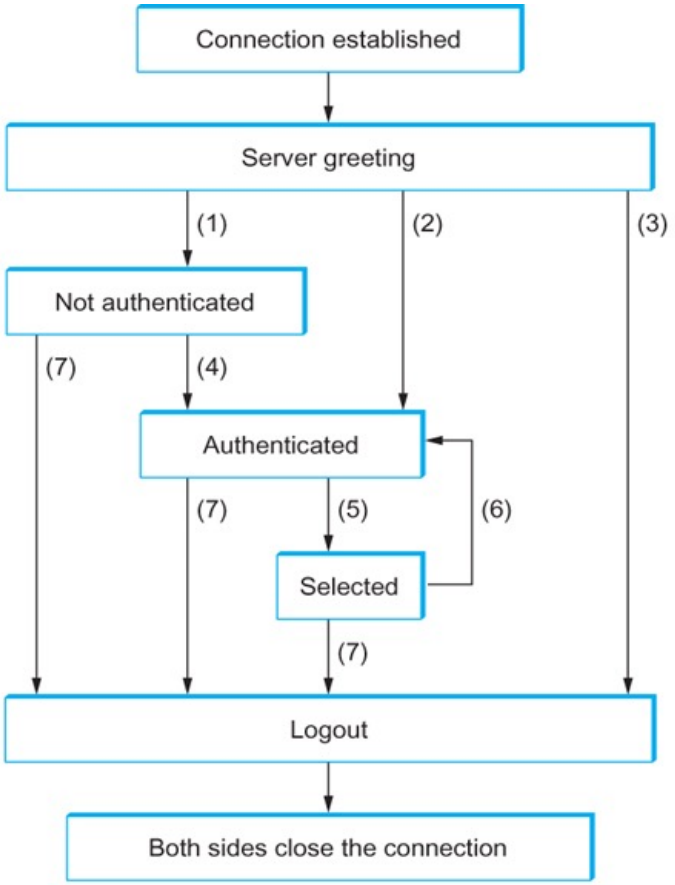
IMAP Commands

An IMAP client and email server exchange commands

- for session login/logout
- to send or read email
- to search for email
- to manage the email collection on the server

Command	Description
CAPABILITY	List server capabilities
STARTTLS	Start secure transport (TLS; see Chap. 8)
LOGIN	Log on to server
AUTHENTICATE	Log on with other method
SELECT	Select a folder
EXAMINE	Select a read-only folder
CREATE	Create a folder
DELETE	Delete a folder
RENAME	Rename a folder
SUBSCRIBE	Add folder to active set
LIST	List the available folders
LSUB	List the active folders
STATUS	Get the status of a folder
APPEND	Add a message to a folder
CHECK	Get a checkpoint of a folder
FETCH	Get messages from a folder
SEARCH	Find messages in a folder
STORE	Alter message flags
COPY	Make a copy of a message in a folder
EXPUNGE	Remove messages flagged for deletion
UID	Issue commands using unique identifiers
NOOP	Do nothing
CLOSE	Remove flagged messages and close folder
LOGOUT	Log out and close connection

IMAP State Transitions



1. Connection before authentication (OK Greeting)
2. Preauthentication (PREAUTH)
3. Rejected Connection (BYE)
4. Successful LOGIN or AUTHENTICATION
5. Successful SELECT or EXAMINE
6. CLOSE or failed SELECT
7. LOGOUT (server is shutdown or connection is closed)