CSMC 417

Computer Networks

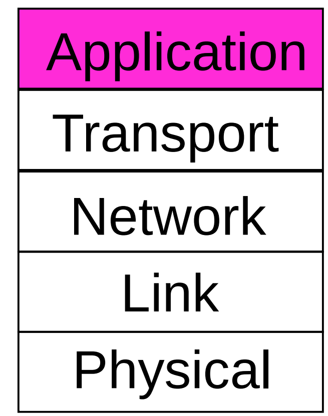
Prof. Ashok K Agrawala

Set 8

The Application Layer

The Application Layer

Uses transport services to build distributed applications



Application-Layer Protocols

- Network applications run on end systems
  - They depend on the network to provide a service
  - … but cannot run software on the network elements
- Network applications run on multiple machines
  - Different end systems communicate with each other
  - Software is often written by multiple parties
- Leading to a need to explicitly define a protocol
  - Types of messages (e.g., requests and responses)
  - Message syntax (e.g., fields, and how to delineate)
  - Semantics of the fields (i.e., meaning of the information)
  - Rules for when and how a process sends messages

Application vs. Application-Layer Protocols

- Application-layer protocol is just one piece
  - Defining how the end hosts communicate
- Example: World Wide Web
  - HyperText Transfer Protocol is the protocol
  - But the Web includes other components, such as document formats (HTML), Web browsers, servers, …
- Example: electronic mail
  - Simple Mail Transfer Protocol (SMTP) is the protocol
  - But e-mail includes other components, such as mail servers, user mailboxes, mail readers

Protocols Tailored to the Application

- Telnet: interacting with account on remote machine
  - Client simply relays user keystrokes to the server
  - … and server simply relays any output to the client
  - TCP connection persists for duration of the login session
  - Network Virtual Terminal format for transmitting ASCII data, and control information (e.g., End-of-Line delimiter)
- FTP: copying files between accounts
  - Client connects to remote machine, “logs in, and issues commands for transferring files to/from the account
  - … and server responds to commands and transfers files
  - Separate TCP connections for control and data
  - Control connection uses same NVT format as Telnet

Protocols Tailored to the Application

- SMTP: sending e-mail to a remote mail server
  - Sending mail server transmits e-mail message to a mail server running on a remote machine
  - Each server in the path adds its identifier to the message
  - Single TCP connection for control and data
  - SMTP replaced the earlier use of FTP for e-mail
- HTTP: satisfying requests based on a global URL
  - Client sends a request with method, URL, and meta-data
  - … and the server applies the request to the resource and returns the response, including meta-data
  - Single TCP connection for control and data

Comparing the Protocols

- Commands and replies
  - Telnet sends commands in binary, whereas the other protocols are text based
  - Many of the protocols have similar request methods and response codes
- Data types
  - Telnet, FTP, and SMTP transmit text data in standard U.S. 7-bit ASCII
  - FTP also supports transfer of data in binary form
  - SMTP uses MIME standard for sending non-text data
  - HTTP incorporates some key aspects of MIME (e.g., classification of data formats)

Comparing the Protocols (Continued)

- Transport
  - Telnet, FTP, SMTP, and HTTP all depend on reliable transport protocol
  - Telnet, SMTP, and HTTP use a single TCP connection
  - … but FTP has separate control and data connections
- State
  - In Telnet, FTP, and SMTP, the server retains information about the session with the client
  - E.g., FTP server remembers client's current directory
  - In contrast, HTTP servers are stateless

Reflecting on Application-Layer Protocols

- Protocols are tailored to the applications
  - Each protocol is customized to a specific need
- Protocols have many key similarities
  - Each new protocol was influenced by the previous ones
  - New protocols commonly borrow from the older ones
- Protocols depend on same underlying substrate
  - Ordered reliable stream of bytes (i.e., TCP)
  - Domain Name System (DNS)
- Relevance of the protocol standards process
  - Important for interoperability across implementations
  - Yet, not necessary if same party writes all of the software
  - …which is increasingly common (e.g., P2P software)

Domain Name System (DNS)

Topics

- Computer science concepts underlying DNS
  - Indirection: names in place of addresses
  - Hierarchy: in names, addresses, and servers
  - Caching: of mappings from names to/from addresses
- Inner-workings of DNS
  - DNS resolvers and servers
  - Iterative and recursive queries
  - TTL-based caching
- Web and DNS
  - Influence of DNS queries on Web performance
  - Server selection and load balancing

DNS – Domain Name System

The DNS resolves high-level human readable names for computers to low-level IP addresses

- DNS name space »
- Domain Resource records »
- Name servers »

Host Names vs. IP addresses

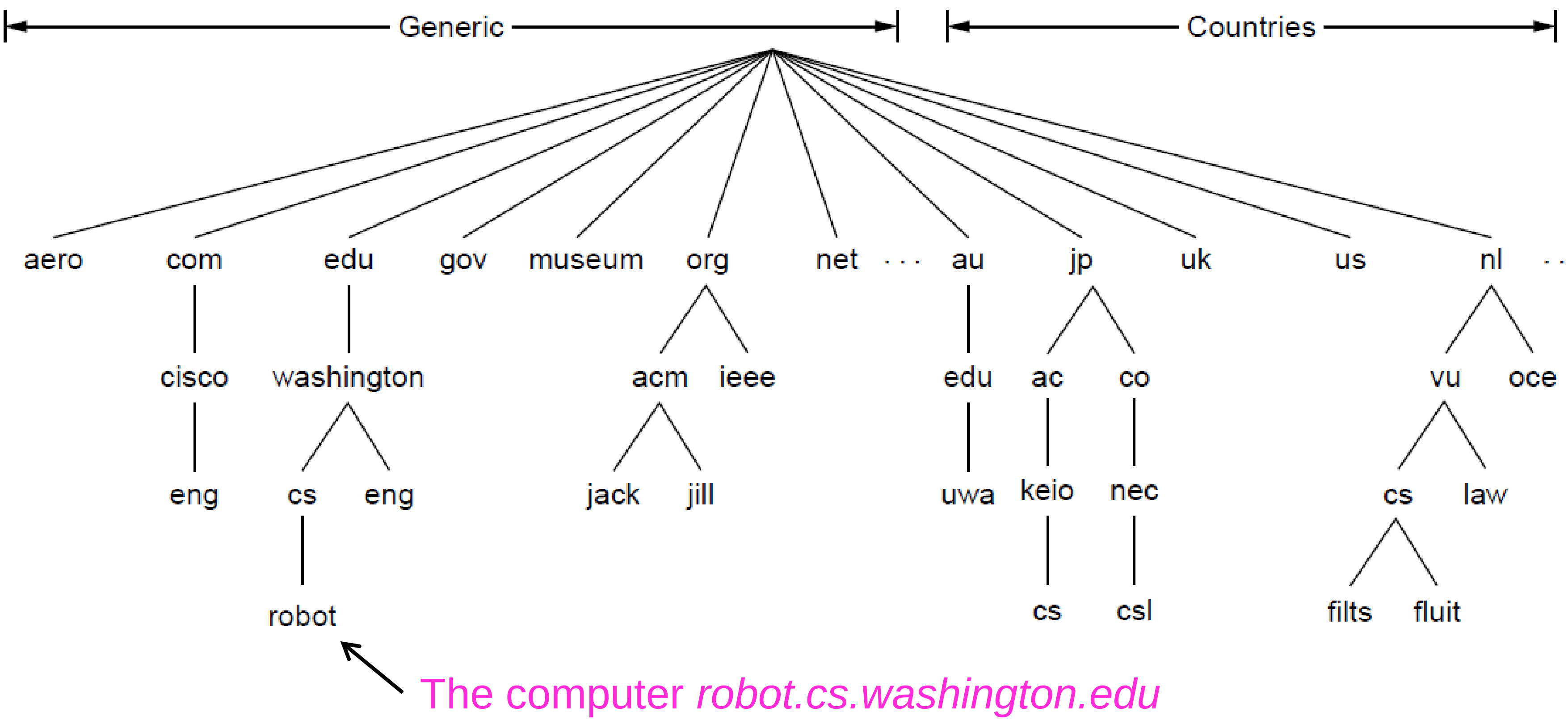
- Host names
  - Mnemonic name appreciated by humans
  - Variable length, alpha-numeric characters
  - Provide little (if any) information about location
  - Examples: www.cnn.com and ftp.eurocom.fr
- IP addresses
  - Numerical address appreciated by routers
  - Fixed length, binary number
  - Hierarchical, related to host location
  - Examples: 64.236.16.20 and 193.30.227.161

Separating Naming and Addressing

- Names are easier to remember
  - www.cnn.com vs. 64.236.16.20
- Addresses can change underneath
  - Move www.cnn.com to 64.236.16.20
  - E.g., renumbering when changing providers
- Name could map to multiple IP addresses
  - www.cnn.com to multiple replicas of the Web site
- Map to different addresses in different places
  - Address of a nearby copy of the Web site
  - E.g., to reduce latency, or return different content
- Multiple names for the same address
  - E.g., aliases like ee.mit.edu and cs.mit.edu

The DNS Name Space (1)

DNS namespace is hierarchical from the root down



The DNS Name Space (2)

Generic top-level domains are controlled by ICANN who appoints registrars to run them

Domain	Intended use	Start date	Restricted?
com	Commercial	1985	No
edu	Educational institutions	1985	Yes
gov	Government	1985	Yes
int	International organizations	1988	Yes
mil	Military	1985	Yes
net	Network providers	1985	No
org	Non-profit organizations	1985	No
aero	Air transport	2001	Yes
biz	Businesses	2001	No
coop	Cooperatives	2001	Yes
info	Informational	2002	No
museum	Museums	2002	Yes
name	People	2002	No
pro	Professionals	2002	Yes
cat	Catalan	2005	Yes
jobs	Employment	2005	Yes
mobi	Mobile devices	2005	Yes
tel	Contact details	2005	Yes
travel	Travel industry	2005	Yes
xxx	Sex industry	2010	No

This one was controversial

Resource Record

Domain Name   Time-to-live   Class Type Value

- Domain Name – Domain to which this record applies
- Time-To-Live – Indication of how stable the record is – 86400 – seconds/day
- Class – For Internet information it is always IN
- Type –
- Value -

DNS Resource Records

DNS: distributed db storing resource records (RR)

RR format: (name, value, type, ttl)

- Type=A
  - name is hostname
  - value is IP address

- Type=NS
  - **name** is domain (e.g. foo.com)
  - **value** is hostname of authoritative name server for this domain

- Type=CNAME
  - name is alias name for some "canonical" (the real) name

    www.ibm.com is really servereast.backup2.ibm.com
  - value is canonical name

- Type=MX
  - value is name of mailserver associated with name

Domain Resource Records (1)

The key resource records in the namespace are IP addresses (A/AAAA) and name servers (NS), but there are others too (e.g.,

Type	Meaning	Value
SOA	Start of authority	Parameters for this zone
A	IPv4 address of a host	32-Bit integer
AAAA	IPv6 address of a host	128-Bit integer
MX	Mail exchange	Priority, domain willing to accept email
NS	Name server	Name of a server for this domain
CNAME	Canonical name	Domain name
PTR	Pointer	Alias for an IP address
SPF	Sender policy framework	Text encoding of mail sending policy
SRV	Service	Host that provides it
TXT	Text	Descriptive ASCII text

Domain Resource Records (2)

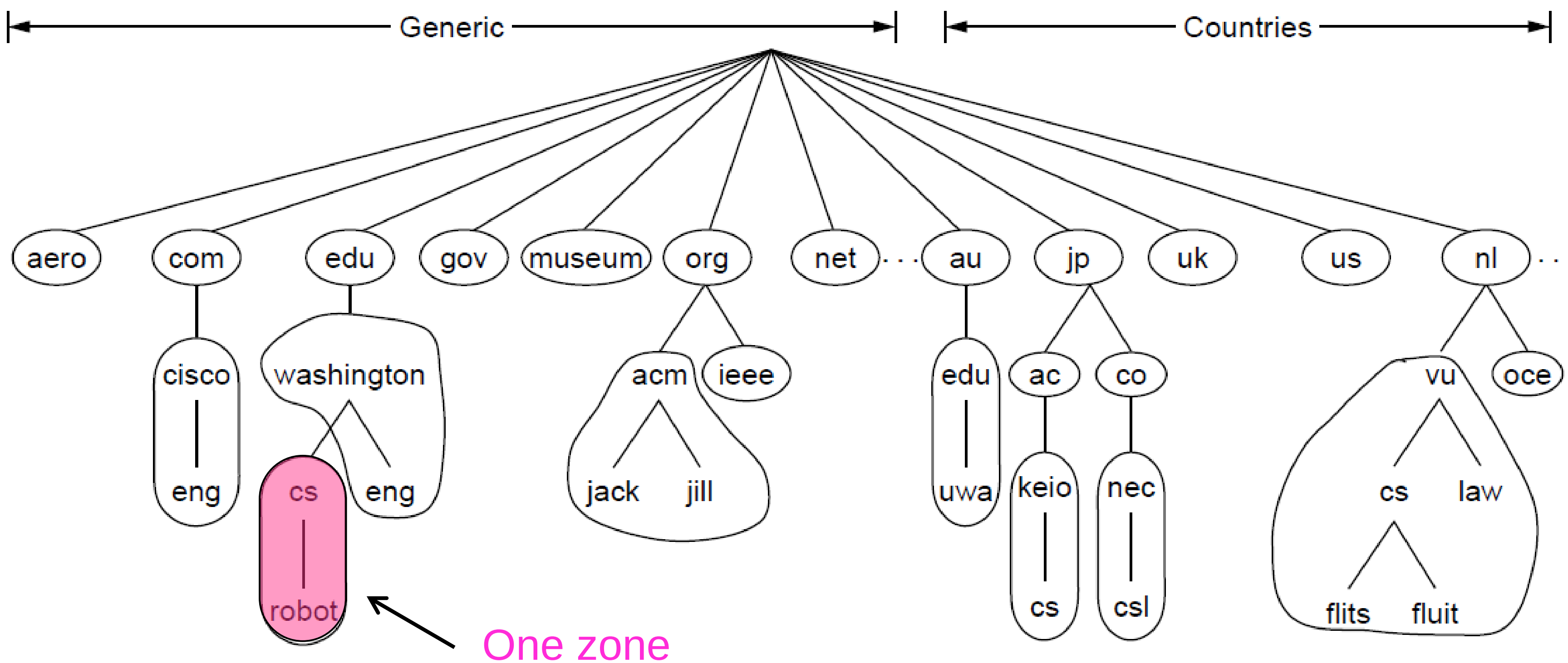
; Authoritative data for cs.vu.nl

cs.vu.nl.	86400	IN	SOA	star boss (9527,7200,7200,241920,86400)	
cs.vu.nl.	86400	IN	MX	1 zephyr	
cs.vu.nl.	86400	IN	MX	2 top	← Name server
cs.vu.nl.	86400	IN	NS	star	
star	86400	IN	A	130.37.56.205	
zephyr	86400	IN	A	130.37.20.10	← IP addresses of computers
top	86400	IN	A	130.37.20.11	
www	86400	IN	CNAME	star.cs.vu.nl	
ftp	86400	IN	CNAME	zephyr.cs.vu.nl	
flits	86400	IN	A	130.37.16.112	
flits	86400	IN	A	192.31.231.165	
flits	86400	IN	MX	1 flits	
flits	86400	IN	MX	2 zephyr	
flits	86400	IN	MX	3 top	
rowboat		IN	A	130.37.56.201	
		IN	MX	1 rowboat	← Mail gateways
		IN	MX	2 zephyr	
little-sister		IN	A	130.37.62.23	
laserjet		IN	A	192.31.231.216	

- A portion of a possible DNS database for cs.vu.nl.

Name Servers (1)

Name servers contain data for portions of the name space called zones (circled).



Name Servers (2)

Finding the IP address for a given hostname is called resolution and is done with the DNS protocol.

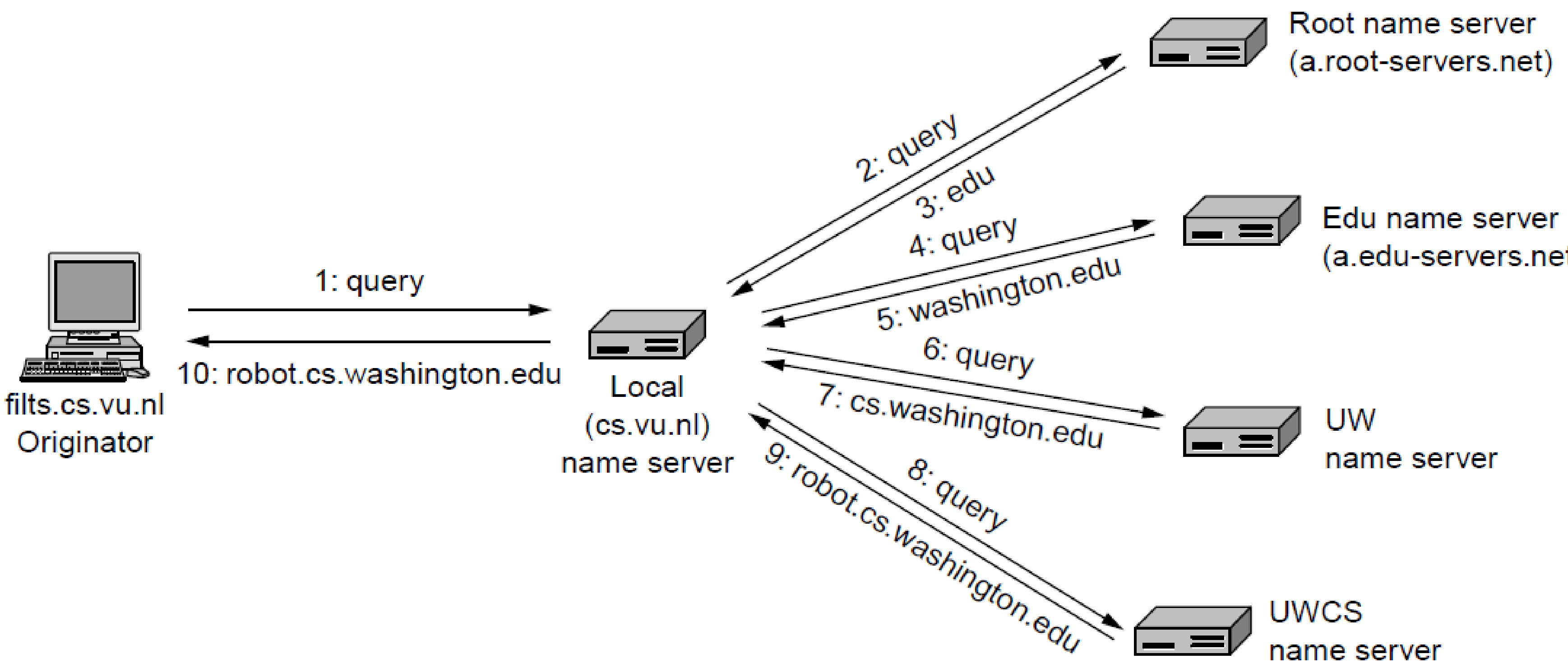
Resolution:

- Computer requests local name server to resolve
- Local name server asks the root name server
- Root returns the name server for a lower zone
- Continue down zones until name server can answer

DNS protocol:

- Runs on UDP port 53, retransmits lost messages
- Caches name server answers for better performance

Name Servers (3)



Example of a computer looking up the IP for a name

Strawman Solution: Local File

- Original name to address mapping
  - Flat namespace
  - /etc/hosts
  - SRI kept main copy
  - Downloaded regularly
- Count of hosts was increasing: moving from a machine per domain to machine per user
  - Many more downloads
  - Many more updates

Strawman Solution #2: Central Server

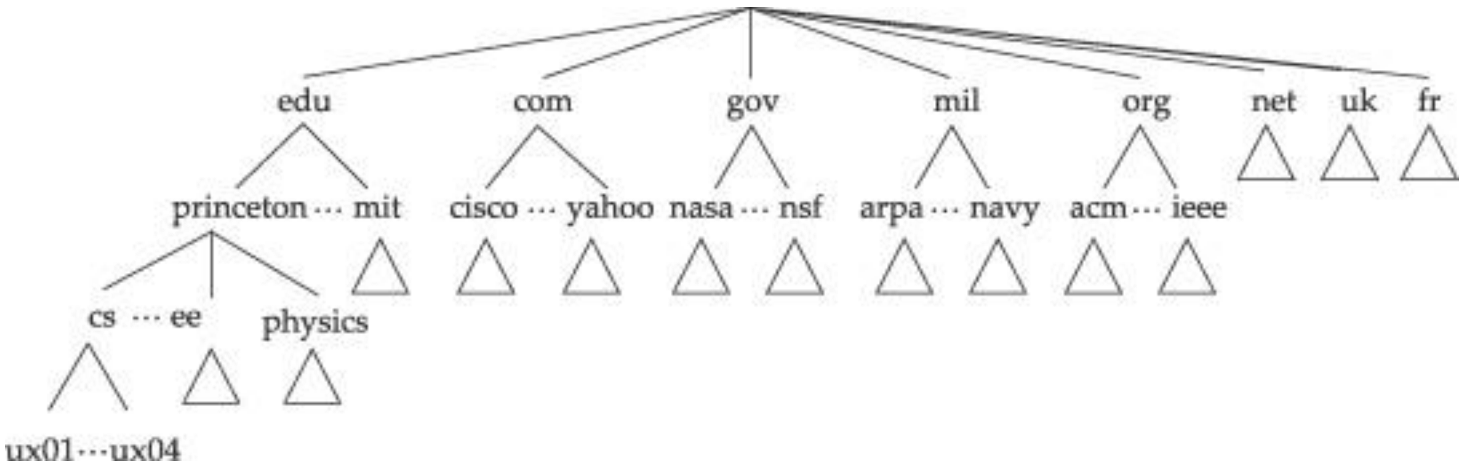
- Central server
  - One place where all mappings are stored
  - All queries go to the central server
- Many practical problems
  - Single point of failure
  - High traffic volume
  - Distant centralized database
  - Single point of update
  - Does not scale

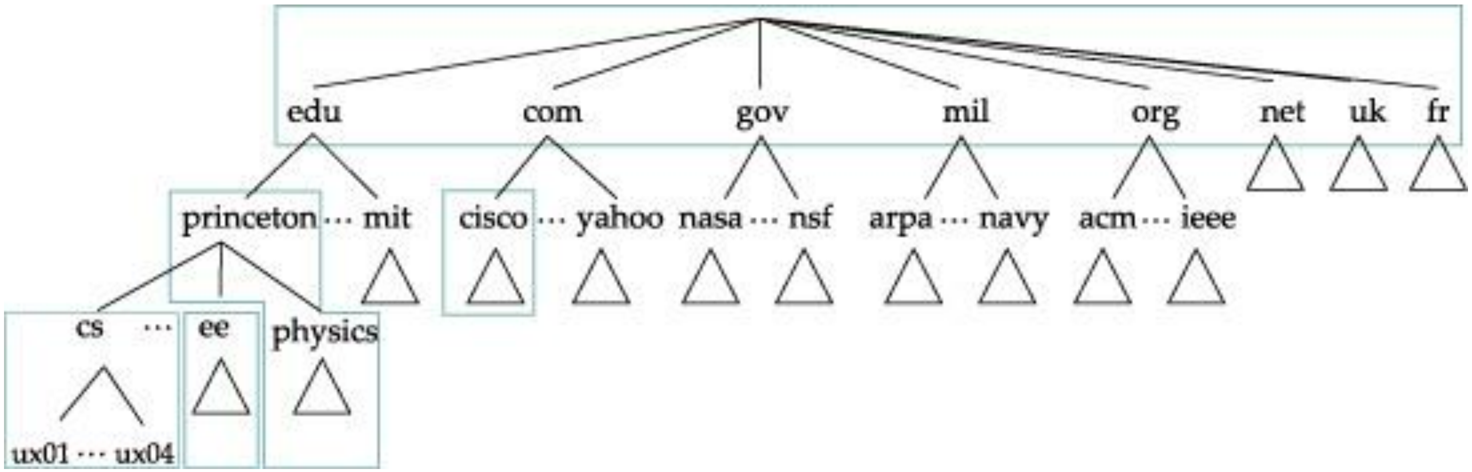
Need a distributed, hierarchical collection of servers

Domain Name System (DNS)

- Properties of DNS
  - Hierarchical name space divided into zones
  - Distributed over a collection of DNS servers
- Hierarchy of DNS servers
  - Root servers
  - Top-level domain (TLD) servers
  - Authoritative DNS servers
- Performing the translations
  - Local DNS servers
  - Resolver software

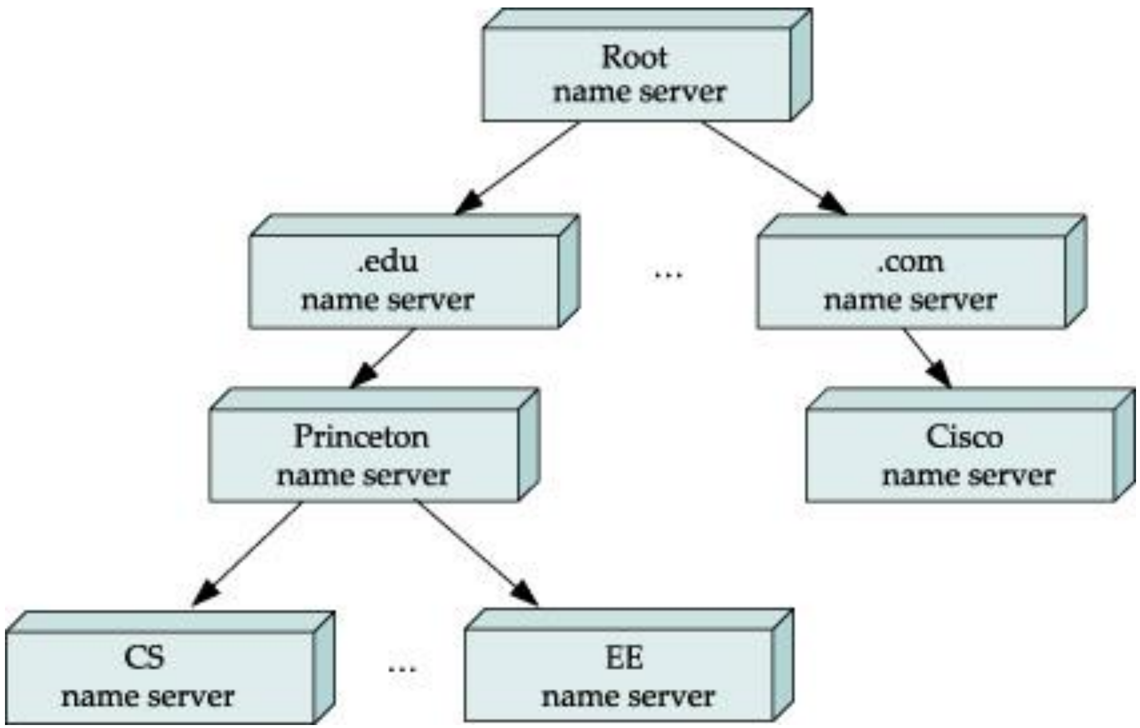
Domain Hierarchy





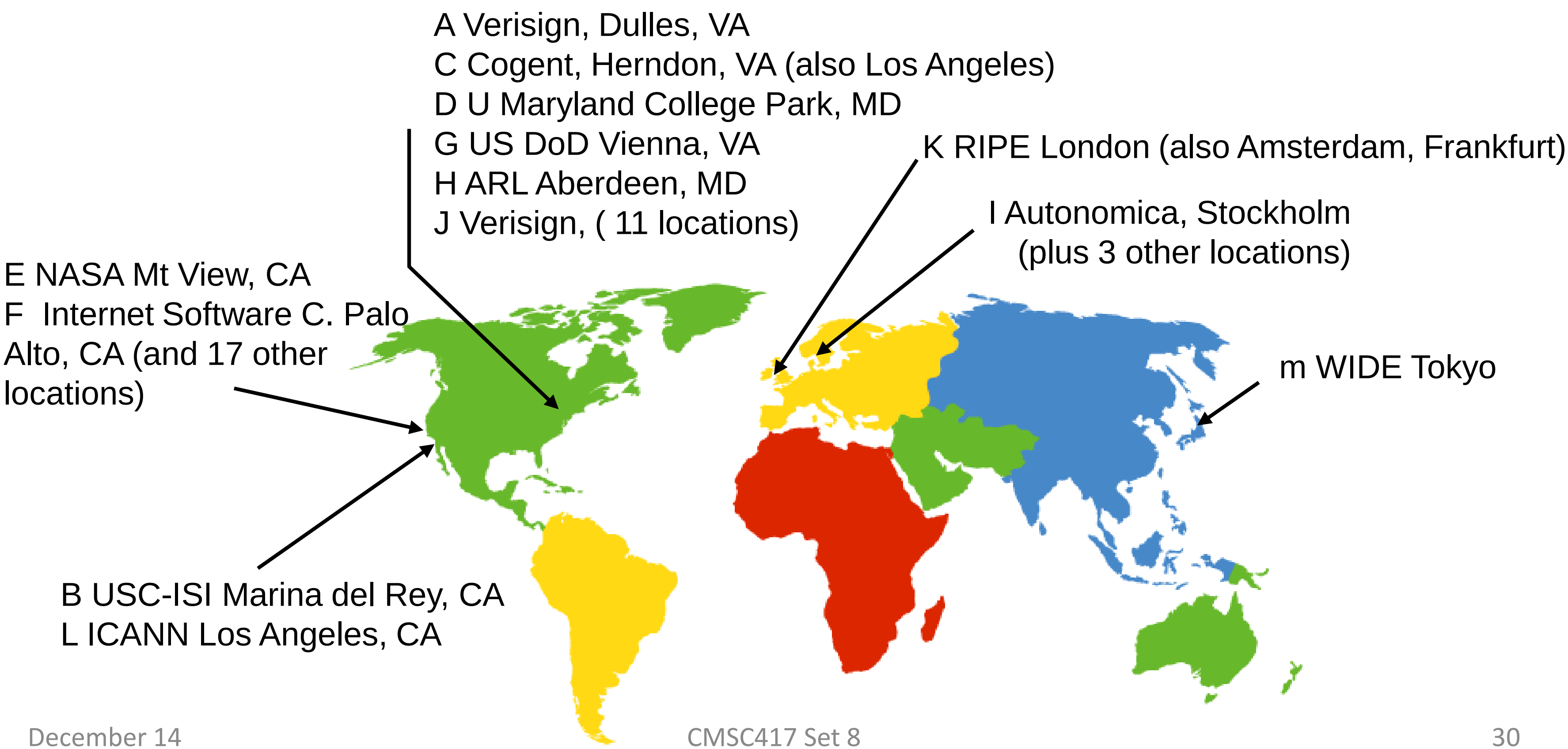
Organized in zones – unit of implementation of DNS

Hierarchy of Name Servers



DNS Root Servers

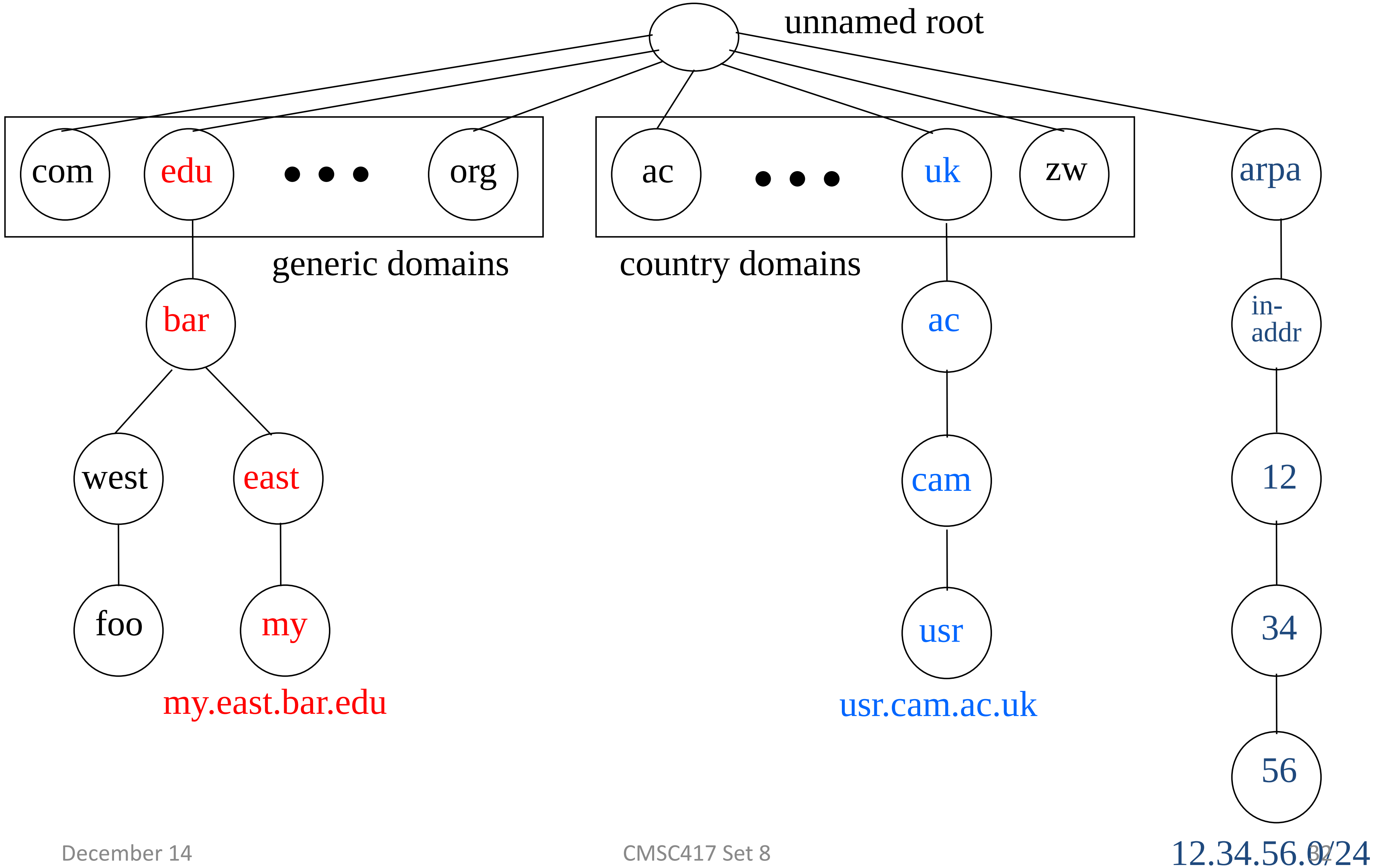
- 13 root servers (see http://www.root-servers.org/)
- Labeled A through M



TLD and Authoritative DNS Servers

- Top-level domain (TLD) servers
  - Generic domains (e.g., com, org, edu)
  - Country domains (e.g., uk, fr, ca, jp)
  - Typically managed professionally
    - Network Solutions maintains servers for “com”
    - Educause maintains servers for “edu”
- Authoritative DNS servers
  - Provide public records for hosts at an organization
  - For the organization’s servers (e.g., Web and mail)
  - Can be maintained locally or by a service provider

Distributed Hierarchical Database



Using DNS

- Local DNS server ("default name server")
  - Usually near the end hosts who use it
  - Local hosts configured with local server (e.g., /etc/resolv.conf) or learn the server via DHCP
- Client application
  - Extract server name (e.g., from the URL)
  - Do *gethostbyname()* to trigger resolver code
- Server application
  - Extract client IP address from socket
  - Optional *gethostbyaddr()* to translate into name

DNS Caching

- Performing all these queries take time
  - And all this before the actual communication takes place
  - E.g., 1-second latency before starting Web download
- Caching can substantially reduce overhead
  - The top-level servers very rarely change
  - Popular sites (e.g., www.cnn.com) visited often
  - Local DNS server often has the information cached
- How DNS caching works
  - DNS servers cache responses to queries
  - Responses include a “time to live” (TTL) field
  - Server deletes the cached entry after TTL expires

Negative Caching

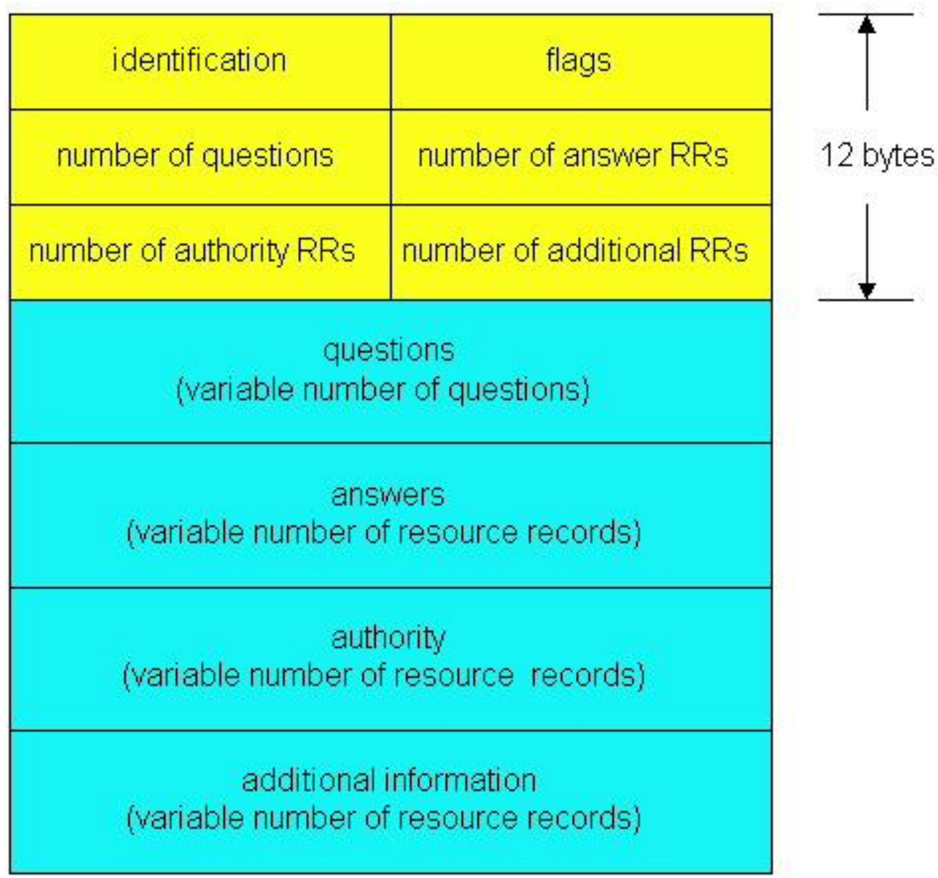
- Remember things that don't work
  - Misspellings like www.cnn.comm and www.cnnn.com
  - These can take a long time to fail the first time
  - Good to remember that they don't work
  - … so the failure takes less time the next time around

DNS Protocol

DNS protocol : *query* and *reply* messages, both with same *message format*

Message header

- Identification: 16 bit # for query, reply to query uses same #
- Flags:
  - Query or reply
  - Recursion desired
  - Recursion available
  - Reply is authoritative

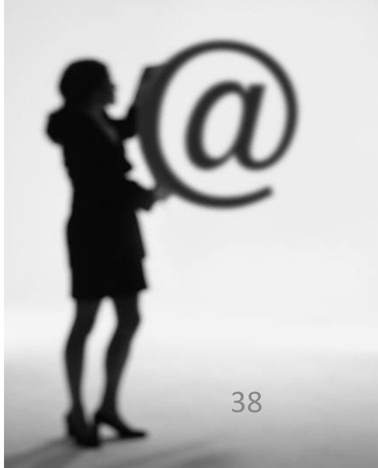


Reliability

- DNS servers are replicated
  - Name service available if at least one replica is up
  - Queries can be load balanced between replicas
- UDP used for queries
  - Need reliability: must implement this on top of UDP
- Try alternate servers on timeout
  - Exponential backoff when retrying same server
- Same identifier for all queries
  - Don’t care which server responds

Inserting Resource Records into DNS

- Example: just created startup "FooBar"
- Register foobar.com at Network Solutions
  - Provide registrar with names and IP addresses of your authoritative name server (primary and secondary)
  - Registrar inserts two RRs into the com TLD server:
    - (foobar.com, dns1.foobar.com, NS)
    - (dns1.foobar.com, 212.212.212.1, A)
- Put in authoritative server dns1.foobar.com
  - Type A record for www.foobar.com
  - Type MX record for foobar.com



DNS and the Web

DNS Query in Web Download

- User types or clicks on a URL
  - E.g., http://www.cnn.com/2006/leadstory.html
- Browser extracts the site name
  - E.g., www.cnn.com
- Browser calls gethostbyname() to learn IP address
  - Triggers resolver code to query the local DNS server
- Eventually, the resolver gets a reply
  - Resolver returns the IP address to the browser
- Then, the browser contacts the Web server
  - Creates and connects socket, and sends HTTP request

Multiple DNS Queries

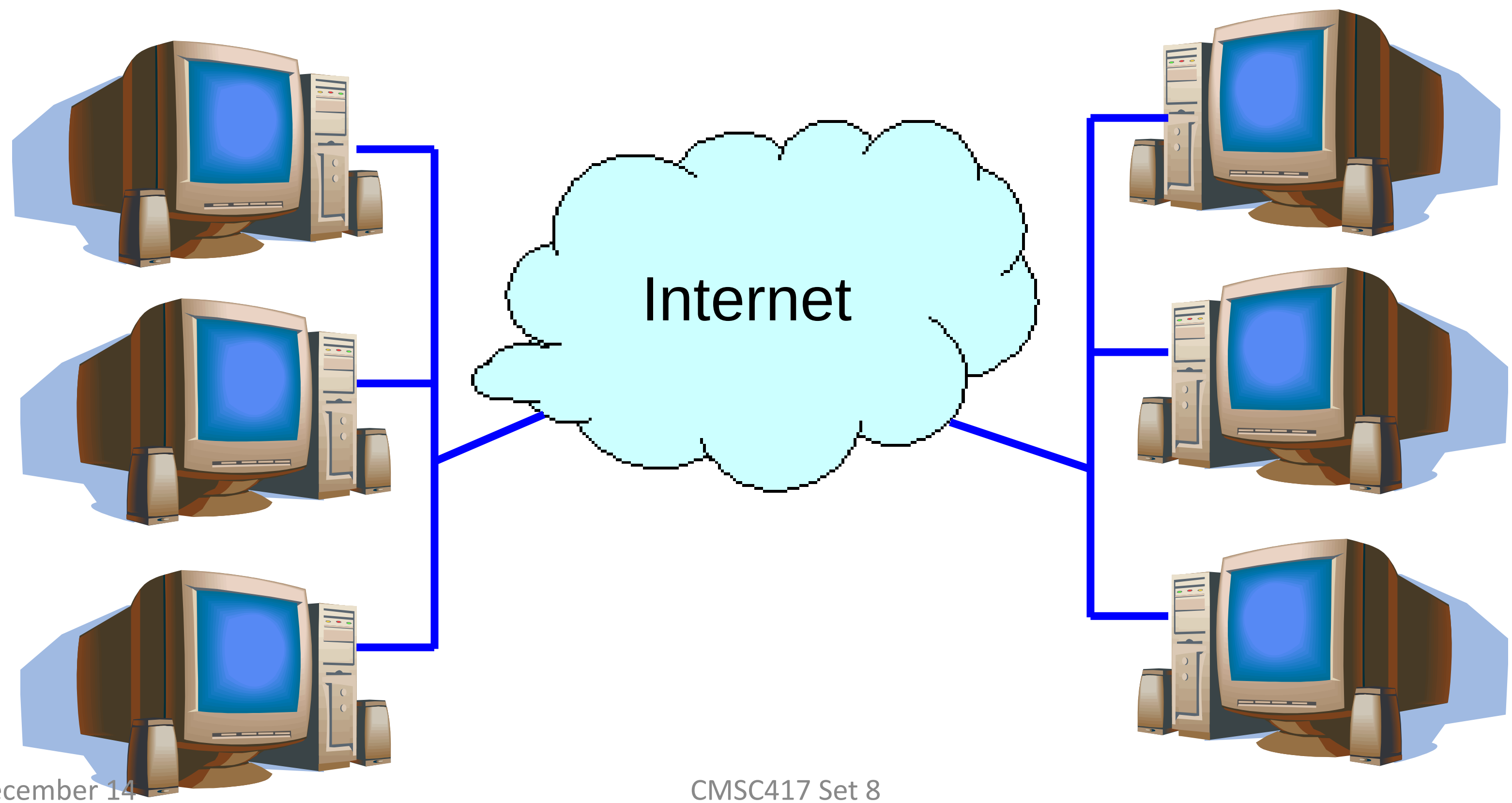
- Often a Web page has embedded objects
  - E.g., HTML file with embedded images
- Each embedded object has its own URL
  - … and potentially lives on a different Web server
  - E.g., http://www.myimages.com/image1.jpg
- Browser downloads embedded objects
  - Usually done automatically, unless configured otherwise
  - Requires learning the address for www.myimages.com

When are DNS Queries Unnecessary?

- Browser is configured to use a proxy
  - E.g., browser sends all HTTP requests through a proxy
  - Then, the proxy takes care of issuing the DNS request
- Requested Web resource is locally cached
  - E.g., cache has http://www.cnn.com/2006/leadstory.html
  - No need to fetch the resource, so no need to query
- Browser recently queried for this host name
  - E.g., user recently visited http://www.cnn.com/
  - So, the browser already called *gethostbyname()*
  - … and may be locally caching the resulting IP address

Web Server Replicas

- Popular Web sites can be easily overloaded
  - Web site often runs on multiple server machines



Directing Web Clients to Replicas

- Simple approach: different names
  - www1.cnn.com, www2.cnn.com, www3.cnn.com
  - But, this requires users to select specific replicas
- More elegant approach: different IP addresses
  - Single name (e.g., www.cnn.com), multiple addresses
  - E.g., 64.236.16.20, 64.236.16.52, 64.236.16.84, …
- Authoritative DNS server returns many addresses
  - And the local DNS server selects one address
  - Authoritative server may vary the order of addresses

Clever Load Balancing Schemes

- Selecting the “best” IP address to return
  - Based on server performance
  - Based on geographic proximity
  - Based on network load
  - …
- Example policies
  - Round-robin scheduling to balance server load
  - U.S. queries get one address, Europe another
  - Tracking the current load on each of the replicas

Challenge: What About DNS Caching?

- Problem: DNS caching
  - What if performance properties change?
  - Web clients still learning old “best” Web server
  - … until the cached information expires
- Solution: Small Time-to-Live values
  - Setting artificially small TTL values
  - … so replicas picked based on fresh information
- Disadvantages: abuse of DNS?
  - Many more DNS request/response messages
  - Longer latency in initiating the Web requests

Electronic Mail

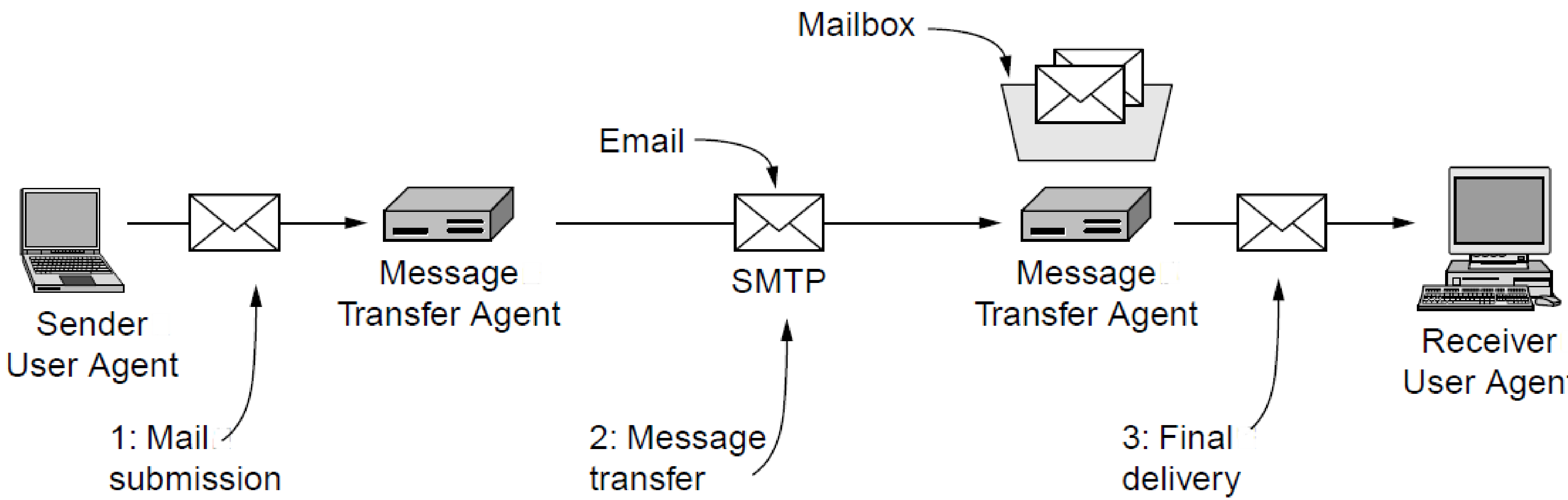
Architecture and Services

Basic functions

- Composition
- Transfer
- Reporting
- Displaying
- Disposition

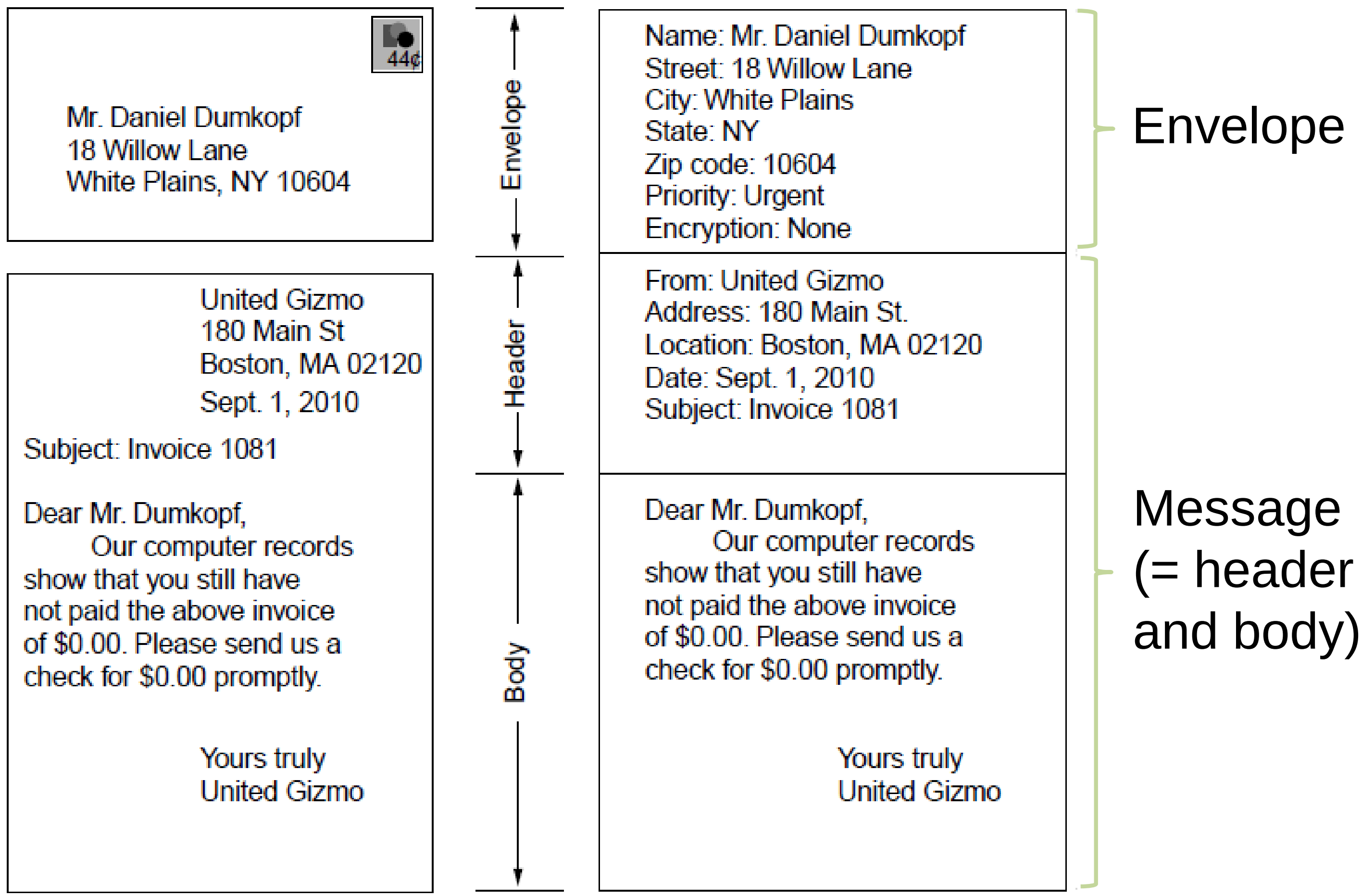
Architecture and Services (1)

- The key components and steps (numbered) to send email



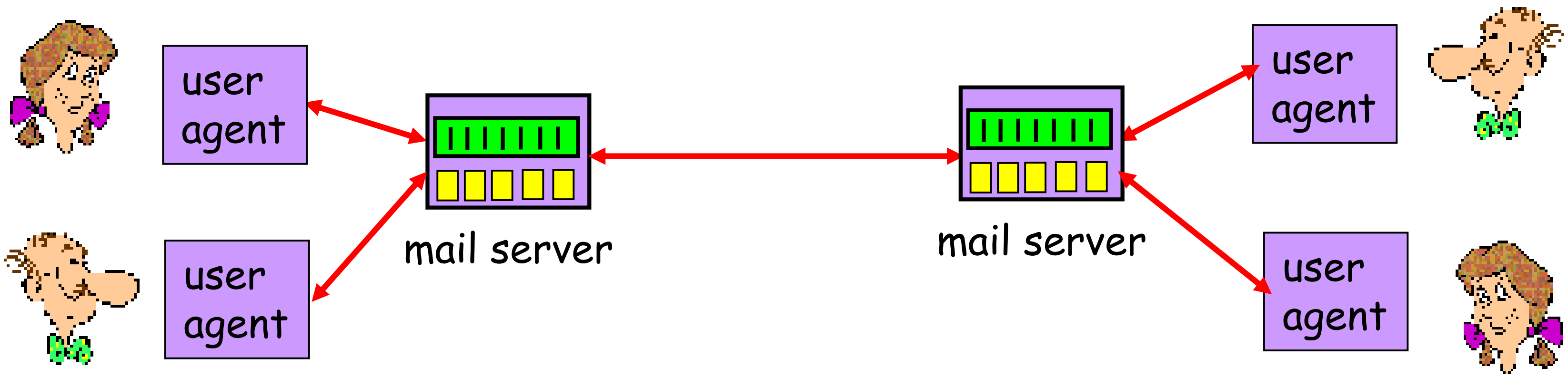
Architecture of the email system

Architecture and Services (2)



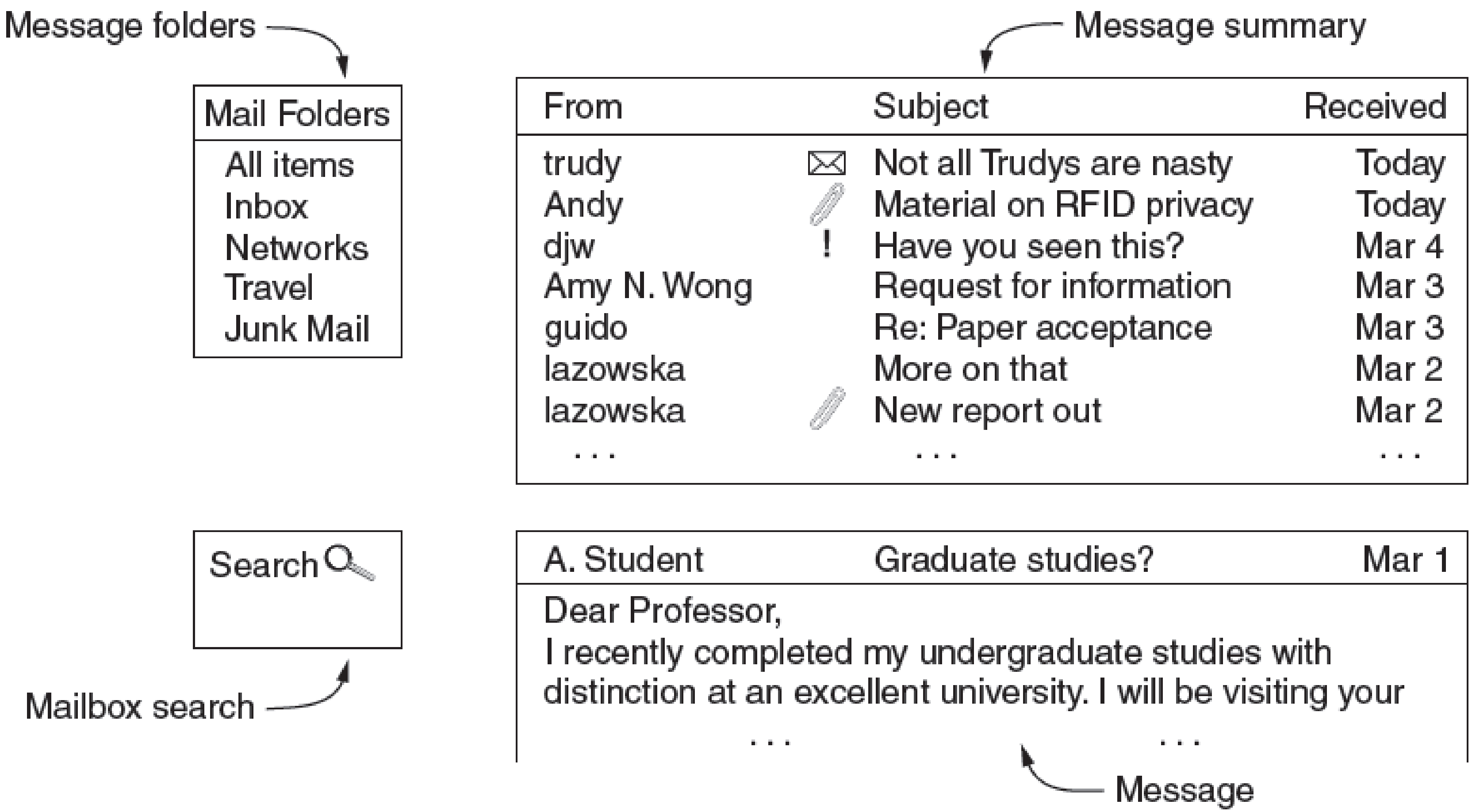
Paper mail　　Electronic mail

Mail Servers and User Agents



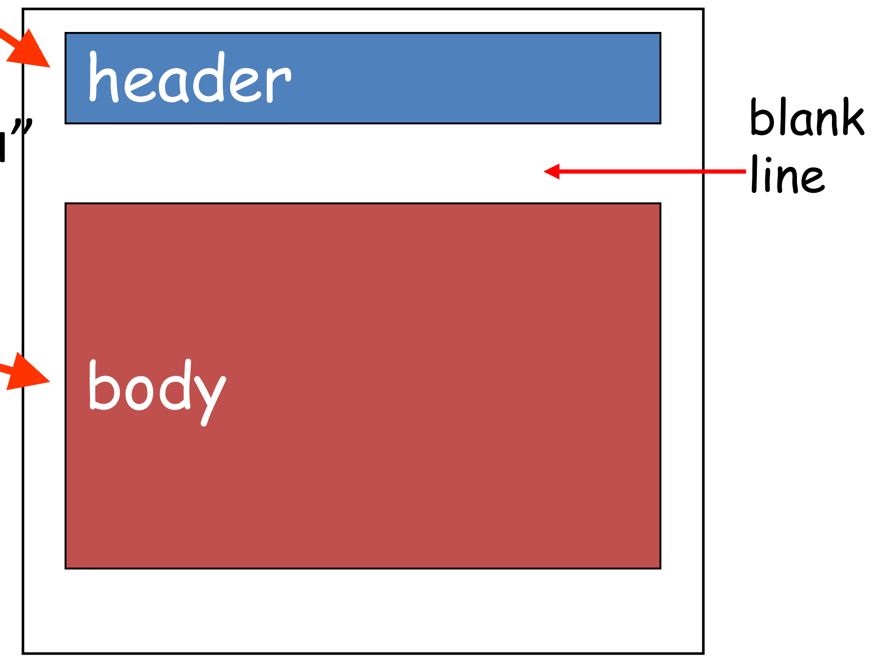
- Mail servers
  - Always on and always accessible
  - Transferring e-mail to and from other servers
- User agents
  - Sometimes on and sometimes accessible
  - Intuitive interface for the user

The User Agent



E-Mail Message

- E-mail messages have two parts
  - A header, in 7-bit U.S. ASCII text
  - A body, also represented in 7-bit U.S. ASCII text
- Header
  - Lines with "type: value"
  - "To: agrawala@cs.umd.edu"
  - "Subject: Go Terps!"
- Body
  - The text message
  - No particular structure or meaning



E-Mail Message Format (RFC 822)

- E-mail messages have two parts
  - A header, in 7-bit U.S. ASCII text
  - A body, also represented in 7-bit U.S. ASCII text
- Header
  - Series of lines ending in carriage return and line feed
  - Each line contains a type and value, separated by ":"
  - E.g., "To: jrex@princeton.edu" and "Subject: Go Tigers"
  - Additional blank line before the body begins
- Body
  - Series of text lines with no additional structure/meaning
  - Conventions arose over time (e.g., e-mail signatures)

Message Formats (1)

Header fields related to message transport; headers are readable ASCII text

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

Message Formats (2)

Other header fields useful for user agents

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

Message Formats (3)

MIME header fields used to describe what content is in the body of the message

Header	Meaning
MIME-Version:	Identifies the MIME version
Content-Description:	Human-readable string telling what is in the message
Content-Id:	Unique identifier
Content-Transfer-Encoding:	How the body is wrapped for transmission
Content-Type:	Type and format of the content

Message Formats (5)

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

One part (HTML)

Another (audio)

```
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--
```

Multipurpose Internet Mail Extensions

- Additional headers to describe the message body
  - MIME-Version: the version of MIME being used
  - Content-Type: the type of data contained in the message
  - Content-Transfer-Encoding: how the data are encoded
- Definitions for a set of content types and subtypes
  - E.g., image with subtypes gif and jpeg
  - E.g., text with subtypes plain, html, and richtext
  - E.g., application with subtypes postscript and msword
  - E.g., multipart for messages with multiple data types
- A way to encode the data in ASCII format
  - Base64 encoding, as in uuencode/uudecode

MIME – Multipurpose Internet Mail Extensions

Problems with international languages:

- Languages with accents (French, German).
- Languages in non-Latin alphabets (Hebrew, Russian).
- Languages without alphabets (Chinese, Japanese).
- Messages not containing text at all (audio or images).

Message Formats (3)

Header	Meaning
MIME-Version:	Identifies the MIME version
Content-Description:	Human-readable string telling what is in the message
Content-Id:	Unique identifier
Content-Transfer-Encoding:	How the body is wrapped for transmission
Content-Type:	Type and format of the content

Message headers added by MIME
Multipurpose Internet Mail Extensions.

MIME (3)

The MIME types and subtypes defined in RFC 2045.

Type	Subtype	Description
Text	Plain	Unformatted text
	Enriched	Text including simple formatting commands
Image	Gif	Still picture in GIF format
	Jpeg	Still picture in JPEG format
Audio	Basic	Audible sound
Video	Mpeg	Movie in MPEG format
Application	Octet-stream	An uninterpreted byte sequence
	Postscript	A printable document in PostScript
Message	Rfc822	A MIME RFC 822 message
	Partial	Message has been split for transmission
	External-body	Message itself must be fetched over the net
Multipart	Mixed	Independent parts in the specified order
	Alternative	Same message in different formats
	Parallel	Parts must be viewed simultaneously
	Digest	Each part is a complete RFC 822 message

Example: E-Mail Message Using MIME

MIME version

method used to encode data

type and subtype

encoded data

```
From: jrex@cs.princeton.edu
To: feamster@cc.gatech.edu
Subject: picture of Thomas Sweet
MIME-Version: 1.0
Content-Transfer-Encoding: base64
Content-Type: image/jpeg

base64 encoded data .....
.........................
......base64 encoded data
```

Message Transfer (1)

Messages are transferred with SMTP (Simple Mail Transfer Protocol)

- Readable text commands
- Submission from user agent to MTA on port 587
- One MTA to the next MTA on port 25
- Other protocols for final delivery (IMAP, POP3)

Message Transfer (2)

Sending a message:

- From Alice to Bob
- SMTP commands are marked [pink]

```
                    S: 220 ee.uwa.edu.au SMTP service ready
C: HELO abcd.com
                    S: 250 cs.washington.edu says hello to ee.uwa.edu.au
C: MAIL FROM: <alice@cs.washington.edu>
                    S: 250 sender ok
C: RCPT TO: <bob@ee.uwa.edu.au>
                    S: 250 recipient ok
C: DATA
                    S: 354 Send mail; end with "." on a line by itself
C: From: alice@cs.washington.edu
C: To: bob@ee.uwa.edu.au
C: MIME-Version: 1.0
C: Message-Id: <0704760941.AA00747@ee.uwa.edu.au>
C: Content-Type: multipart/alternative; boundary=qwertyuiopasdfghjklzxcvbnm
C: Subject: Earth orbits sun integral number of times
C:
C: This is the preamble. The user agent ignores it. Have a nice day.
C:
C: --qwertyuiopasdfghjklzxcvbnm
C: Content-Type: text/html
C:
C: <p>Happy birthday to you
C: Happy birthday to you
  . . . (rest of message) . . .
C: --qwertyuiopasdfghjklzxcvbnm
C: .
                    S: 250 message accepted
C: QUIT
                    S: 221 ee.uwa.edu.au closing connection
```

Message Transfer (3)

Common SMTP extensions (not in simple example)

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

E-Mail Addresses

- Components of an e-mail address
  - Local mailbox (e.g., jrex or bob.flower)
  - Domain name (e.g., cs.princeton.edu)
- Domain name is not necessarily the mail server
  - Mail server may have longer/cryptic name
    - E.g., cs.princeton.edu vs. mail.cs.princeton.edu
  - Multiple servers may exist to tolerate failures
    - E.g., cnn.com vs. atlmail3.turner.com and nycmail2.turner.com
- Identifying the mail server for a domain
  - DNS query asking for MX records (Mail eXchange)
    - E.g., nslookup –q=mx cs.princeton.edu
  - Then, a regular DNS query to learn the IP address

SMTP Store-and-Forward Protocol

- Messages sent through a series of servers
  - A server stores incoming messages in a queue
  - … to await attempts to transmit them to the next hop
- If the next hop is not reachable
  - The server stores the message and tries again later
- Each hop adds its identity to the message
  - By adding a “Received” header with its identity
  - Helpful for diagnosing problems with e-mail

Example With Received Header

Return-Path: <casado@cs.stanford.edu>
Received: from ribavirin.CS.Princeton.EDU (ribavirin.CS.Princeton.EDU [128.112.136.44])
    by newark.CS.Princeton.EDU (8.12.11/8.12.11) with SMTP id k04M5R7Y023164
    for <jrex@newark.CS.Princeton.EDU>; Wed, 4 Jan 2006 17:05:37 -0500 (EST)
Received: from bluebox.CS.Princeton.EDU ([128.112.136.38])
    by ribavirin.CS.Princeton.EDU (SMSSMTP 4.1.0.19) with SMTP id M2006010417053607946
    for <jrex@newark.CS.Princeton.EDU>; Wed, 04 Jan 2006 17:05:36 -0500
Received: from smtp-roam.Stanford.EDU (smtp-roam.Stanford.EDU [171.64.10.152])
    by bluebox.CS.Princeton.EDU (8.12.11/8.12.11) with ESMTP id k04M5XNQ005204
    for <jrex@cs.princeton.edu>; Wed, 4 Jan 2006 17:05:35 -0500 (EST)
Received: from [192.168.1.101] (adsl-69-107-78-147.dsl.pltn13.pacbell.net [69.107.78.147])
    (authenticated bits=0)
    by smtp-roam.Stanford.EDU (8.12.11/8.12.11) with ESMTP id k04M5W92018875
    (version=TLSv1/SSLv3 cipher=DHE-RSA-AES256-SHA bits=256 verify=NOT);
    Wed, 4 Jan 2006 14:05:32 -0800
Message-ID: <43BC46AF.3030306@cs.stanford.edu>
Date: Wed, 04 Jan 2006 14:05:35 -0800
From: Martin Casado <casado@cs.stanford.edu>
User-Agent: Mozilla Thunderbird 1.0 (Windows/20041206)
MIME-Version: 1.0
To: jrex@CS.Princeton.EDU
CC: Martin Casado <casado@cs.stanford.edu>
Subject: Using VNS in Class
Content-Type: text/plain; charset=ISO-8859-1; format=flowed
Content-Transfer-Encoding: 7bit

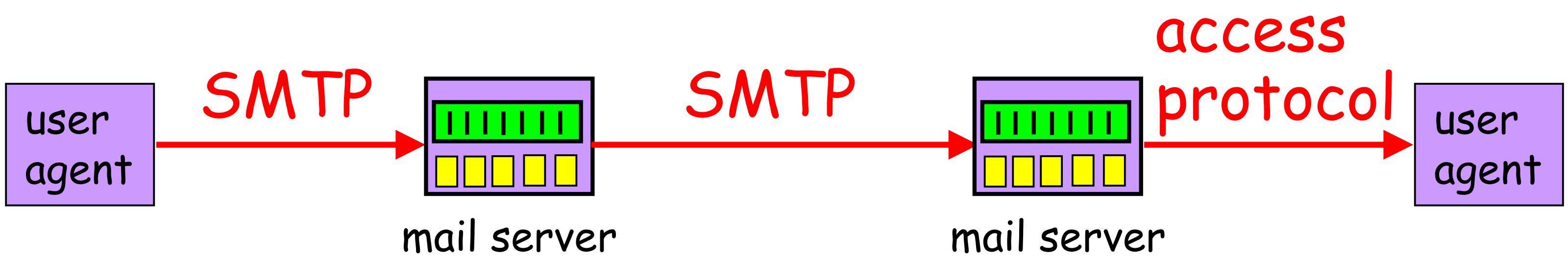
Multiple Server Hops

- Typically at least two mail servers
  - Sending and receiving sides
- May be more
  - Separate servers for key functions
    - Spam filtering
    - Virus scanning
  - Servers that redirect the message
    - From jrex@princeton.edu to jrex@cs.princeton.edu
    - Messages to princeton.edu go through extra hops
  - Electronic mailing lists
    - Mail delivered to the mailing list's server
    - … and then the list is expanded to each recipient

Electronic Mailing Lists

- Community of users reachable by one address
  - Allows groups of people to receive the messages
- Exploders
  - Explode a single e-mail message into multiple messages
  - One copy of the message per recipient
- Handling bounced messages
  - Mail may bounce for several reasons
  - E.g., recipient mailbox does not exist; resource limits
- E-mail digests
  - Sending a group of mailing-list messages at once
  - Messages delimited by boundary strings
  - … or transmitted using multiple/digest format

Simple Mail Transfer Protocol



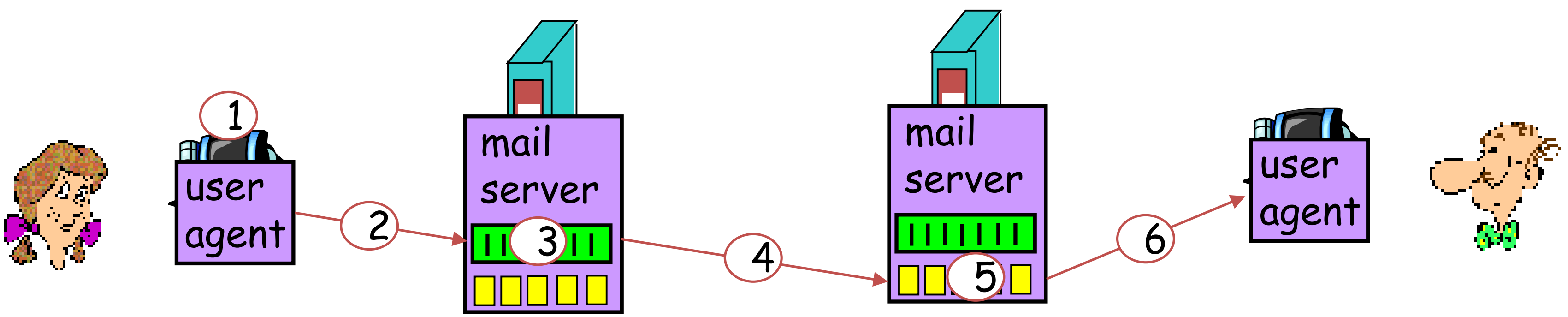
- Client-server protocol
  - Client is the sending mail server
  - Server is the receiving mail server
- Reliable data transfer
  - Built on top of TCP (on port 25)
- Push protocol
  - Sending server pushes the file to the receiving server
  - … rather than waiting for the receiver to request it

Simple Mail Transfer Protocol (Cont.)

- Command/response interaction
  - Commands: ASCII text
  - Response: three-digit status code and phrase
- Synchronous
  - Sender awaits response from a command
  - … before issuing the next command
  - Though pipelining of commands was added later
- Three phases of transfer
  - Handshaking (greeting)
  - Transfer of messages
  - Closure

Scenario: Alice Sends Message to Bob

1) Alice uses UA to compose message "to" `bob@someschool.edu`
2) Alice's UA sends message to her mail server; message placed in message queue
3) Client side of SMTP opens TCP connection with Bob's mail server
4) SMTP client sends Alice's message over the TCP connection
5) Bob's mail server places the message in Bob's mailbox
6) Bob invokes his user agent to read message



Sample SMTP interaction

```
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
```

Try SMTP For Yourself

- Running SMTP
  - Run “telnet servername 25” at UNIX prompt
  - See 220 reply from server
  - Enter HELO, MAIL FROM, RCPT TO, DATA commands
- Thinking about spoofing?
  - Very easy
  - Just forge the argument of the “FROM” command
  - … leading to all sorts of problems with spam
- Spammers can be even more clever
  - E.g., using open SMTP servers to send e-mail
  - E.g., forging the “Received” header

Retrieving E-Mail From the Server

- Server stores incoming e-mail by mailbox
  - Based on the “From” field in the message
- Users need to retrieve e-mail
  - Asynchronous from when the message was sent
  - With a way to view the message and reply
  - With a way to organize and store the messages
- In the olden days…
  - User logged on to the machine where mail was delivered
  - Users received e-mail on their main work machine

Influence of PCs on E-Mail Retrieval

- Separate machine for personal use
  - Users did not want to log in to remote machines
- Resource limitations
  - Most PCs did not have enough resources to act as a full-fledged e-mail server
- Intermittent connectivity
  - PCs only sporadically connected to the network
  - … due to dial-up connections, and shutting down of PC
  - Too unwieldy to have sending server keep trying
- Led to the creation of Post Office Protocol (POP)

Post Office Protocol (POP)

- POP goals
  - Support users with intermittent network connectivity
  - Allow them to retrieve e-mail messages when connected
  - … and view/manipulate messages when disconnected
- Typical user-agent interaction with a POP server
  - Connect to the server
  - Retrieve all e-mail messages
  - Store messages on the user’s PCs as new messages
  - Delete the messages from the server
  - Disconnect from the server
- User agent still uses SMTP to send messages

POP3 Protocol

Authorization phase

- Client commands:
  - **`user`**: declare username
  - **`pass`**: password
- Server responses
  - **`+OK`**
  - **`-ERR`**

Transaction phase, client:

- **`list`**: list message numbers
- **`retr`**: retrieve message by number
- **`dele`**: delete
- **`quit`**

```
S: +OK POP3 server ready
C: user bob
S: +OK
C: pass hungry
S: +OK user successfully logged on
```

```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: <message 1 contents>
S: .
C: dele 1
C: retr 2
S: <message 1 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

POP3

Using POP3 to fetch three messages.

```
                    S: +OK POP3 server ready
C: USER carolyn
                    S: +OK
C: PASS vegetables
                    S: +OK login successful
C: LIST
                    S: 1 2505
                    S: 2 14302
                    S: 3 8122
                    S: .
C: RETR 1
                    S: (sends message 1)
C: DELE 1
C: RETR 2
                    S: (sends message 2)
C: DELE 2
C: RETR 3
                    S: (sends message 3)
C: DELE 3
C: QUIT
                    S: +OK POP3 server disconnecting
```

Limitations of POP

- Does not handle multiple mailboxes easily
  - Designed to put user's incoming e-mail in one folder
- Not designed to keep messages on the server
  - Instead, designed to download messages to the client
- Poor handling of multiple-client access to mailbox
  - Increasingly important as users have home PC, work PC, laptop, cyber café computer, friend's machine, etc.
- High network bandwidth overhead
  - Transfers all of the e-mail messages, often well before they are read (and they might not be read at all!)

Interactive Mail Access Protocol (IMAP)

- Supports connected and disconnected operation
  - Users can download message contents on demand
- Multiple clients can connect to mailbox at once
  - Detects changes made to the mailbox by other clients
  - Server keeps state about message (e.g., read, replied to)
- Access to MIME parts of messages & partial fetch
  - Clients can retrieve individual parts separately
  - E.g., text of a message without downloading attachments
- Multiple mailboxes on the server
  - Client can create, rename, and delete mailboxes
  - Client can move messages from one folder to another
- Server-side searches
  - Search on server before downloading messages

Final Delivery (1)

- User agent uses protocol like IMAP for final delivery
  - Has commands to manipulate folders / messages [right]
- Alternatively, a Web interface (with proprietary protocol) might be used

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

A comparison of POP3 and IMAP.

Feature	POP3	IMAP
Where is protocol defined?	RFC 1939	RFC 2060
Which TCP port is used?	110	143
Where is e-mail stored?	User's PC	Server
Where is e-mail read?	Off-line	On-line
Connect time required?	Little	Much
Use of server resources?	Minimal	Extensive
Multiple mailboxes?	No	Yes
Who backs up mailboxes?	User	ISP
Good for mobile users?	No	Yes
User control over downloading?	Little	Great
Partial message downloads?	No	Yes
Are disk quotas a problem?	No	Could be in time
Simple to implement?	Yes	No
Widespread support?	Yes	Growing

Web-Based E-Mail

- User agent is an ordinary Web browser
  - User communicates with server via HTTP
  - E.g., Gmail, Yahoo mail, and Hotmail
- Reading e-mail
  - Web pages display the contents of folders
  - … and allow users to download and view messages
  - "GET" request to retrieve the various Web pages
- Sending e-mail
  - User types the text into a form and submits to the server
  - "POST" request to upload data to the server
  - Server uses SMTP to deliver message to other servers
- Easy to send anonymous e-mail (e.g., spam)

World Wide Web

Web Topics

- Main ingredients of the Web
  - URL, HTML, and HTTP
- Key properties of HTTP
  - Request-response, stateless, and resource meta-data
- Web components
  - Clients, proxies, and servers
  - Caching vs. replication
- Interaction with underlying network protocols
  - DNS and TCP
  - TCP performance for short transfers
  - Parallel connections, persistent connections, pipelining

Web History

- Before the 1970s-1980s
  - Internet used mainly by researchers and academics
  - Log in remote machines, transfer files, exchange e-mail
- Late 1980s and early 1990s
  - Initial proposal for the Web by Berners-Lee in 1989
  - Competing systems for searching/accessing documents
    - Gopher, Archie, WAIS (Wide Area Information Servers), …
    - All eventually subsumed by the World Wide Web
- Growth of the Web in the 1990s
  - 1991: first Web browser and server
  - 1993: first version of Mosaic browser

Enablers for Success of the Web

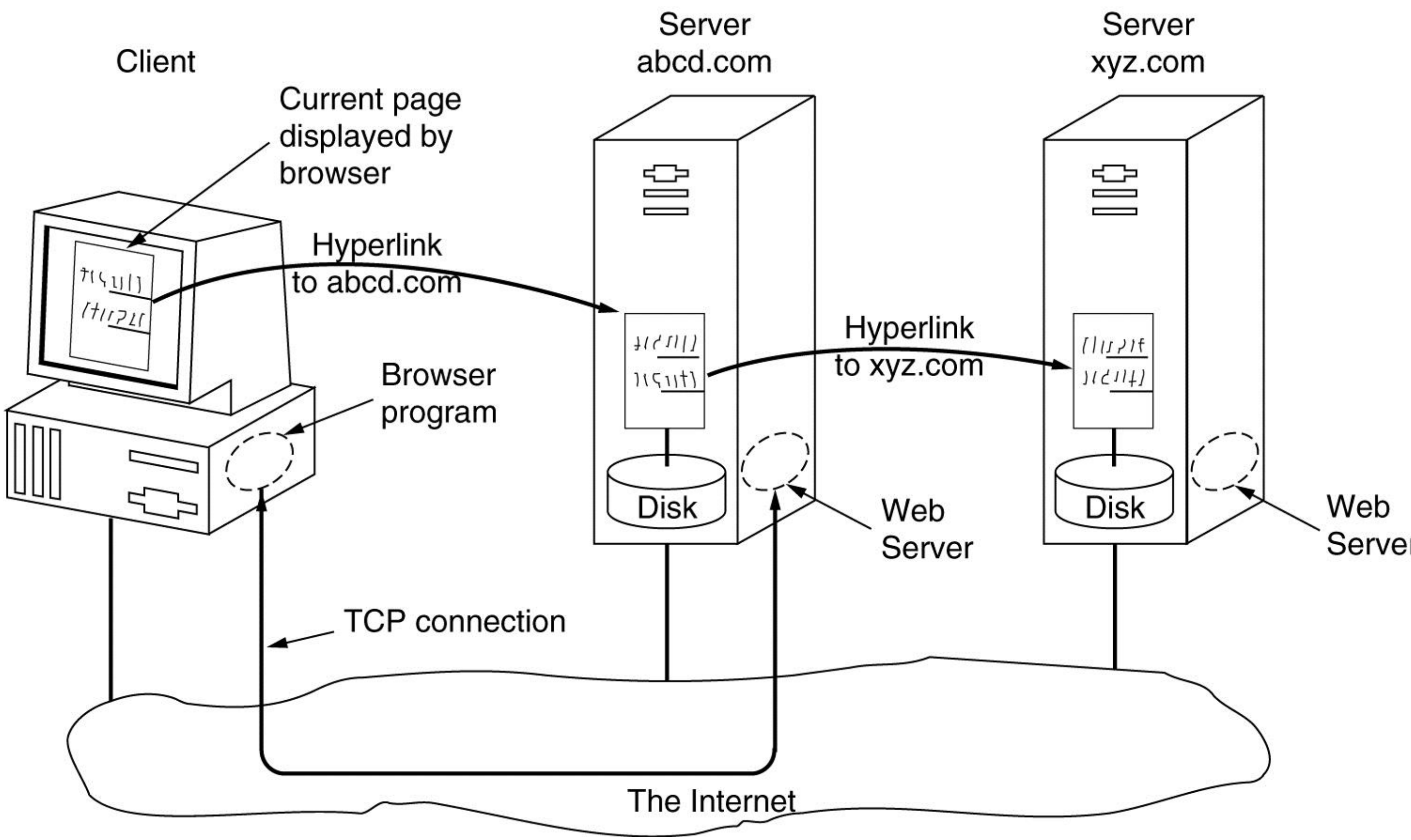
- Internet growth and commercialization
  - 1988: ARPANET gradually replaced by the NSFNET
  - Early 1990s: NSFNET begins to allow commercial traffic
- Personal computer
  - 1980s: Home computers with graphical user interfaces
  - 1990s: Power of PCs increases, and cost decreases
- Hypertext
  - 1945: Vannevar Bush’s “As We May Think”
  - 1960s: Hypertext proposed, and the mouse invented
  - 1980s: Proposals for global hypertext publishing systems

Web Components

- Clients
  - Send requests and receive responses
  - Browsers, spiders, and agents
- Servers
  - Receive requests and send responses
  - Store or generate the responses
- Proxies
  - Act as a server for the client, and a client to the server
  - Perform extra functions such as anonymization, logging, transcoding, blocking of access, caching, etc.

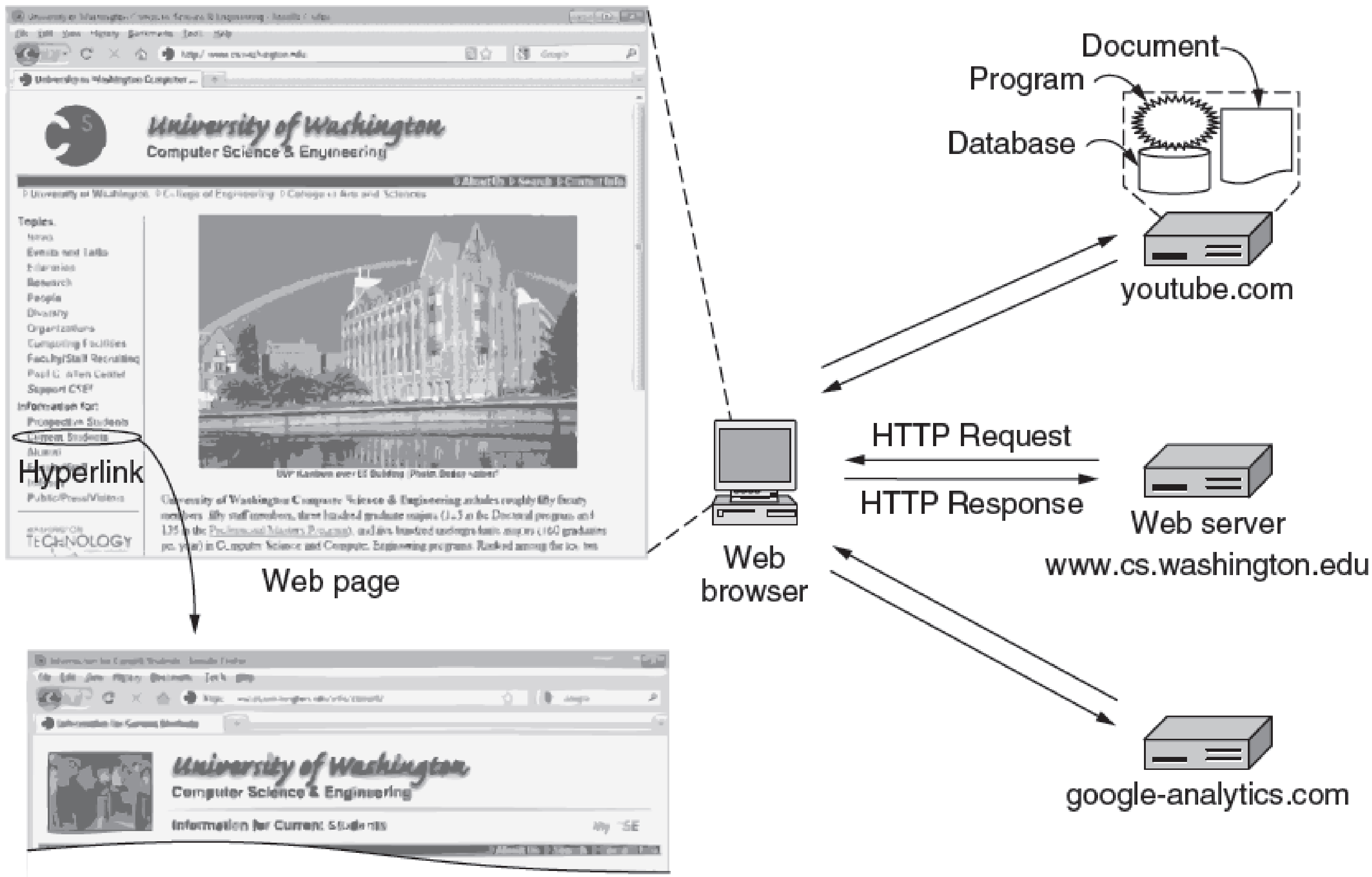
Architectural Overview

The parts of the Web model.



Architectural Overview (1)

- HTTP transfers pages from servers to browsers



Architectural Overview (2)

Three questions had to be answered before a selected page could be displayed:

1. What is the page called?
2. Where is the page located?
3. How can the page be accessed?

Architectural Overview (2)

- Pages are named with URLs (Uniform Resource Locators)
  - Example: $\underbrace{\text{http://}}_{\text{Protocol}}\underbrace{\text{www.phdcomics.com}}_{\text{Server}}\underbrace{\text{/comics.php}}_{\text{Page on server}}$

Our focus

Name	Used for	Example
http	Hypertext (HTML)	http://www.ee.uwa.edu/~rob/
https	Hypertext with security	https://www.bank.com/accounts/
ftp	FTP	ftp://ftp.cs.vu.nl/pub/minix/README
file	Local file	file:///usr/suzanne/prog.c
mailto	Sending email	mailto:JohnUser@acm.org
rtsp	Streaming media	rtsp://youtube.com/montypython.mpg
sip	Multimedia calls	sip:eve@adversary.com
about	Browser information	about:plugins

Common URL protocols

Architectural Overview (3)

Steps a client (browser) takes to follow a hyperlink:

- Determine the protocol (HTTP)
- Ask DNS for the IP address of server
- Make a TCP connection to server
- Send request for the page; server sends it back
- Fetch other URLs as needed to display the page
- Close idle TCP connections

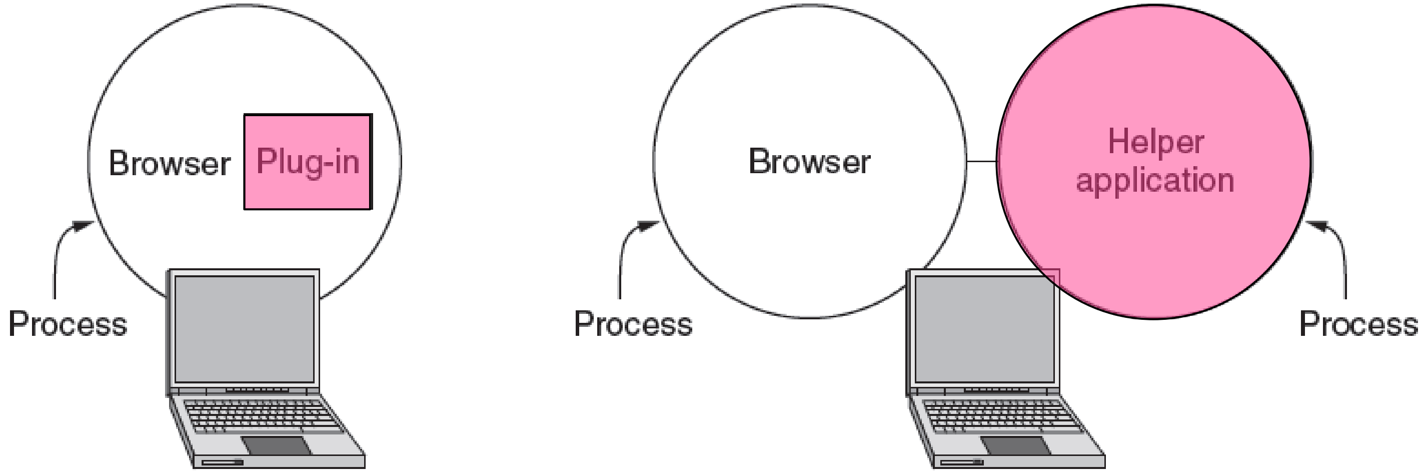
Steps a server takes to serve pages:

- Accept a TCP connection from client
- Get page request and map it to a resource (e.g., file name)
- Get the resource (e.g., file from disk)
- Send contents of the resource to the client.
- Release idle TCP connections

Architectural Overview (4)

Content type is identified by MIME types

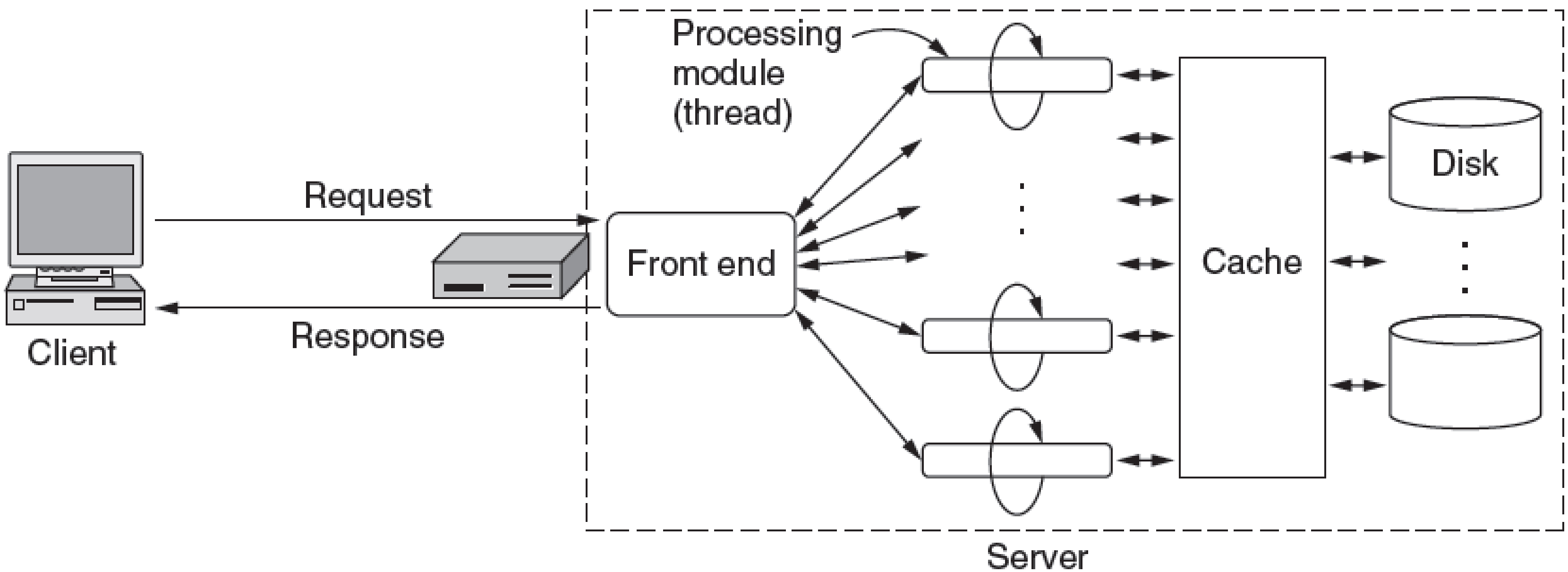
- Browser takes the appropriate action to display
- Plug-ins / helper apps extend browser for new types



Architectural Overview (5)

To scale performance, Web servers can use:

- Caching, multiple threads, and a front end



Architectural Overview (6)

Steps server performs in main loop

1. Accept a TCP connection from client
2. Get path to page, name of file requested.
3. Get the file (from disk).
4. Send contents of the file to the client.
5. Release the TCP connection.

Architectural Overview (6)

Server steps, revisited:

- Resolve name of Web page requested
- Perform access control on the Web page
- Check the cache
- Fetch requested page from disk or run program
- Determine the rest of the response
- Return the response to the client
- Make an entry in the server log

Architectural Overview (7)

Cookies support stateful client/server interactions

- Server sends cookies (state) with page response
- Client stores cookies across page fetches
- Client sends cookies back to server with requests

Domain	Path	Content	Expires	Secure
toms-casino.com	/	CustomerID=297793521	15-10-10 17:00	Yes
jills-store.com	/	Cart=1-00501;1-07031;2-13721	11-1-11 14:22	No
aportal.com	/	Prefs=Stk:CSCO+ORCL;Spt:Jets	31-12-20 23:59	No
sneaky.com	/	UserID=4627239101	31-12-19 23:59	No

Examples of cookies

Web Browser

- Generating HTTP requests
  - User types URL, clicks a hyperlink, or selects bookmark
  - User clicks “reload”, or “submit” on a Web page
  - Automatic downloading of embedded images
- Layout of response
  - Parsing HTML and rendering the Web page
  - Invoking helper applications (e.g., Acrobat, PowerPoint)
- Maintaining a cache
  - Storing recently-viewed objects
  - Checking that cached objects are fresh

Typical Web Transaction (Continued)

- Browser parses the HTTP response message
  - Extract the URL for each embedded image
  - Create new TCP connections and send new requests
  - Render the Web page, including the images

- Opportunities for caching in the browser
  - HTML file
  - Each embedded image
  - IP address of the Web site

Web Server

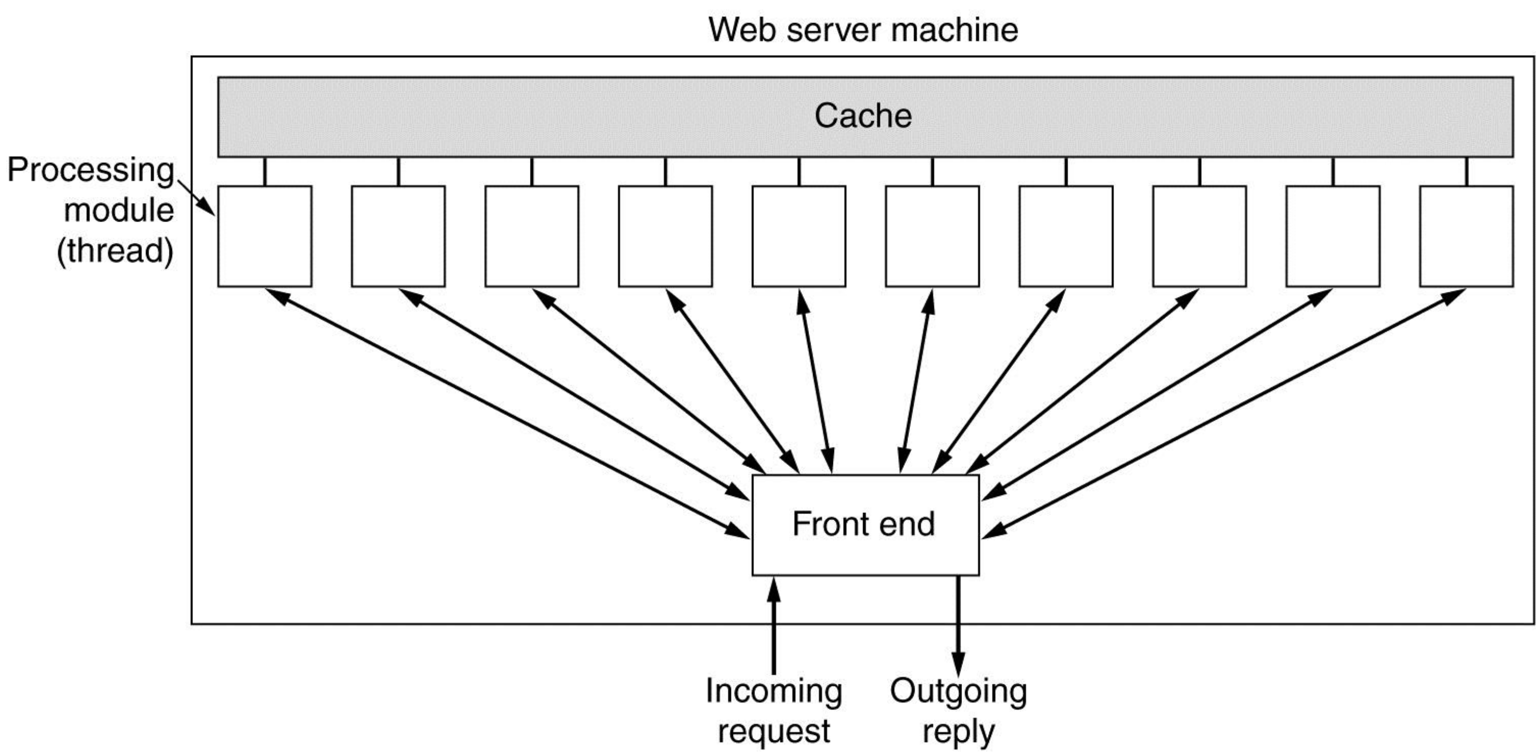
- Web site vs. Web server
  - Web site: collections of Web pages associated with a particular host name
  - Web server: program that satisfies client requests for Web resources
- Handling a client request
  - Accept the TCP connection
  - Read and parse the HTTP request message
  - Translate the URL to a filename
  - Determine whether the request is authorized
  - Generate and transmit the response

Web Server: Generating a Response

- Returning a file
  - URL corresponds to a file (e.g., /www/index.html)
  - … and the server returns the file as the response
  - … along with the HTTP response header
- Returning meta-data with no body
  - Example: client requests object "if-modified-since"
  - Server checks if the object has been modified
  - … and simply returns a "HTTP/1.1 304 Not Modified"
- Dynamically-generated responses
  - URL corresponds to a program the server needs to run
  - Server runs the program and sends the output to client

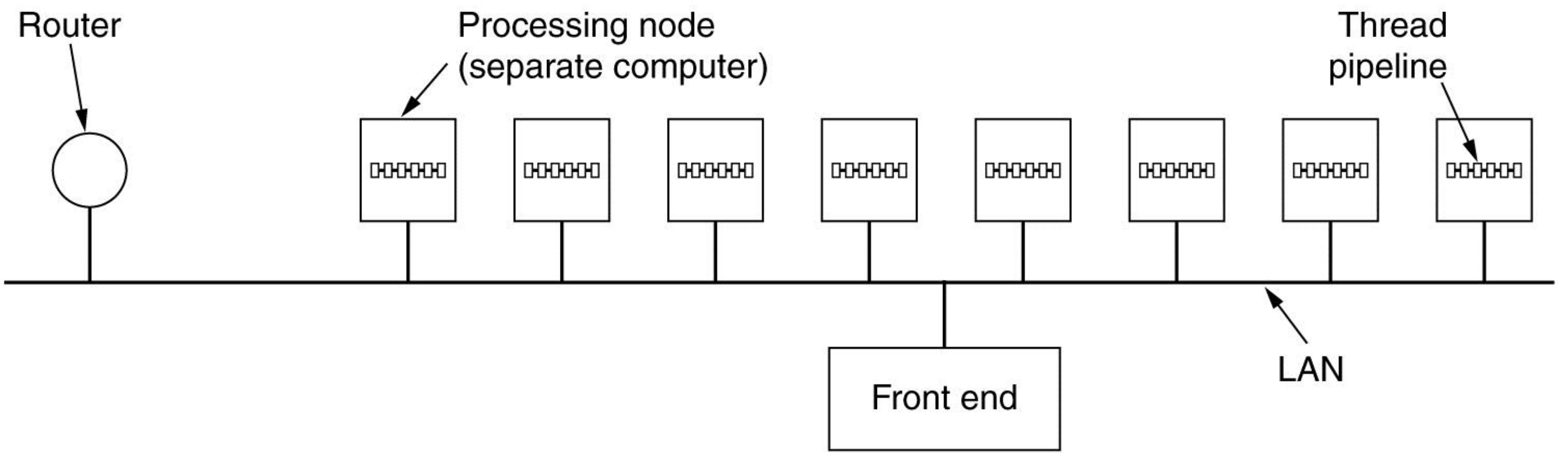
The Server Side

A multithreaded Web server with a front end and processing modules.

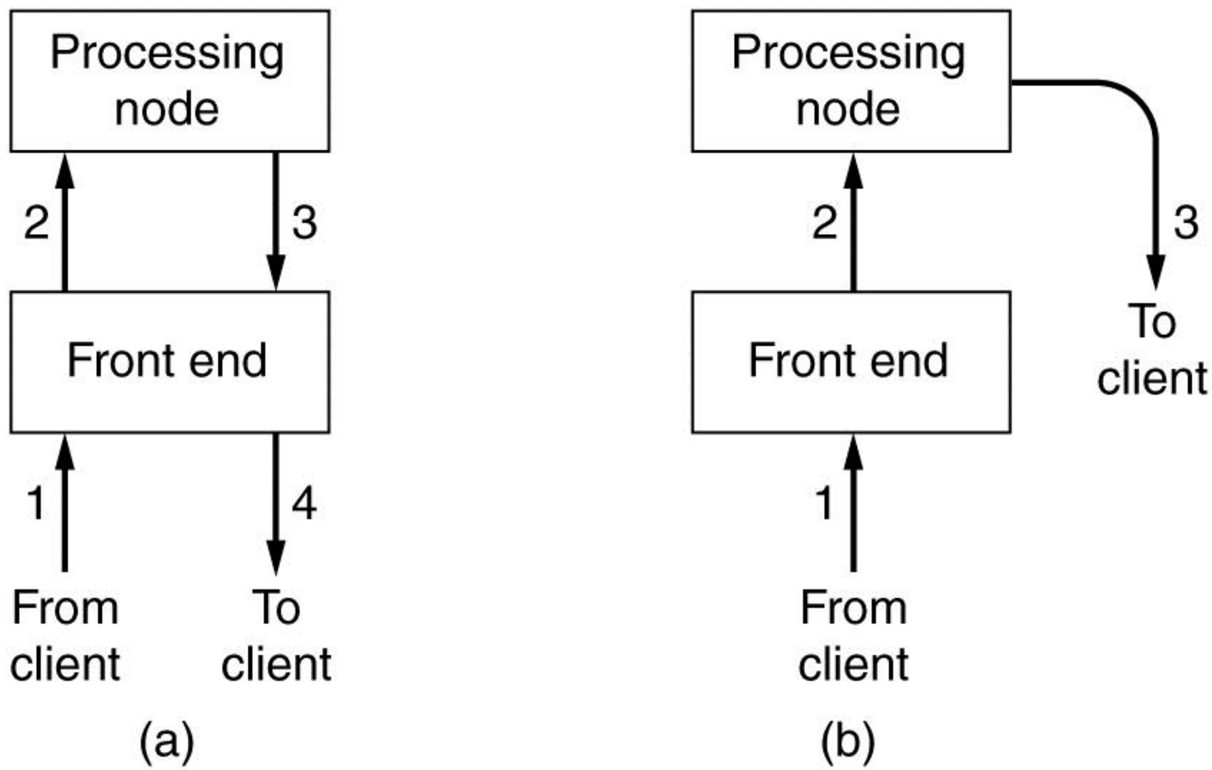


The Server Side (2)

A server farm.



The Server Side (3)



(a) Normal request-reply message sequence.
(b) Sequence when TCP handoff is used.

Hosting: Multiple Sites Per Machine

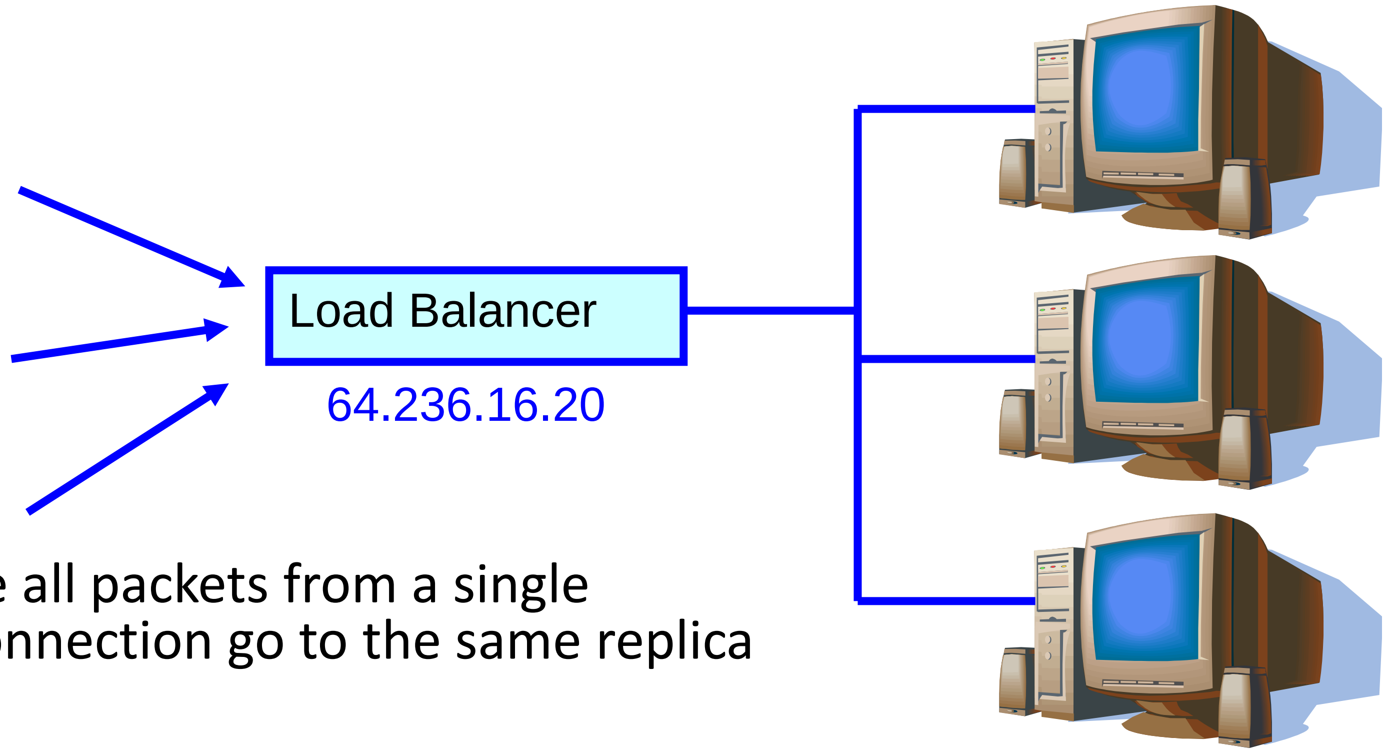
- Multiple Web sites on a single machine
  - Hosting company runs the Web server on behalf of multiple sites (e.g., www.foo.com and www.bar.com)
- Problem: returning the correct content
  - www.foo.com/index.html vs. www.bar.com/index.html
  - How to differentiate when both are on same machine?
- Solution #1: multiple servers on the same machine
  - Run multiple Web servers on the machine
  - Have a separate IP address for each server
- Solution #2: include site name in the HTTP request
  - Run a single Web server with a single IP address
  - … and include "Host" header (e.g., "Host: www.foo.com")

Hosting: Multiple Machines Per Site

- Replicating a popular Web site
  - Running on multiple machines to handle the load
  - … and to place content closer to the clients
- Problem: directing client to a particular replica
  - To balance load across the server replicas
  - To pair clients with nearby servers
- Solution #1: manual selection by clients
  - Each replica has its own site name
  - A Web page lists the replicas (e.g., by name, location)
  - … and asks clients to click on a hyperlink to pick

Hosting: Multiple Machines Per Site

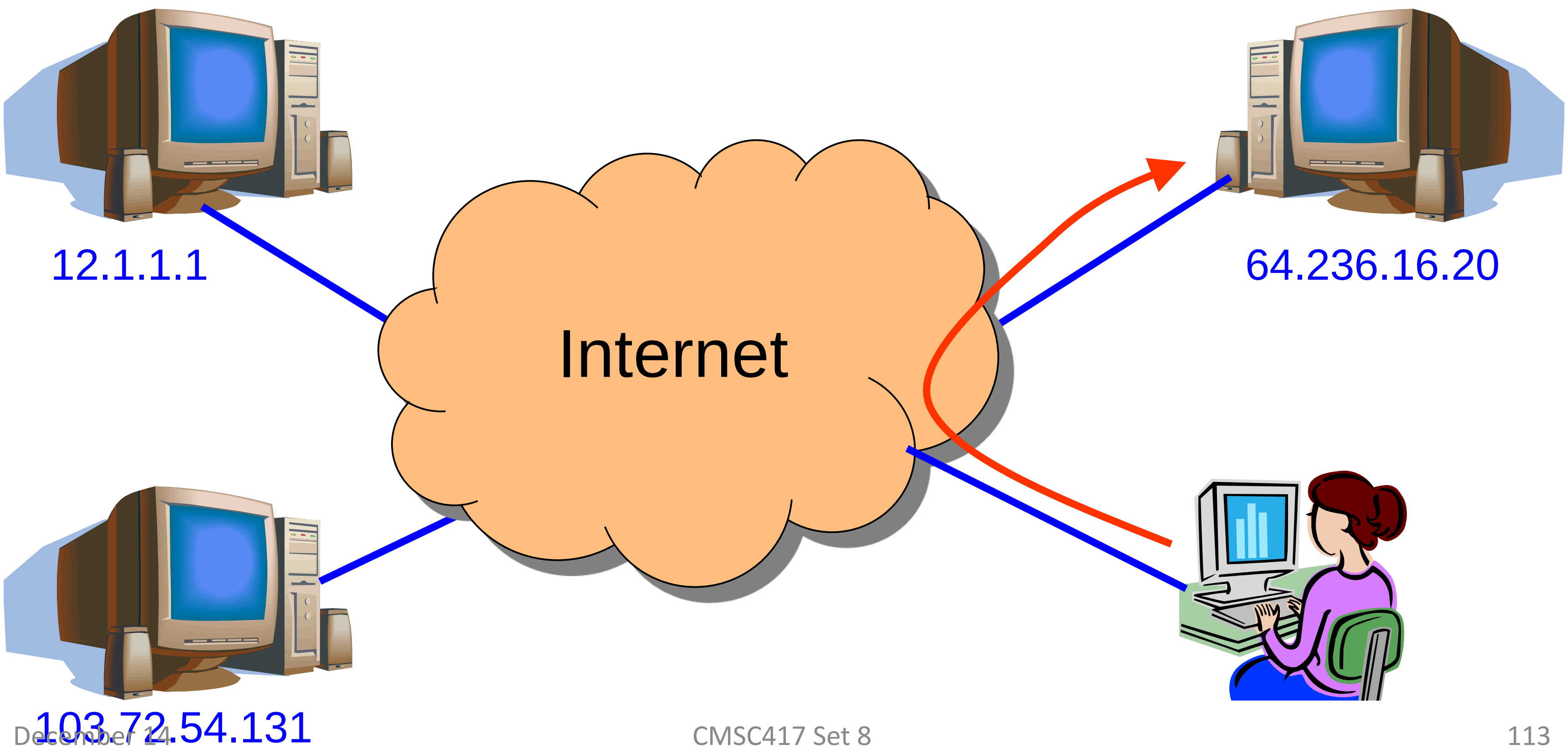
- Solution #2: single IP address, multiple machines
  - Same name and IP address for all of the replicas
  - Run multiple machines behind a single IP address



  - Ensure all packets from a single TCP connection go to the same replica

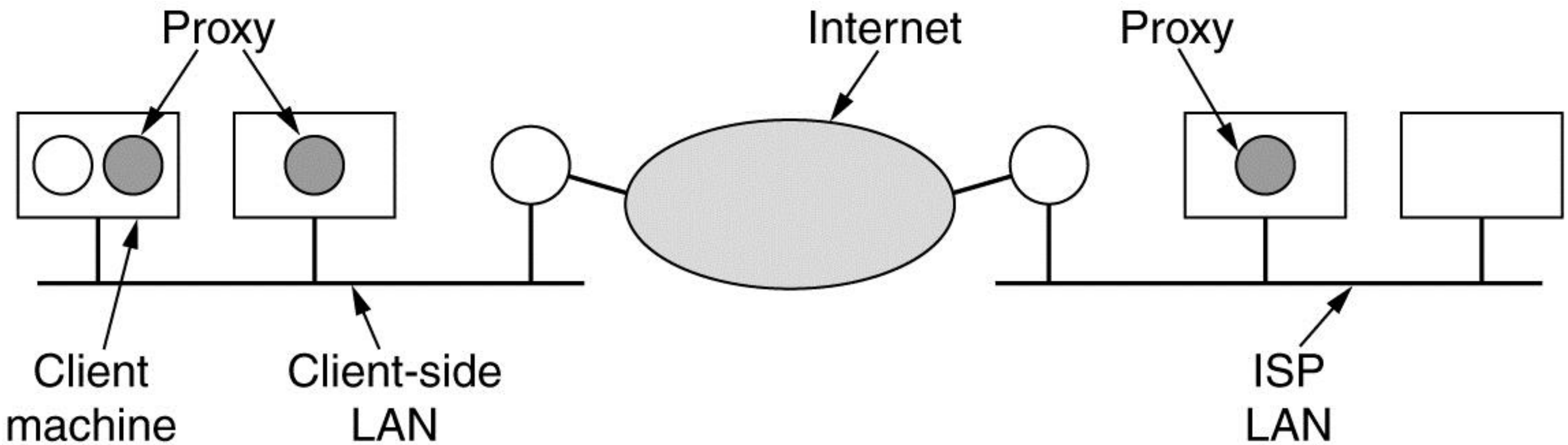
Hosting: Multiple Machines Per Site

- Solution #3: multiple addresses, multiple machines
  - Same name but different addresses for all of the replicas
  - Configure DNS server to return different addresses



Caching

Hierarchical caching with three proxies.



Caching vs. Replication

- Motivations for moving content close to users
  - Reduce latency for the user
  - Reduce load on the network and the server
  - Reduce cost for transferring data on the network
- Caching
  - Replicating the content “on demand” after a request
  - Storing the response message locally for future use
  - May need to verify if the response has changed
  - … and some responses are not cacheable
- Replication
  - Planned replication of the content in multiple locations
  - Updating of resources is handled outside of HTTP
  - Can replicate scripts that create dynamic responses

Caching vs. Replication (Continued)

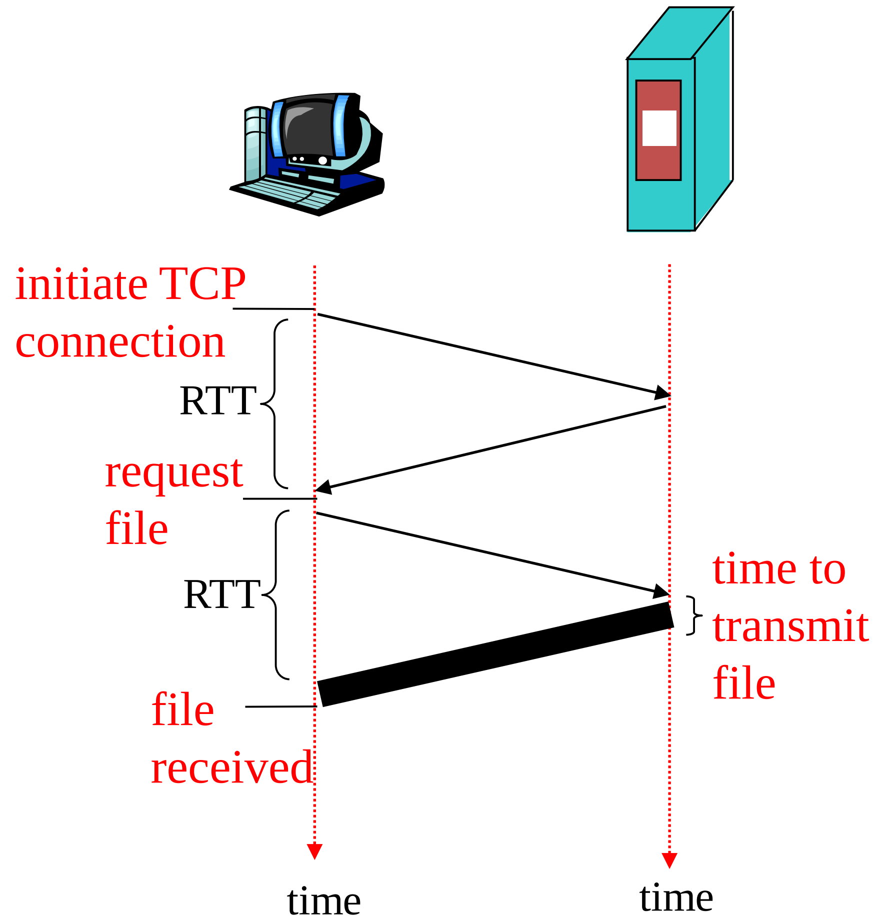
- Caching initially viewed as very important in HTTP
  - Many additions to HTTP to support caching
  - … and, in particular, cache validation
- Deployment of caching proxies in the 1990s
  - Service providers and enterprises deployed proxies
  - … to cache content across a community of users
  - Though, sometimes the gains weren't very dramatic
- Then, content distribution networks emerged
  - Companies (like Akamai) that replicate Web sites
  - Host all (or part) of a Web site for a content provider
  - Place replicas all over the world on many machines

TCP Interaction: Multiple Transfers

- Most Web pages have multiple objects
  - E.g., HTML file and multiple embedded images
- Serializing the transfers is not efficient
  - Sending the images one at a time introduces delay
  - Cannot start retrieving second images until first arrives
- Parallel connections
  - Browser opens multiple TCP connections (e.g., 4)
  - … and retrieves a single image on each connection
- Performance trade-offs
  - Multiple downloads sharing the same network links
  - Unfairness to other traffic traversing the links

TCP Interaction: Short Transfers

- Most HTTP transfers are short
  - Very small request message (e.g., a few hundred bytes)
  - Small response message (e.g., a few kilobytes)
- TCP overhead may be big
  - Three-way handshake to establish connection
  - Four-way handshake to tear down the connection



TCP Interaction: Short Transfers

- Round-trip time estimation
  - Very large at the start of a connection (e.g., 3 seconds)
  - Leads to latency in detecting lost packets
- Congestion window
  - Small value at beginning of connection (e.g., 1 MSS)
  - May not reach a high value before transfer is done
- Timeout vs. triple-duplicate ACK
  - Two main ways of detecting packet loss
  - Timeout is slow, and triple-duplicate ACK is fast
  - However, triple-dup-ACK requires many packets in flight
  - … which doesn't happen for very short transfers

TCP Interaction: Persistent Connections

- Handle multiple transfers per connection
  - Maintain the TCP connection across multiple requests
  - Either the client or server can tear down the connection
  - Added to HTTP after the Web became very popular
- Performance advantages
  - Avoid overhead of connection set-up and tear-down
  - Allow TCP to learn a more accurate RTT estimate
  - Allow the TCP congestion window to increase
- Further enhancement: pipelining
  - Send multiple requests one after the other
  - … before receiving the first response

Key Ideas

- Key ideas underlying the Web
  - Uniform Resource Identifier (URI)
  - HyperText Markup Language (HTML)
  - HyperText Transfer Protocol (HTTP)
  - Browser helper applications based on content type
- Main Web components
  - Clients, proxies, and servers
- Dependence on underlying Internet protocols
  - DNS and TCP

Main Components: URL

- Uniform Resource Identifier (URI)
  - Denotes a resource independent of its location or value
  - A pointer to a “black box” that accepts request methods
- Formatted string
  - Protocol for communicating with server (e.g., http)
  - Name of the server (e.g., www.foo.com)
  - Name of the resource (e.g., coolpic.gif)
- Name (URN), Locator (URL), and Identifier (URI)
  - URN: globally unique name, like an ISBN # for a book
  - URI: identifier representing the contents of the book
  - URL: location of the book

URLs – Uniform Resource Locaters

Some common URLs.

Name	Used for	Example
http	Hypertext (HTML)	http://www.cs.vu.nl/~ast/
ftp	FTP	ftp://ftp.cs.vu.nl/pub/minix/README
file	Local file	file:///usr/suzanne/prog.c
news	Newsgroup	news:comp.os.minix
news	News article	news:AA0134223112@cs.utah.edu
gopher	Gopher	gopher://gopher.tc.umn.edu/11/Libraries
mailto	Sending e-mail	mailto:JohnUser@acm.org
telnet	Remote login	telnet://www.w3.org:80

Main Components: HTML

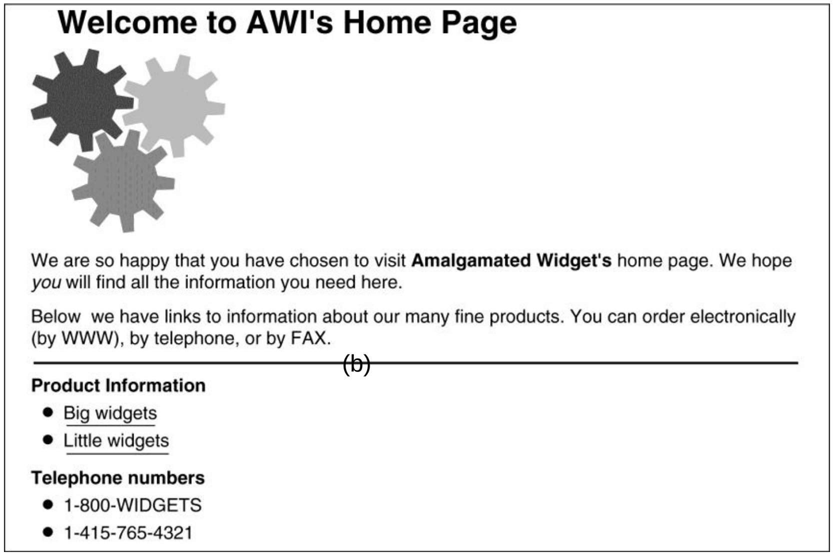
- HyperText Markup Language (HTML)
  - Representation of hyptertext documents in ASCII format
  - Format text, reference images, embed hyperlinks
  - Interpreted by Web browsers when rendering a page
- Straight-forward and easy to learn
  - Simplest HTML document is a plain text file
    - Easy to add formatting, references, bullets, etc.
  - Automatically generated by authoring programs
    - Tools to aid users in creating HTML files
- Web page
  - Base HTML file referenced objects (e.g., images)
  - Each object has its own URL

HTML – HyperText Markup Language

(a) The HTML for a sample Web page. (b) The formatted page.

```
<html>
<head><title> AMALGAMATED   WIDGET,  INC. </title> </head>
<body> <h1> Welcome to AWI’s  Home Page</h1>
<img src="http://www.widget.com/images/logo.gif" ALT="AWI  Logo"> <br>
We are so happy that you have chosen to visit <b> Amalgamated  Widget’s </b>
home page. We hope <i> you </i> will  find  all  the information you need here.
<p>Below we have links to information  about our many fine products.
You can order  electronically (by WWW), by  telephone, or by fax. </p>
<hr>
<h2> Product information  </h2>
<ul>
   <li> <a href="http://widget.com/products/big">  Big widgets </a>
   <li> <a href="http://widget.com/products/little">   Little widgets </a>
</ul>
<h2> Telephone numbers</h2>
<ul>
   <li> By telephone: 1-800-WIDGETS
   <li> By fax:  1-415-765-4321
</ul>
</body>
</html>
```

(a)



(b)

HTML (2)

Tag	Description
<html> ... </html>	Declares the Web page to be written in HTML
<head> ... </head>	Delimits the page's head
<title> ... </title>	Defines the title (not displayed on the page)
<body> ... </body>	Delimits the page's body
<h *n*> ... </h*n*>	Delimits a level *n* heading
 ... 	Set ... in boldface
<i> ... </i>	Set ... in italics
<center> ... </center>	Center ... on the page horizontally
...	Brackets an unordered (bulleted) list
...	Brackets a numbered list
	Starts a list item (there is no)
 	Forces a line break here
<p>	Starts a paragraph
<hr>	Inserts a Horizontal rule
<img src="...">	Displays an image here
 ... 	Defines a hyperlink

A selection of common HTML tags. some can have additional parameters.

Forms

```
<html>
<head> <title> A sample page with a table </title> </head>
<body>
<table border=1 rules=all>
<caption> Some Differences between HTML Versions </caption>
<col align=left>
<col align=center>
<col align=center>
<col align=center>
<col align=center>
<tr> <th>Item  <th>HTML 1.0  <th>HTML 2.0  <th>HTML 3.0 <th>HTML 4.0 </tr>
<tr> <th> Hyperlinks <td> x <td> x <td> x <td> x </tr>
<tr> <th> Images <td> x <td> x <td> x <td> x </tr>
<tr> <th> Lists <td> x <td> x <td> x <td> x </tr>
<tr> <th> Active Maps and Images <td>   <td> x <td> x <td> x </tr>
<tr> <th> Forms <td>   <td> x <td> x <td> x </tr>
<tr> <th> Equations <td>   <td>   <td> x <td> x </tr>
<tr> <th> Toolbars <td>   <td>   <td> x <td> x </tr>
<tr> <th> Tables <td>   <td>   <td> x <td> x </tr>
<tr> <th> Accessibility features <td>   <td>   <td>   <td> x </tr>
<tr> <th> Object embedding <td>   <td>   <td>   <td> x </tr>
<tr> <th> Scripting <td>   <td>   <td>   <td> x </tr>
</table>
</body>
</html>
```

(a)

Some Differences between HTML Versions

Item	HTML 1.0	HTML 2.0	HTML 3.0	HTML 4.0
Hyperlinks	x	x	x	x
Images	x	x	x	x
Lists	x	x	x	x
Active Maps and Images		x	x	x
Forms		x	x	x
Equations			x	x
Toolbars			x	x
Tables			x	x
Accessibility features				x
Object embedding				x
Scripting				x

(b)

(a) An HTML table.

(b) A possible rendition of this table.

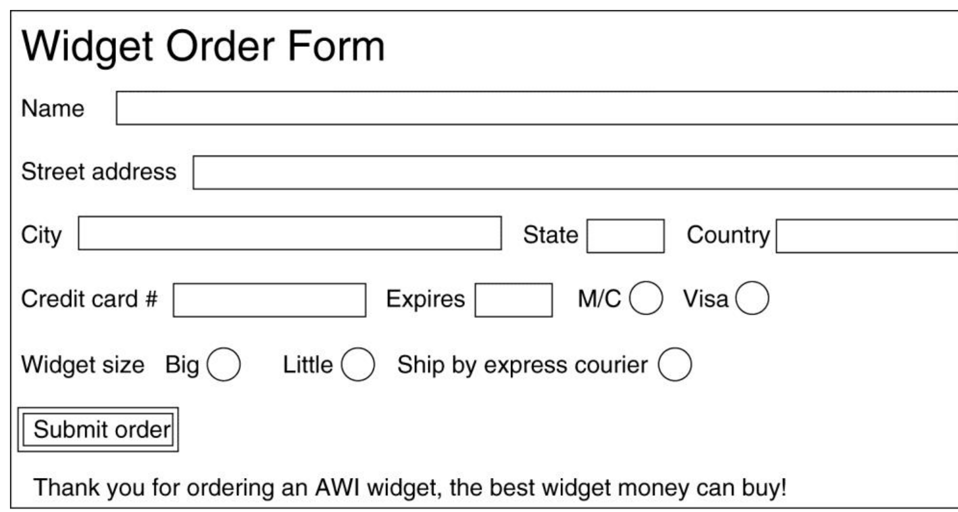
Forms (2)

(a) The HTML for an order form.

(b) The formatted page.

```
<html>
<head> <title> AWI CUSTOMER ORDERING FORM </title> </head>
<body>
<h1> Widget Order Form </h1>
<form ACTION="http://widget.com/cgi-bin/widgetorder" method=POST>
<p> Name <input name="customer" size=46> </p>
<p> Street Address <input name="address" size=40> </p>
<p> City <input name="city" size=20> State <input name="state" size =4>
Country <input name="country" size=10> </p>
<p> Credit card # <input name="cardno" size=10>
Expires <input name="expires" size=4>
M/C <input name="cc" type=radio value="mastercard">
VISA <input name="cc" type=radio value="visacard"> </p>
<p> Widget size Big <input name="product" type=radio value="expensive">
Little <input name="product" type=radio value="cheap">
Ship by express courier <input name="express" type=checkbox> </p>
<p><input type=submit value="submit order"> </p>
Thank you for ordering an AWI widget, the best widget money can buy!
</form>
</body>
</html>
```

(a)



(b)

Forms (3)

A possible response from the browser to the server with information filled in by the user.

```
customer=John+Doe&address=100+Main+St.&city=White+Plains&
state=NY&country=USA&cardno=1234567890&expires=6/98&cc=mastercard&
product=cheap&express=on
```

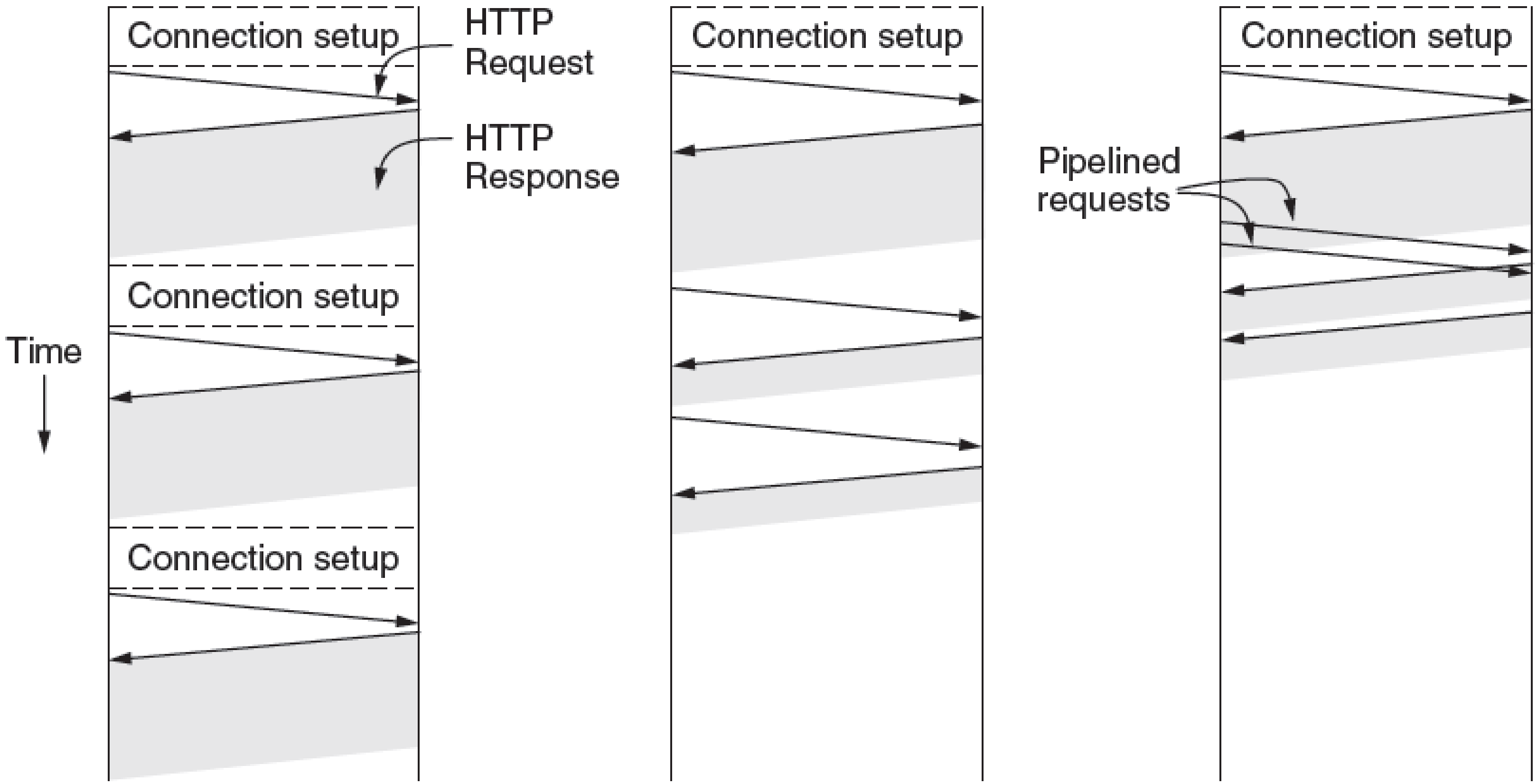
HTTP (1)

HTTP (HyperText Transfer Protocol) is a request-response protocol that runs on top of TCP

- Fetches pages from server to client
- Server usually runs on port 80
- Headers are given in readable ASCII
- Content is described with MIME types
- Protocol has support for pipelining requests
- Protocol has support for caching

HTTP (2)

- HTTP uses persistent connections to improve performance



One connection for each request

Sequential requests on one connection

Pipelined requests on one connection

HTTP (3)

HTTP has several request methods.

Fetch a page → GET

Used to send input data to a server program → POST

Method	Description
GET	Read a Web page
HEAD	Read a Web page's header
POST	Append to a Web page
PUT	Store a Web page
DELETE	Remove the Web page
TRACE	Echo the incoming request
CONNECT	Connect through a proxy
OPTIONS	Query options for a page

HTTP (4)

Response codes tell the client how the request fared:

Code	Meaning	Examples
1xx	Information	100 = server agrees to handle client's request
2xx	Success	200 = request succeeded; 204 = no content present
3xx	Redirection	301 = page moved; 304 = cached page still valid
4xx	Client error	403 = forbidden page; 404 = page not found
5xx	Server error	500 = internal server error; 503 = try again later

HTTP (5)

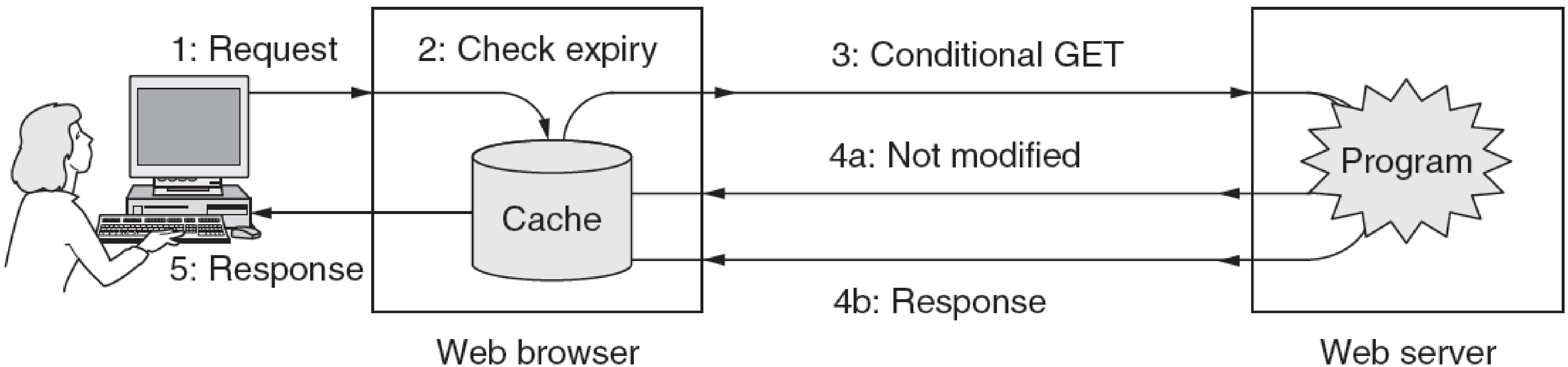
Many headers carry key information:

Function	**Example Headers**
Browser capabilities (client → server)	User-Agent, Accept, Accept-Charset, Accept-Encoding, Accept-Language
Caching related (mixed directions)	If-Modified-Since, If-None-Match, Date, Last-Modified, Expires, Cache-Control, ETag
Browser context (client → server)	Cookie, Referer, Authorization, Host
Content delivery (server → client)	Content-Encoding, Content-Length, Content-Type, Content-Language, Content-Range, Set-Cookie

HTTP (6)

HTTP caching checks to see if the browser has a known fresh copy, and if not if the server has updated the page

- Uses a collection of headers for the checks
- Can include further levels of caching (e.g., proxy)



Main Components: HTTP

- HyperText Transfer Protocol (HTTP)
  - Client-server protocol for transferring resources
  - Client sends request and server sends response
- Important properties of HTTP
  - Request-response protocol
  - Reliance on a global URI
  - Resource metadata
  - Statelessness
  - ASCII format

```
telnet www.cs.umd.edu 80

GET /~agrawala/ HTTP/1.1
Host: www.cs.umd.edu
```

Example: HyperText Transfer Protocol

GET /courses/archive/fall2008/cmsc417/ HTTP/1.1
Host: www.cs.umd.edu
User-Agent: Mozilla/4.03
<CRLF>

Request

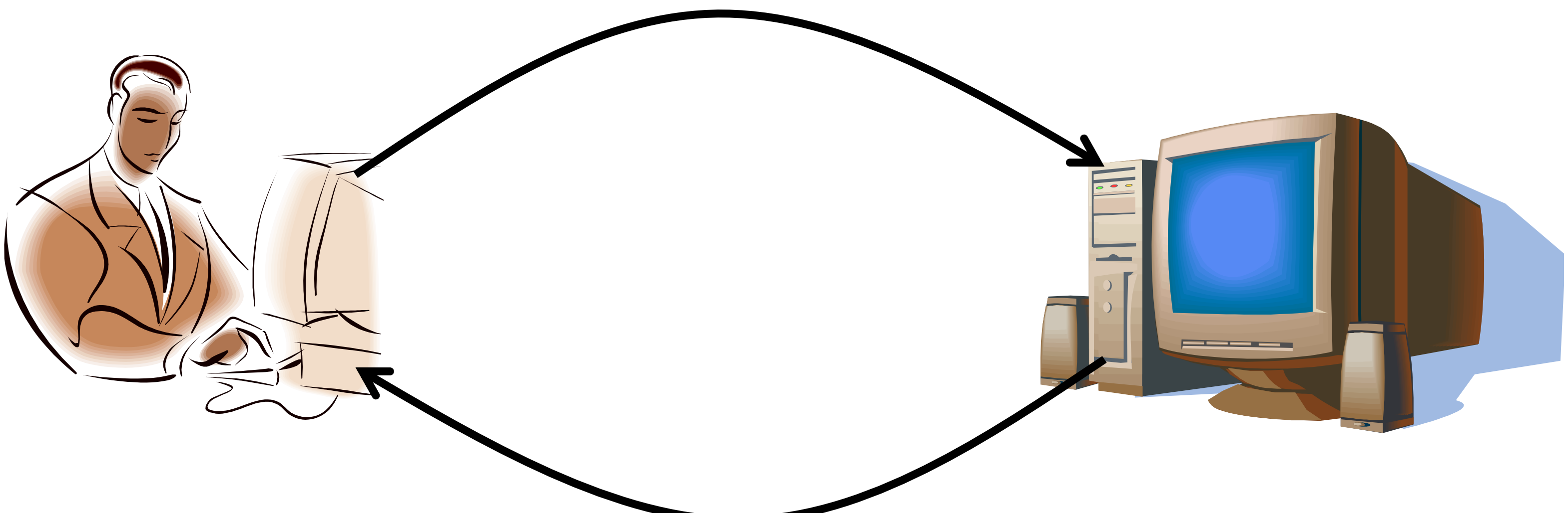
HTTP/1.1 200 OK
Date: Mon, Dec 3, 2008, 13:09:03 GMT
Server: Netscape-Enterprise/3.5.1
Last-Modified: Mon, Dec 1, 2008, 11:12:23 GMT
Content-Length: 21
<CRLF>
Site under construction

Response

HTTP: Request-Response Protocol

- Client program
  - Running on end host
  - Requests service
  - E.g., Web browser
- Server program
  - Running on end host
  - Provides service
  - E.g., Web server

`GET /index.html`



`"Site under construction"`

HTTP Request Message

- Request message sent by a client
  - Request line: method, resource, and protocol version
  - Request headers: provide information or modify request
  - Body: optional data (e.g., to “POST” data to the server)

request line
(GET, POST,
HEAD commands)

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
User-agent: Mozilla/4.0
Connection: close
Accept-language:fr
```

header
lines

(extra carriage return, line feed)

Carriage return,
line feed
indicates end
of message

Example: Conditional GET Request

- Fetch resource only if it has changed at the server

```
GET /courses/archive/fall2008/cmsc417/ HTTP/1.1
Host: www.cs.umd.edu
User-Agent: Mozilla/4.03
If-Modified-Since: Mon, Dec 1, 2008 11:12:23 GMT
<CRLF>
```

- Server avoids wasting resources to send again
  - Server inspects the “last modified” time of the resource
  - … and compares to the “if-modified-since” time
  - Returns “304 Not Modified” if resource has not changed
  - …. or a “200 OK” with the latest version otherwise

HTTP Response Message

- Response message sent by a server
  - Status line: protocol version, status code, status phrase
  - Response headers: provide information
  - Body: optional data

status line
(protocol
status code
status phrase)

```
HTTP/1.1 200 OK
Connection close
Date: Thu, 06 Aug 1998 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 1998 …...
Content-Length: 6821
Content-Type: text/html

data data data data data ...
```

header
lines

data, e.g.,
requested
HTML file

HTTP Resource Meta-Data

- Meta-data
  - Information relating to a resource
  - … but not part of the resource itself
- Example meta-data
  - Size of a resource
  - Type of the content
  - Last modification time
- Concept borrowed from e-mail protocols
  - Multipurpose Internet Mail Extensions (MIME)
  - Data format classification (e.g., Content-Type: text/html)
  - Enables browsers to automatically launch a viewer

HTTP Message Headers

Header	Type	Contents
User-Agent	Request	Information about the browser and its platform
Accept	Request	The type of pages the client can handle
Accept-Charset	Request	The character sets that are acceptable to the client
Accept-Encoding	Request	The page encodings the client can handle
Accept-Language	Request	The natural languages the client can handle
Host	Request	The server's DNS name
Authorization	Request	A list of the client's credentials
Cookie	Request	Sends a previously set cookie back to the server
Date	Both	Date and time the message was sent
Upgrade	Both	The protocol the sender wants to switch to
Server	Response	Information about the server
Content-Encoding	Response	How the content is encoded (e.g., gzip)
Content-Language	Response	The natural language used in the page
Content-Length	Response	The page's length in bytes
Content-Type	Response	The page's MIME type
Last-Modified	Response	Time and date the page was last changed
Location	Response	A command to the client to send its request elsewhere
Accept-Ranges	Response	The server will accept byte range requests
Set-Cookie	Response	The server wants the client to save a cookie

Some HTTP message headers.

Stateless Protocol

- Stateless protocol
  - Each request-response exchange treated independently
  - Clients and servers not required to retain state
- Statelessness to improve scalability
  - Avoid need for the server to retain info across requests
  - Enable the server to handle a higher rate of requests
- However, some applications need state
  - To uniquely identify the user or store temporary info
  - E.g., personalize a Web page, compute profiles or access statistics by user, keep a shopping cart, etc.
  - Lead to the introduction of “cookies” in the mid 1990s

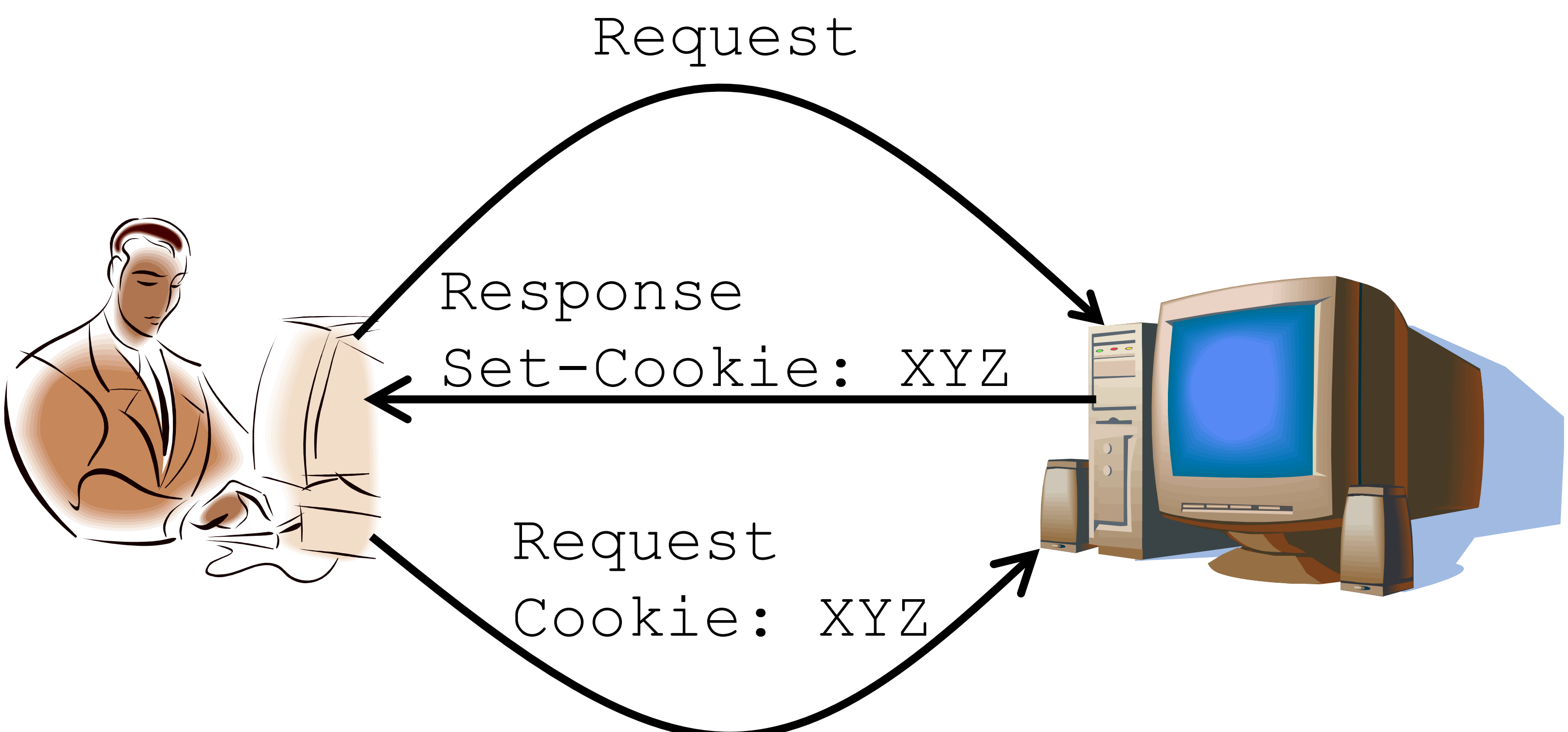
Statelessness and Cookies

Domain	Path	Content	Expires	Secure
toms-casino.com	/	CustomerID=497793521	15-10-02 17:00	Yes
joes-store.com	/	Cart=1-00501;1-07031;2-13721	11-10-02 14:22	No
aportal.com	/	Prefs=Stk:SUNW+ORCL;Spt:Jets	31-12-10 23:59	No
sneaky.com	/	UserID=3627239101	31-12-12 23:59	No

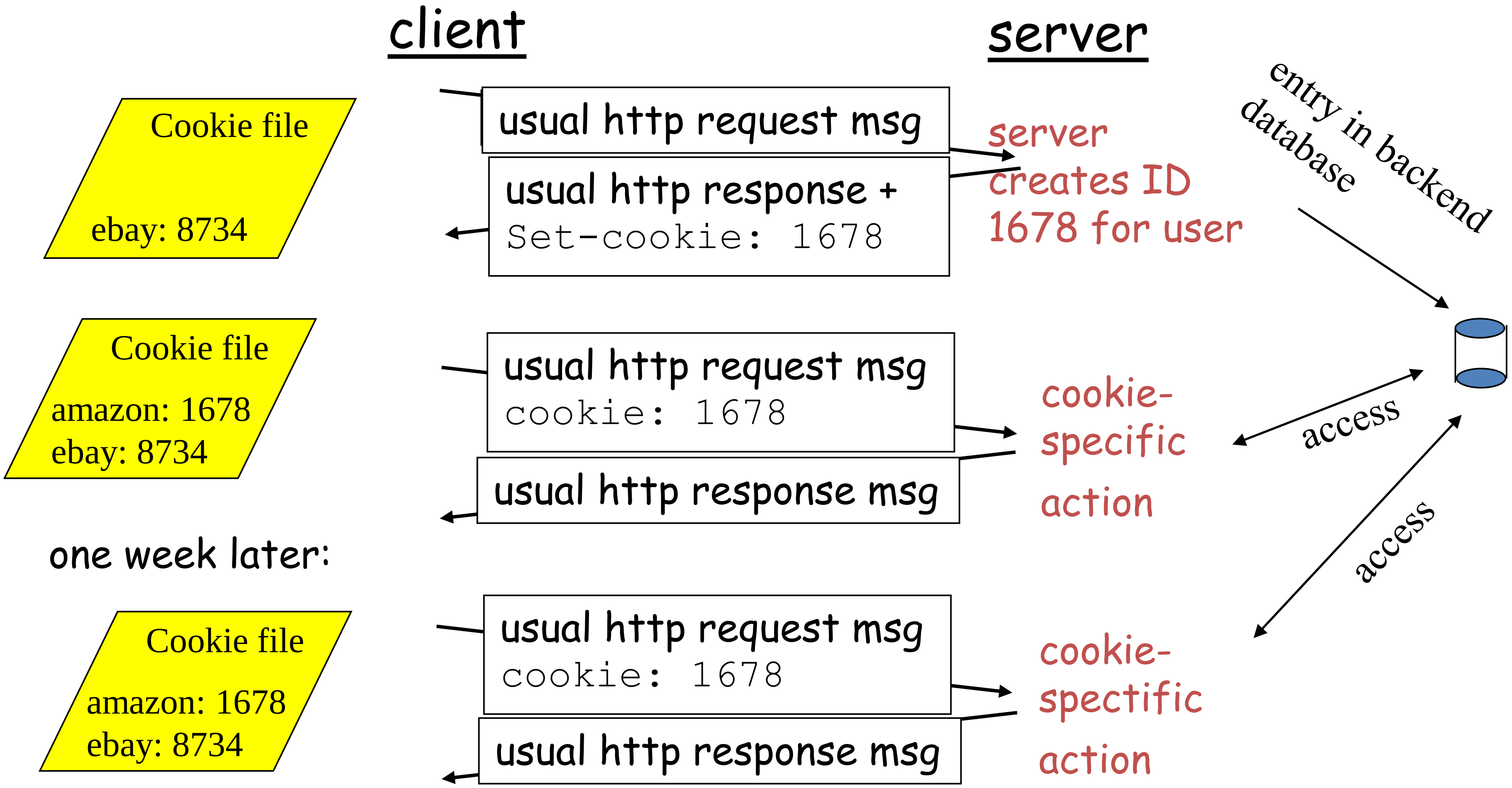
Some examples of cookies.

Cookies

- Cookie
  - Small state stored by client on behalf of server
  - Included in future requests to the server



Cookies Examples



XML and XSL

A simple Web page in XML.

```xml
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" href="b5.xsl"?>

<book_list>

<book>
   <title> Computer Networks, 4/e </title>
   <author> Andrew S. Tanenbaum </author>
   <year> 2003 </year>
</book>

<book>
   <title> Modern Operating Systems, 2/e </title>
   <author> Andrew S. Tanenbaum </author>
   <year> 2001 </year>
</book>

<book>
   <title> Structured Computer Organization, 4/e </title>
   <author> Andrew S. Tanenbaum </author>
   <year> 1999 </year>
</book>

</book_list>
```

XML and XSL (2)

```
<?xml version='1.0'?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:template match="/">

<html>
<body>

<table border="2">
  <tr>
     <th> Title</th>
     <th> Author</th>
     <th> Year </th>
  </tr>

  <xsl:for-each select="book_list/book">
  <tr>
     <td> <xsl:value-of select="title"/> </td>
     <td> <xsl:value-of select="author"/> </td>
     <td> <xsl:value-of select="year"/> </td>
 </tr>
 </xsl:for-each>
</table>

</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

A style sheet in XSL.

Static Web Pages (1)

Static Web pages are simply files

- Have the same contents for each viewing

Can be visually rich and interactive nonetheless:

- HTML that mixes text and images
- Forms that gather user input
- Style sheets that tailor presentation
- Vector graphics, videos, and more (over) . . .

Static Web Pages (2)

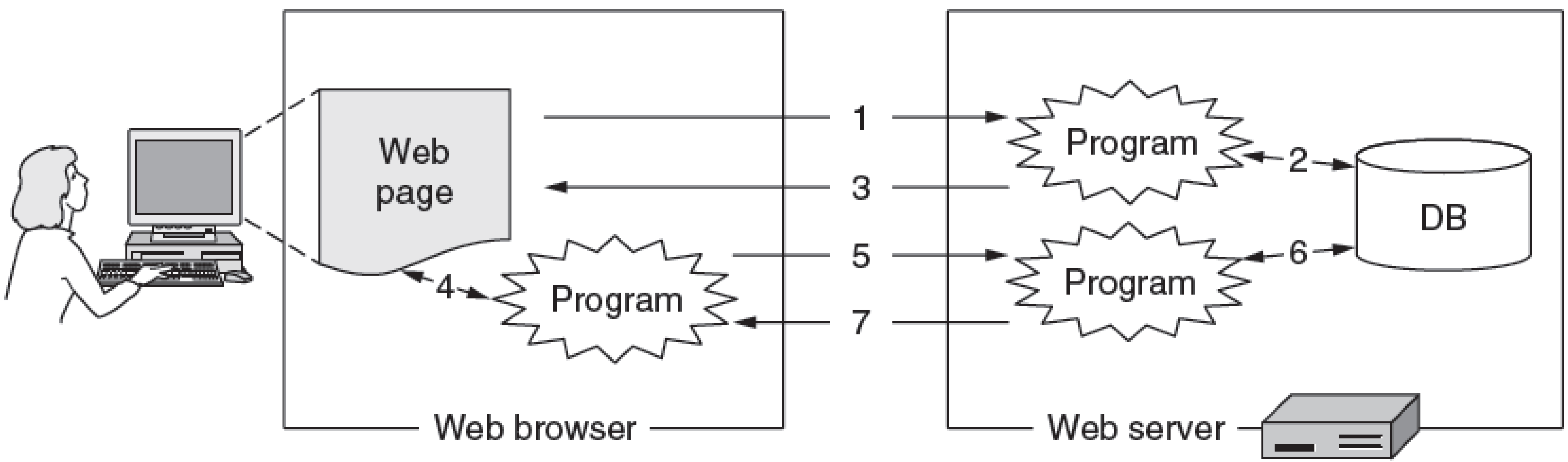
Progression of features through HTML 5.0

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 (1)

Dynamic pages are generated by programs running at the server (with a database) and the client

- E.g., PHP at server, JavaScript at client



Dynamic Pages & Web Applications (2)

Web page that gets form input and calls a server program

```
<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

PHP calls

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

Resulting Web page (for inputs “Barbara” and “32”)

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

Dynamic Pages & Web Applications (3)

JavaScript program produces result page in the browser

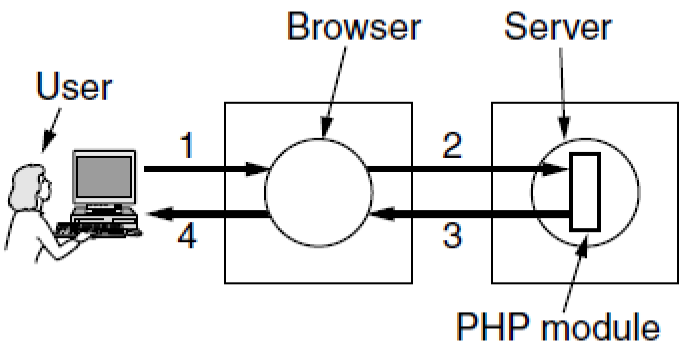
```
<html>
<head>
<script language="javascript" type="text/javascript">
function response(test_form) {
   var person = test_form.name.value;
   var years = eval(test_form.age.value) + 1;
   document.open();
   document.writeln("<html> <body>");
   document.writeln("Hello " + person + ".<br>");
   document.writeln("Prediction: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

First page with form, gets input and calls program above

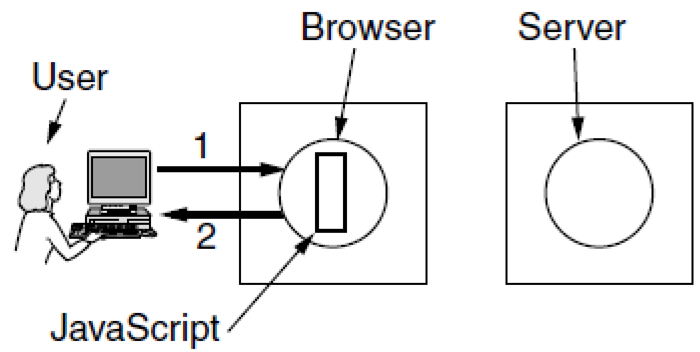
```
<body>
<form>
Please enter your name: <input type="text" name="name">
<p>
Please enter your age: <input type="text" name="age">
<p>
<input type="button" value="submit" onclick="response(this.form)">
</form>
</body>
</html>
```

Dynamic Pages & Web Applications (4)

The difference between server and client programs



Server-side scripting with PHP



Client-side scripting with JavaScript

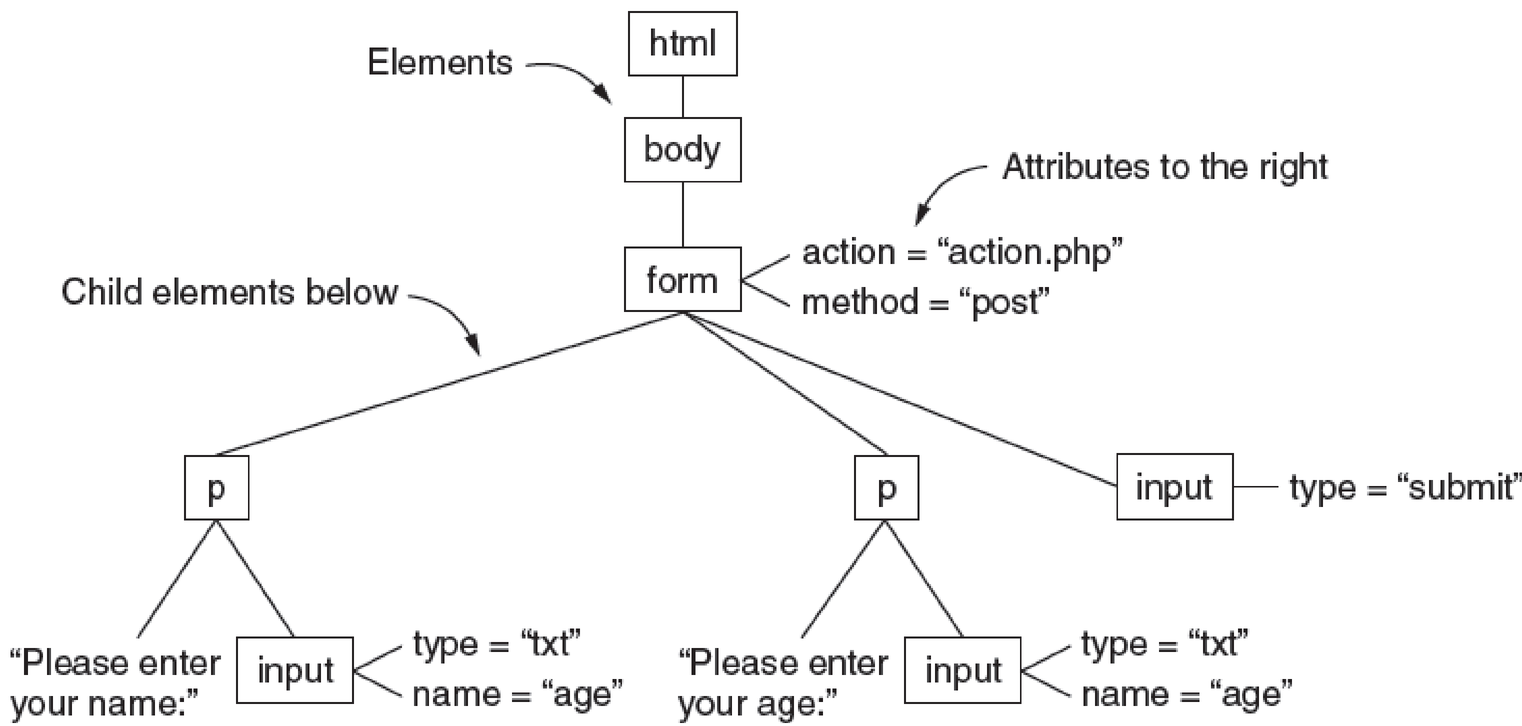
Dynamic Pages & Web Applications (5)

Web applications use a set of technologies that work together, e.g. AJAX:

- HTML: present information as pages.
- DOM: change parts of pages while they are viewed.
- XML: let programs exchange data with the server.
- Asynchronous way to send and retrieve XML data.
- JavaScript as a language to bind all this together.

Dynamic Pages & Web Applications (6)

The DOM (Document Object Model) tree represents Web pages as a structure that



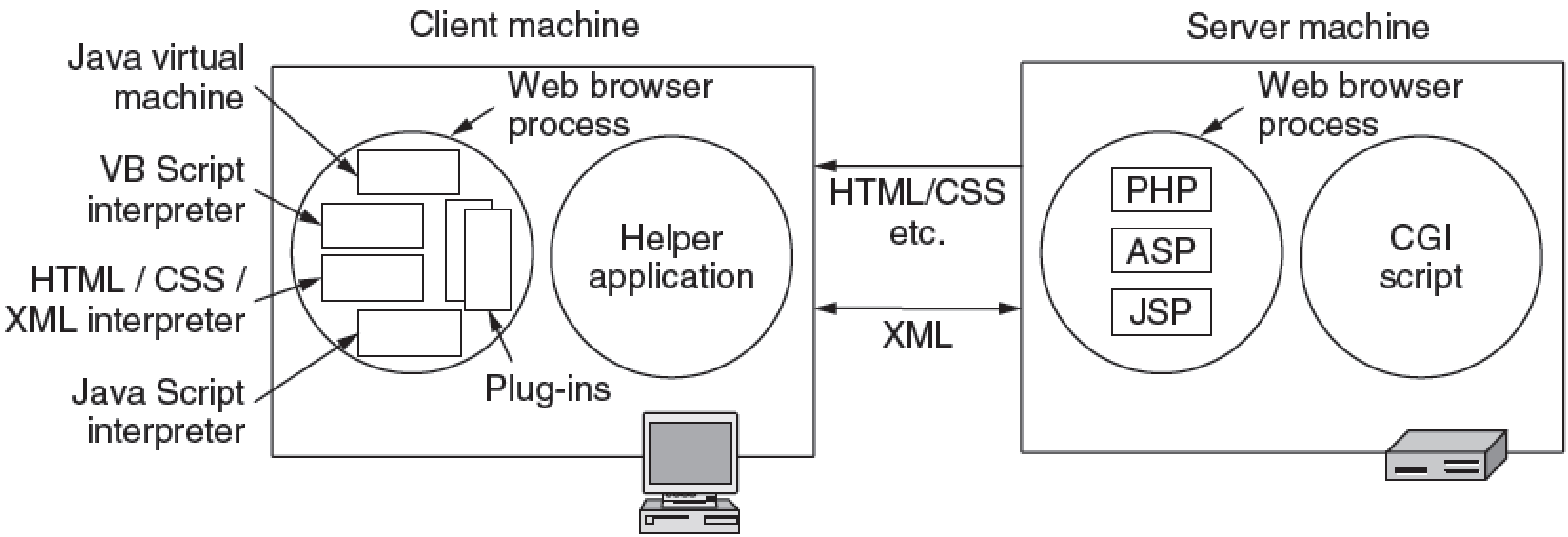
Dynamic Pages & Web Applications (7)

XML captures document structure, not presentation like HTML. For example:

```
<?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 (8)

Web applications use a set of technologies, revisited:



The Mobile Web (1)

Difficulties for mobile phones browsing the web

1. Relatively small screens
2. Limited input capabilities, lengthy input.
3. Network bandwidth is limited
4. Connectivity may be intermittent.
5. Computing power is limited

The Mobile Web (2)

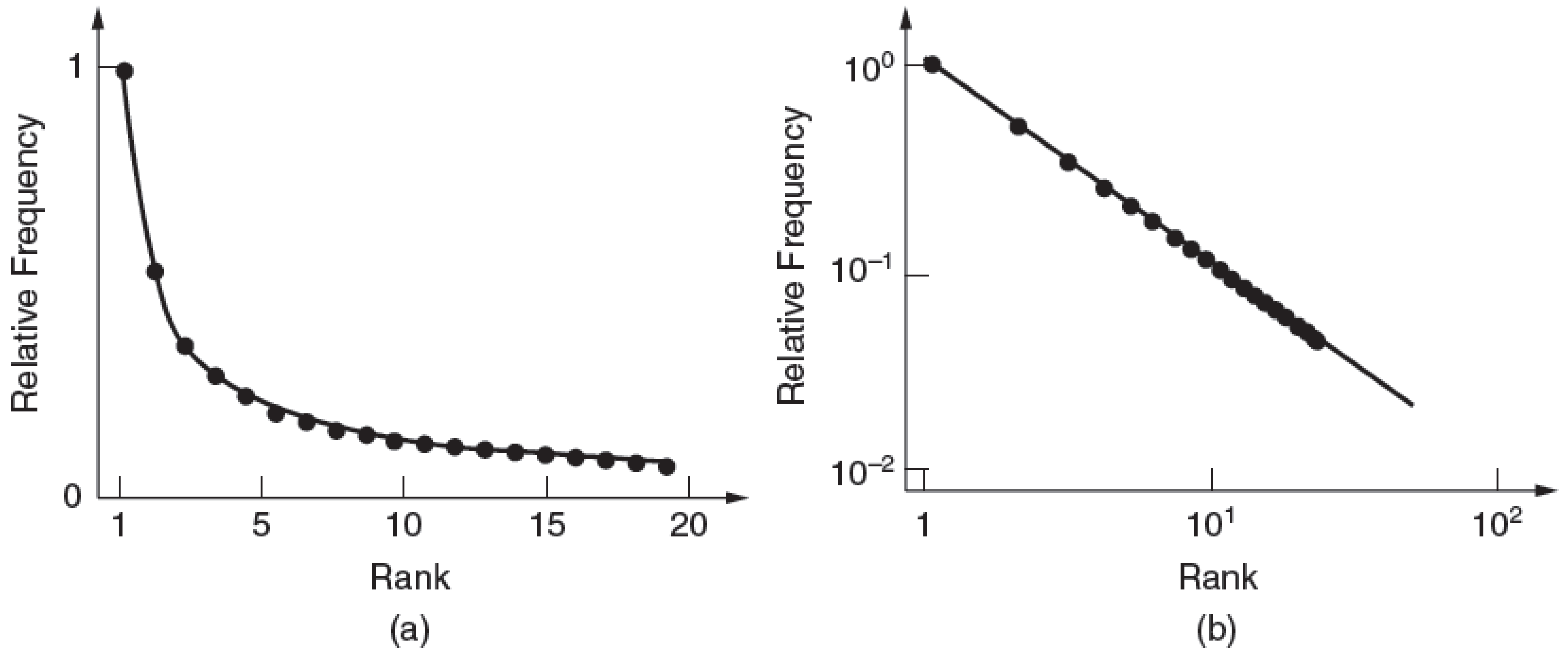
Module	Req.?	Function	Example tags
Structure	Yes	Doc. structure	body, head, html, title
Text	Yes	Information	br, code, dfn, em, h*n*, kbd, p, strong
Hypertext	Yes	Hyperlinks	a
List	Yes	Itemized lists	dl, dt, dd, ol, ul, li
Forms	No	Fill-in forms	form, input, label, option, textarea
Tables	No	Rectangular tables	caption, table, td, th, tr
Image	No	Pictures	img
Object	No	Applets, maps, etc.	object, param
Meta-information	No	Extra info	meta
Link	No	Similar to <a>	link
Base	No	URL starting point	base

The XHTML Basic modules and tags.

Content Delivery

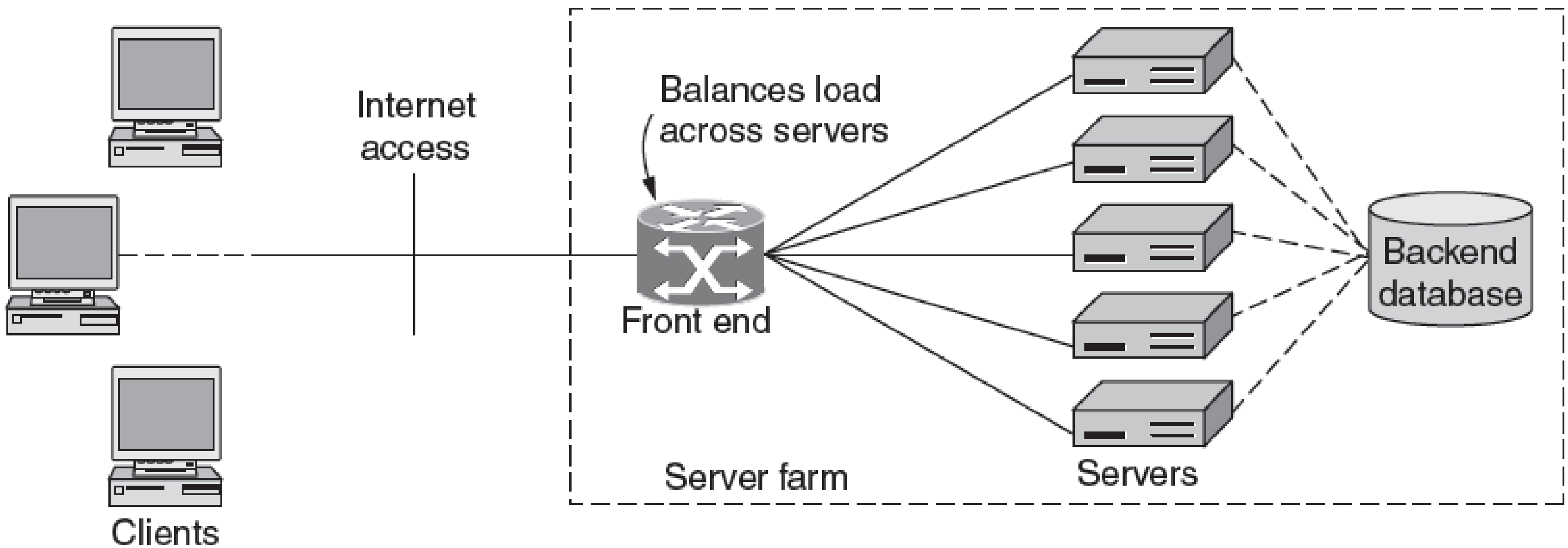
- Content and internet traffic
- Server farms and web proxies
- Content delivery networks
- Peer-to-peer networks

Content and Internet Traffic



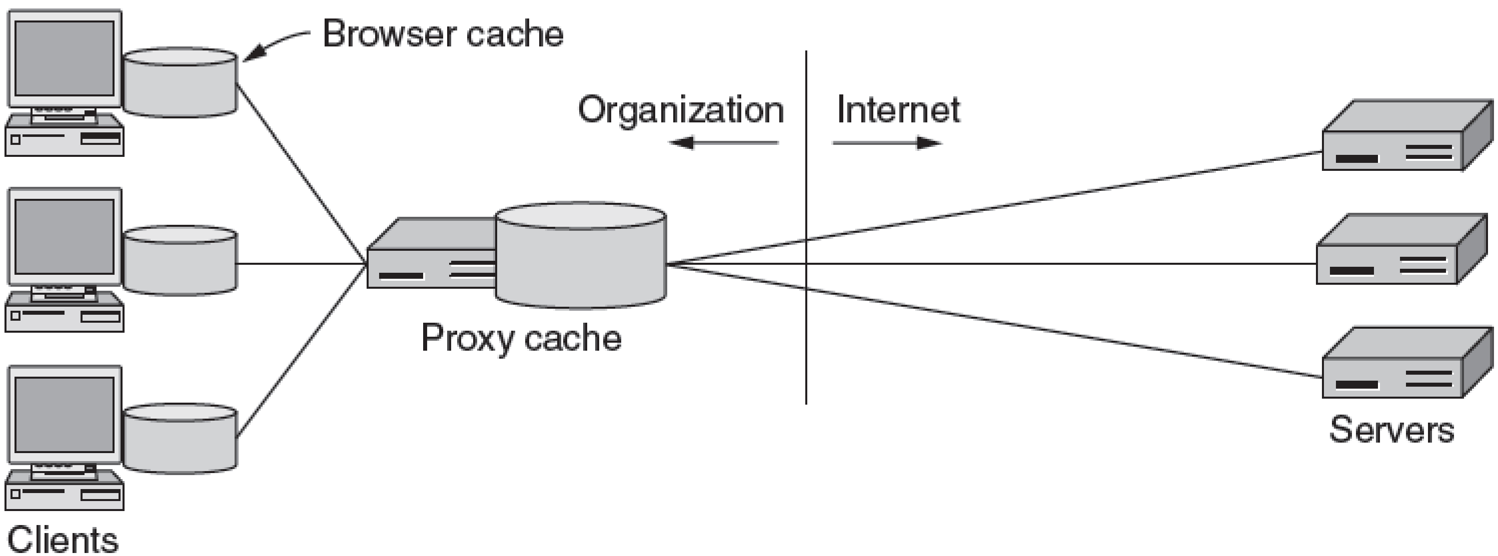
Zipf distribution (a) On a linear scale. (b) On a log-log scale.

Server Farms and Web Proxies (1)



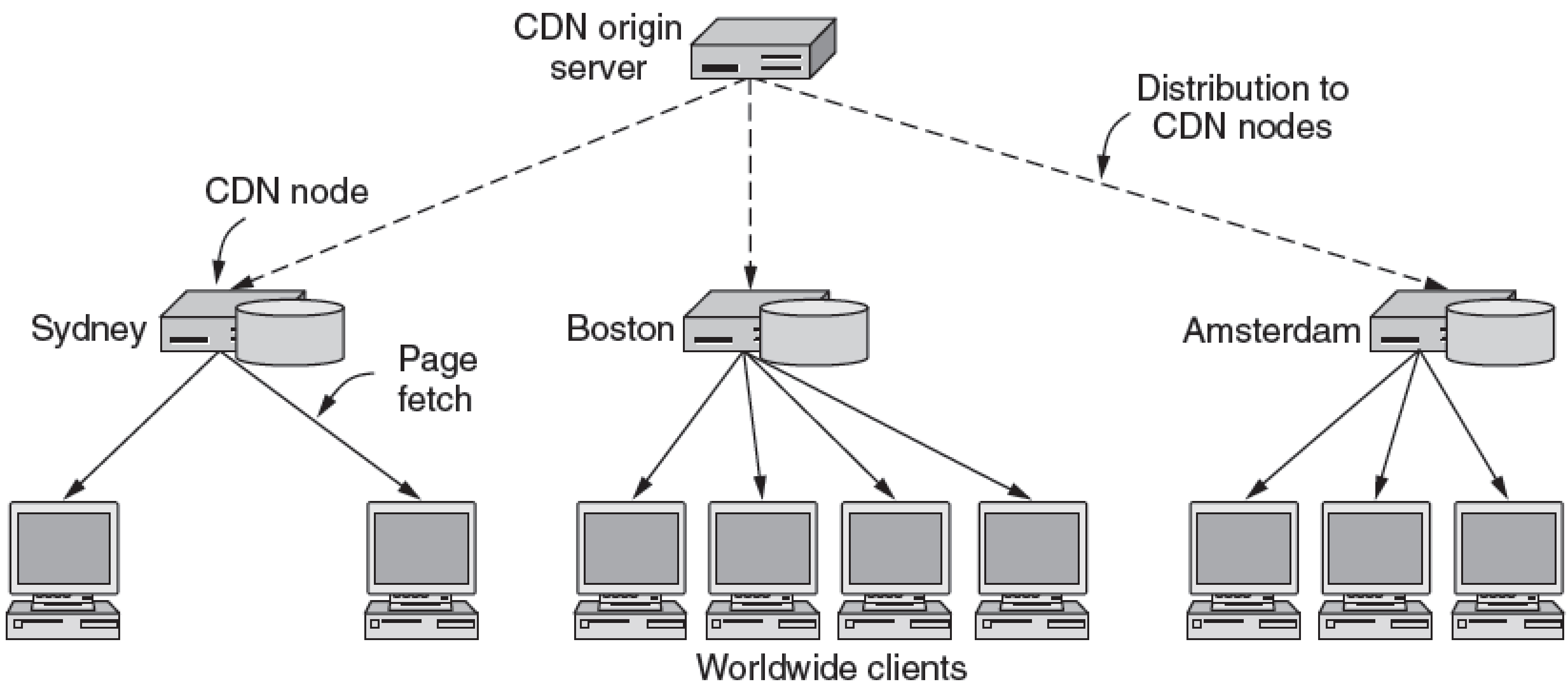
A server farm.

Server Farms and Web Proxies (2)



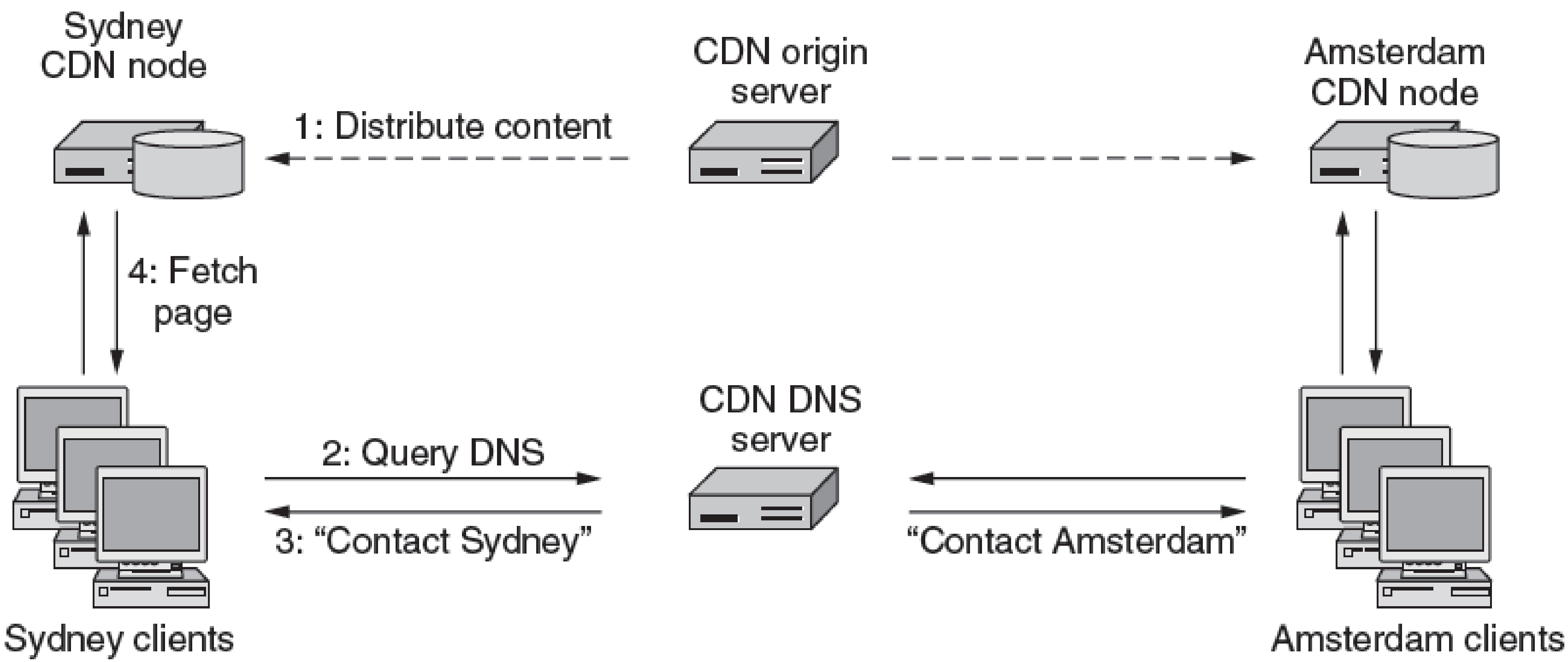
A proxy cache between Web browsers and Web servers.

Content Delivery Networks (1)



CDN distribution tree.

Content Delivery Networks (2)



Directing clients to nearby CDN nodes using DNS.

Content Delivery Networks (3)

```
<html>
<head> <title> Fluffy Video </title> </head>
<body>
<h1> Fluffy Video's Product List </h1>
<p> Click below for free samples. </p>

<a href="koalas.mpg"> Koalas Today </a> <br>
<a href="kangaroos.mpg"> Funny Kangaroos </a> <br>
<a href="wombats.mpg"> Nice Wombats </a> <br>
</body>
</html>
```

(a)

```
<html>
<head> <title> Fluffy Video </title> </head>
<body>
<h1> Fluffy Video's Product List </h1>
<p> Click below for free samples. </p>

<a href="http://www.cdn.com/fluffyvideo/koalas.mpg"> Koalas Today </a> <br>
<a href="http://www.cdn.com/fluffyvideo/kangaroos.mpg"> Funny Kangaroos </a> <br>
<a href="http://www.cdn.com/fluffyvideo/wombats.mpg"> Nice Wombats </a> <br>
</body>
</html>
```

(b)

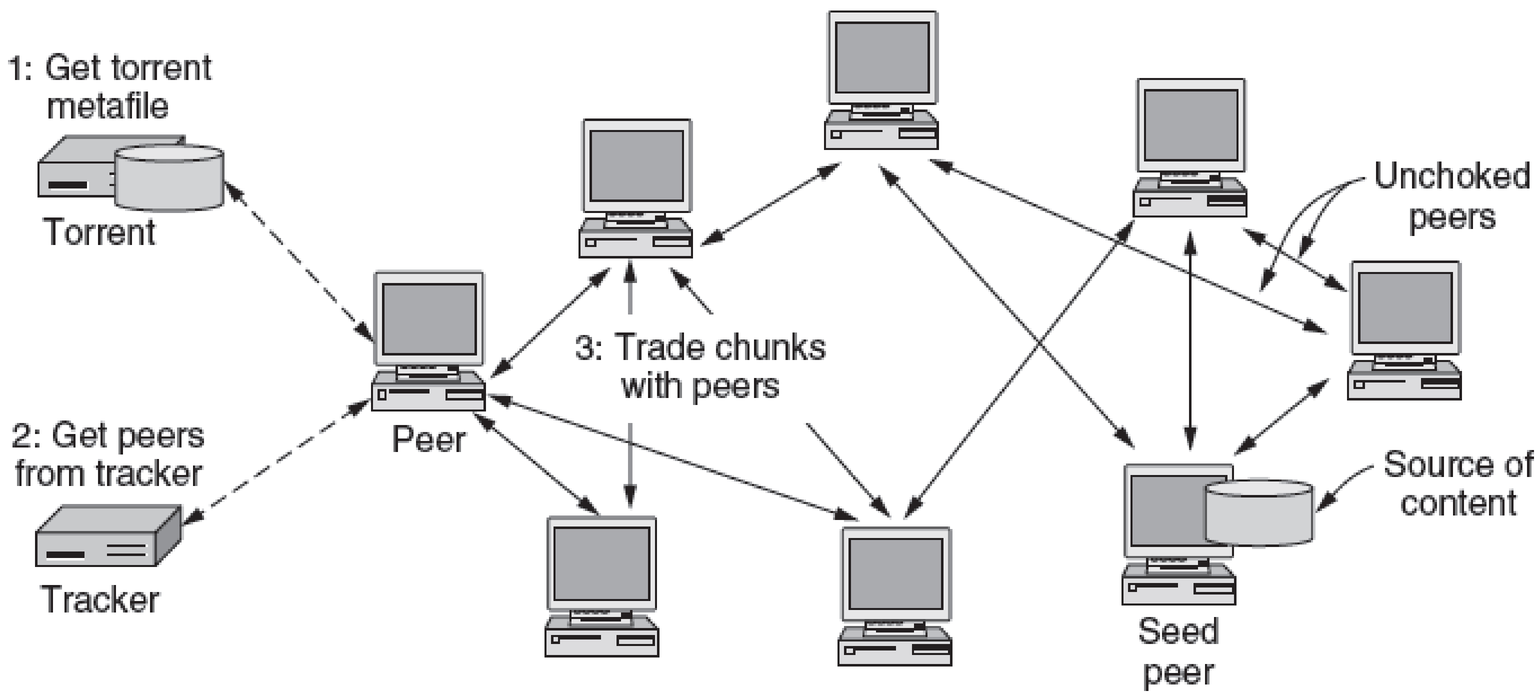
(a) Original Web page. (b) Same page after linking to the CDN

Peer-to-Peer Networks (1)

Problems to be solved with BitTorrent sharing

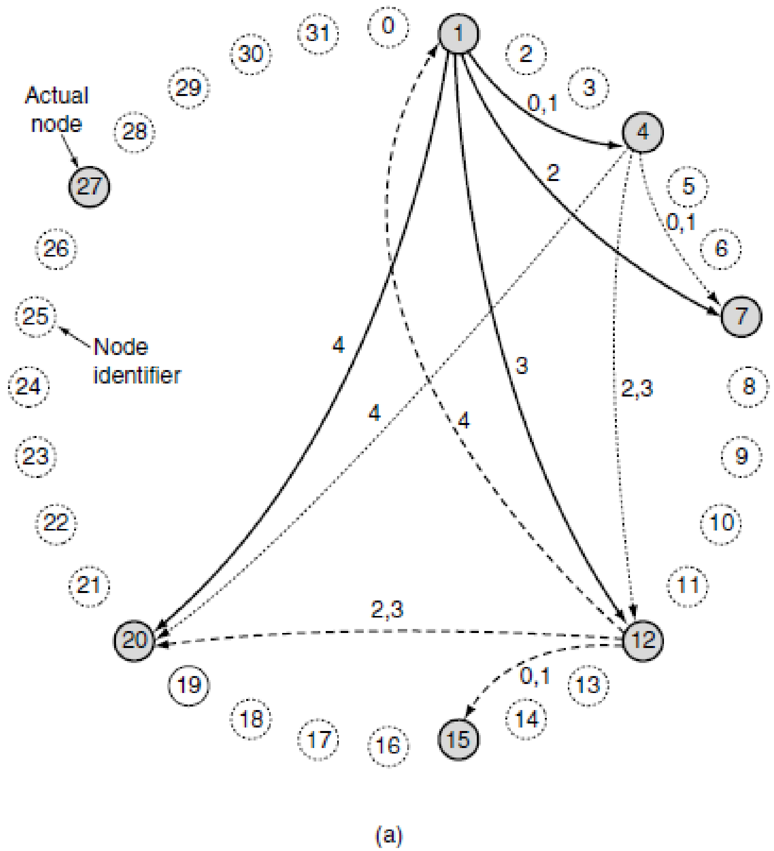
1. How does a peer find other peers
2. How is content replicated by peers to provide high-speed downloads
3. How do peers encourage each other to upload content to others

Peer-to-Peer Networks (2)



BitTorrent.

Peer-to-Peer Networks (3)



Start	IP addr of successor
2	4
3	4
5	7
9	12
17	20

Node 1's finger table

Start	IP addr of successor
5	7
6	7
8	12
12	12
20	20

Node 4's finger table

Start	IP addr of successor
13	15
14	15
16	20
20	20
28	1

Node 12's finger table

(b)

(a) A set of 32 node identifiers arranged in a circle. The shaded ones correspond to actual machines. The arcs show the fingers from nodes 1, 4 and 12. The labels on the arcs are the table indices. (b) Examples of the finger tables.

Content Delivery Networks

```
<html>
<head> <title> Furry Video </title> </head>
<body>
<h1> Furry Video's Product List </h1>
<p> Click below for free samples. </p>

<a href="bears.mpg"> Bears Today </a> <br>
<a href="bunnies.mpg"> Funny Bunnies </a> <br>
<a href="mice.mpg"> Nice Mice </a> <br>
</body>
</html>
```

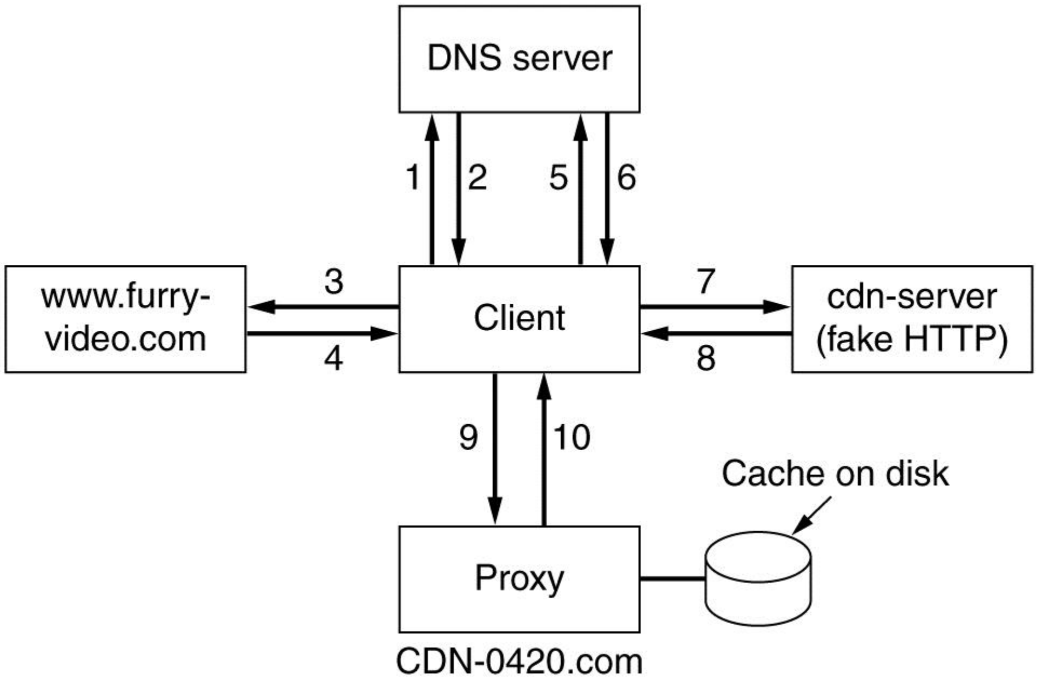
(a)

```
<html>
<head> <title> Furry Video </title> </head>
<body>
<h1> Furry Video's Product List </h1>
<p> Click below for free samples. </p>

<a href="http://cdn-server.com/furryvideo/bears.mpg"> Bears Today </a> <br>
<a href="http://cdn-server.com/furryvideo/bunnies.mpg"> Funny Bunnies </a> <br>
<a href="http://cdn-server.com/furryvideo/mice.mpg"> Nice Mice </a> <br>
</body>
</html>
```

(b)

(a) Original Web page. (b) Same page after transformation.

The Wireless Web



1. Look up www.furryvideo.com
2. Furry's IP address returned
3. Request HTML page from Furry
4. HTML page returned
5. After click, look up cdn-server.com
6. IP address of cdn-server returned
7. Ask cdn-server for bears.mpg
8. Client told to redirect to CDN-0420.com
9. Request bears.mpg
10. Cached file bears.mpg returned

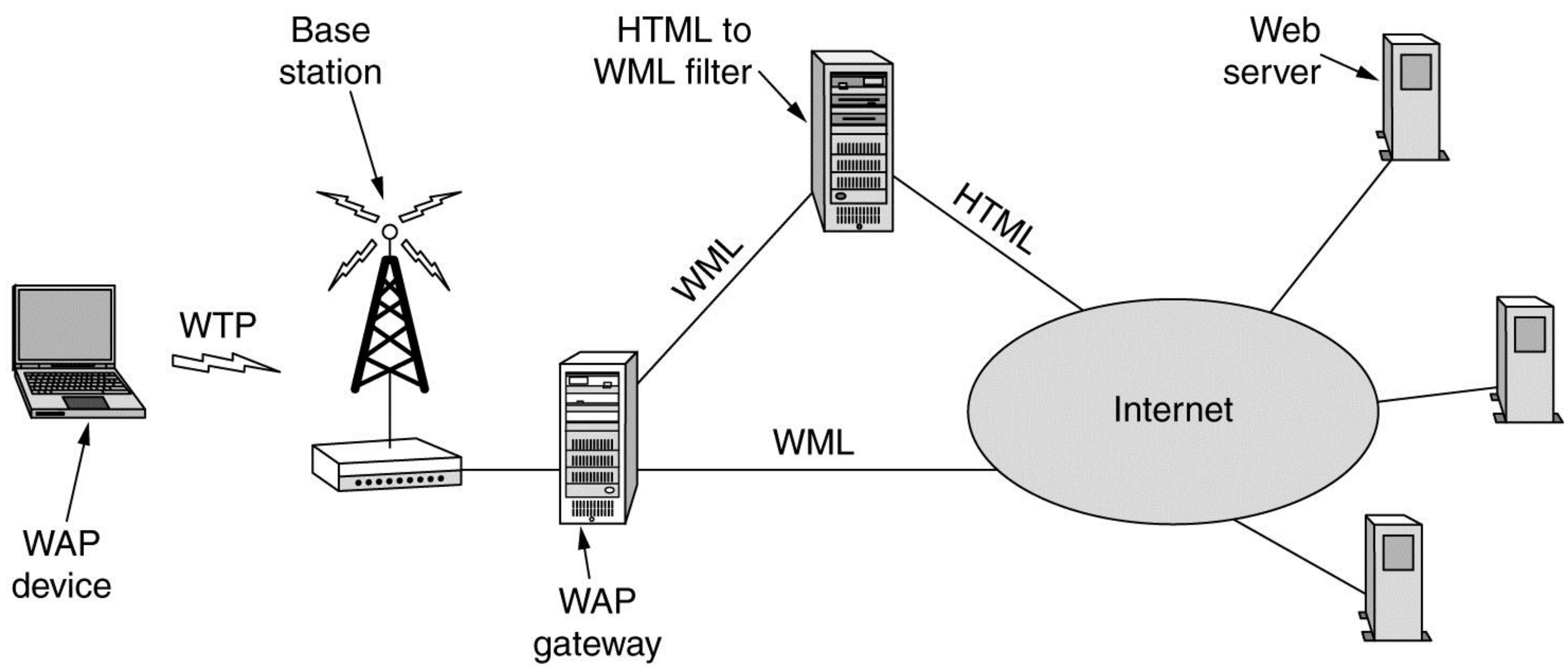
WAP – The Wireless Application Protocol

The WAP protocol stack.

Wireless application environment (WAE)
Wireless session protocol (WSP)
Wireless transaction protocol (WTP)
Wireless transport layer security (WTLS)
Wireless datagram protocol (WDP)
Bearer layer (GSM, CDMA, D-AMPS, GPRS, etc.)

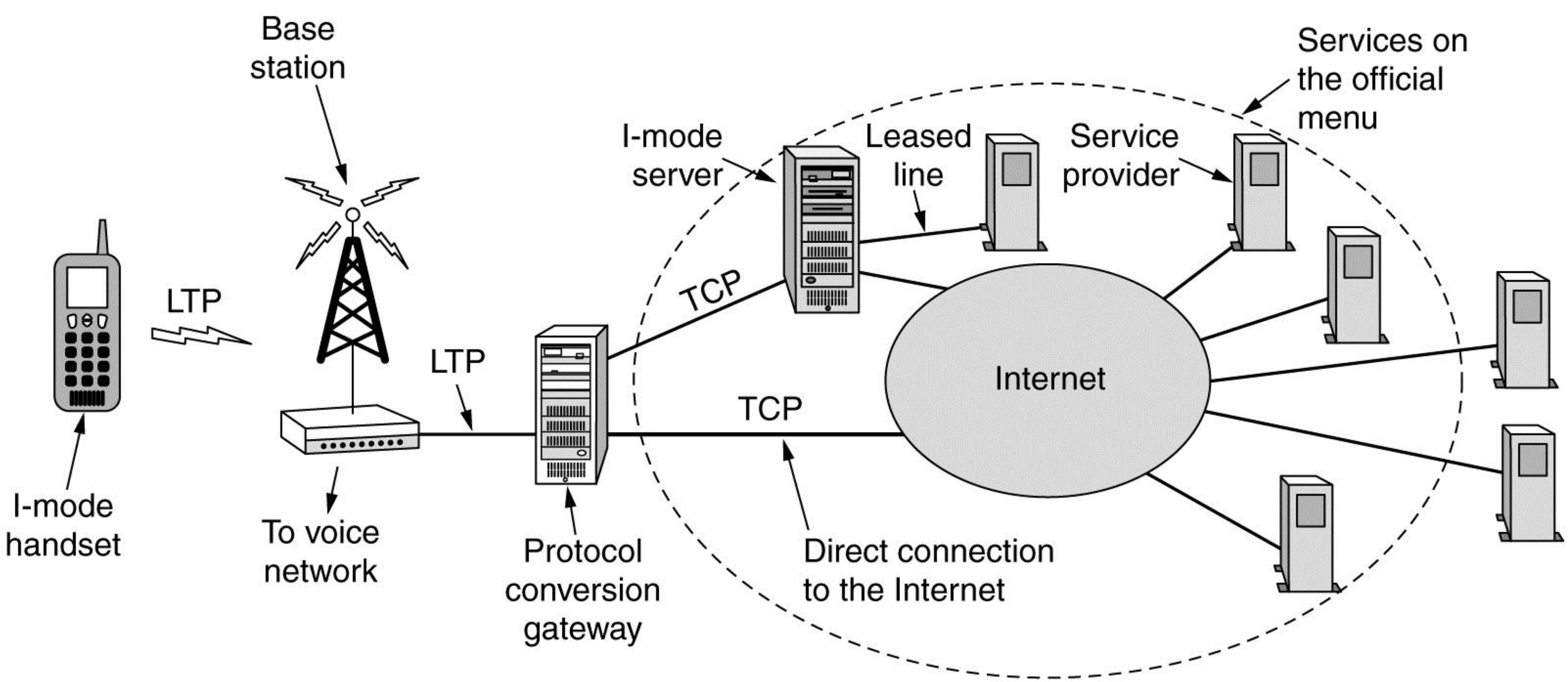
WAP (2)

The WAP architecture.



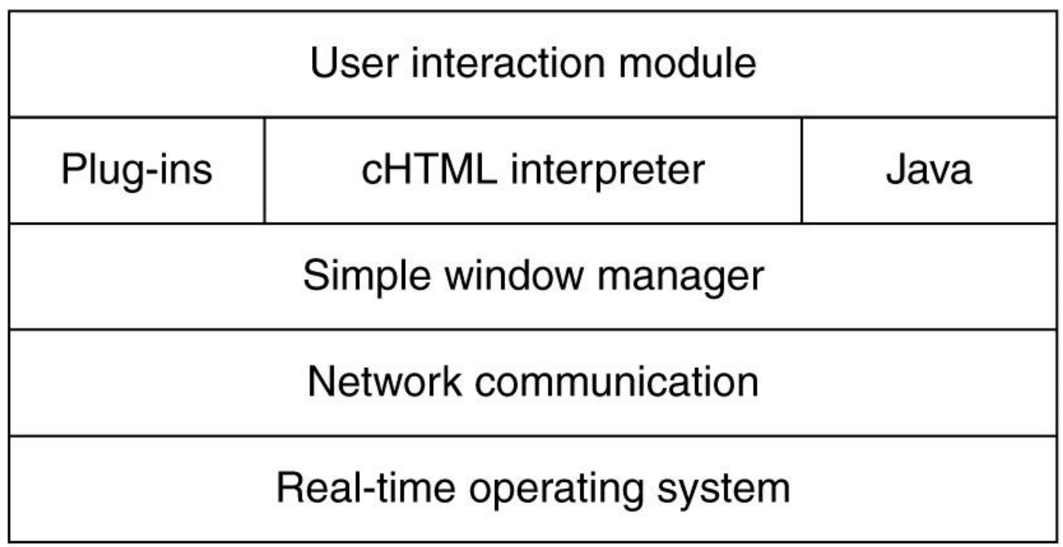
I-Mode

Structure of the i-mode data network showing the transport protocols.



I-Mode (2)

Structure of the i-mode software.



I-Mode (3)

Lewis Carroll meets a 16 x 16 screen.

```
The time has com
e the walrus sai
d to talk of man
y things. Of sho
es and ships and
sealing wax of c
```

(a)

```
The time has
come the walrus
said to talk of
many things. Of
shoes and ships
and sealing wax
```

(b)

I-Mode (4)

An example of cHTML file.

```html
<html>
<body>
<h1> Select an option </h1>
<a href="messages.chtml" accesskey="1"> Check voicemail </a> <br>
<a href="mail.chtml" accesskey="2"> Check e-mail </a> <br>
<a href="games.chtml" accesskey="3"> Play a game </a>
</body>
</html>
```

Second-Generation Wireless Web

A comparison of first-generation WAP and i-mode.

Feature	WAP	I-mode
What it is	Protocol stack	Service
Device	Handset, PDA, notebook	Handset
Access	Dial up	Always on
Underlying network	Circuit-switched	Two: circuit + packet
Data rate	9600 bps	9600 bps
Screen	Monochrome	Color
Markup language	WML (XML application)	cHTML
Scripting language	WMLscript	None
Usage charges	Per minute	Per packet
Pay for shopping	Credit card	Phone bill
Pictograms	No	Yes
Standardization	WAP forum open standard	NTT DoCoMo proprietary
Where used	Europe, Japan	Japan
Typical user	Businessman	Young woman

Second-Generation Wireless Web (2)

New features of WAP 2.0.

- Push model as well as pull model.
- Support for integrating telephony into apps.
- Multimedia messaging.
- Inclusion of 264 pictograms.
- Interface to a storage device.
- Support for plug-ins in the browser.

Second-Generation Wireless Web (3)

WAP 2.0 supports two protocol stacks.

XHTML	
WSP	HTTP
WTP	TLS
WTLS	TCP
WDP	IP
Bearer layer	Bearer layer
WAP 1.0 protocol stack	WAP 2.0 protocol stack

Second-Generation Wireless Web (4)

The XHTML Basic modules and tags.

Module	Req.?	Function	Example tags
Structure	yes	Doc. structure	body, head, html, title
Text	yes	Information	br, code, dfn, em, h*n*, kbd, p, strong
Hypertext	yes	Hyperlinks	a
List	yes	Itemized lists	dl, dt, dd, ol, ul, li
Forms	No	Fill-in forms	form, input, label, option, textarea
Tables	No	Rectangular tables	caption, table, td, th, tr
Image	No	Pictures	img
Object	No	Applets, maps, etc.	object, param
Meta-information	No	Extra info	meta
Link	No	Similar to <a>	link
Base	No	URL starting point	base

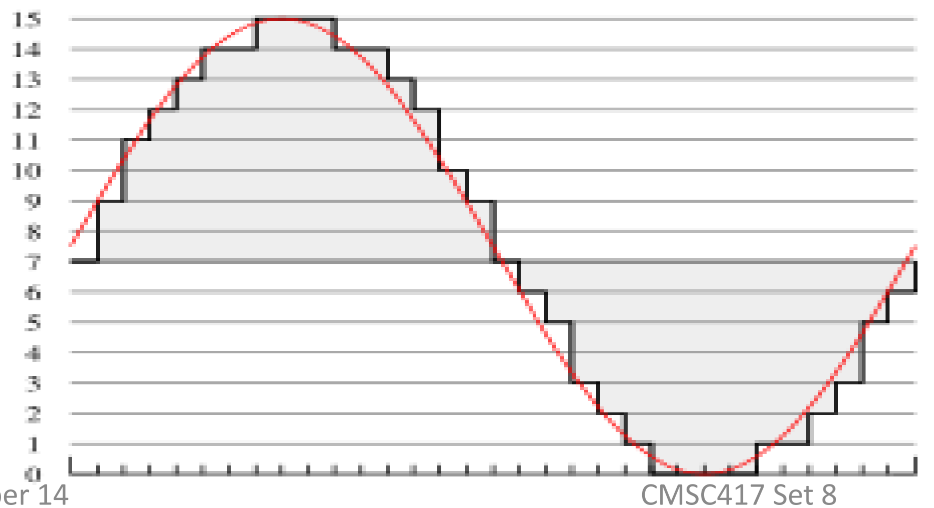
Multimedia Networking

Topics

- Digital audio and video
  - Sampling, quantizing, and compressing
- Multimedia applications
  - Streaming audio and video for playback
  - Live, interactive audio and video
- Multimedia transfers over a best-effort network
  - Tolerating packet loss, delay, and jitter
  - Forward error correction and playout buffers
- Improving the service the network offers
  - Marking, policing, scheduling, and admission control

Digital Audio

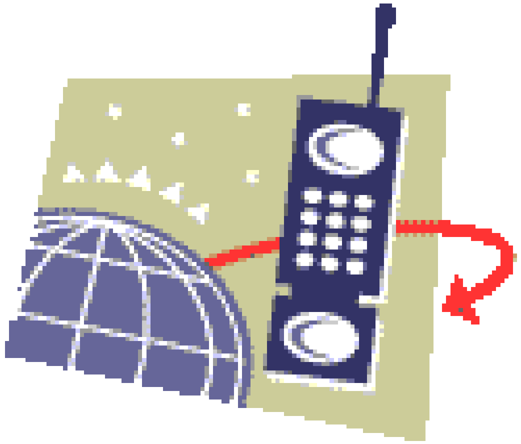
- Sampling the analog signal
  - Sample at some fixed rate
  - Each sample is an arbitrary real number
- Quantizing each sample
  - Round each sample to one of a finite number of values
  - Represent each sample in a fixed number of bits



4 bit representation
(values 0-15)

Audio Examples

- Speech
  - Sampling rate: 8000 samples/second
  - Sample size: 8 bits per sample
  - Rate: 64 kbps



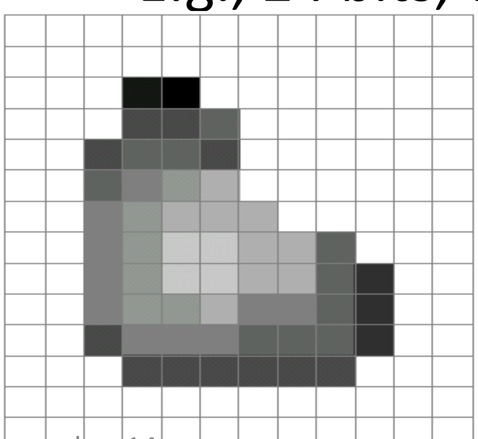
- Compact Disc (CD)
  - Sampling rate: 44,100 samples/second
  - Sample size: 16 bits per sample
  - Rate: 705.6 kbps for mono,
    1.411 Mbps for stereo

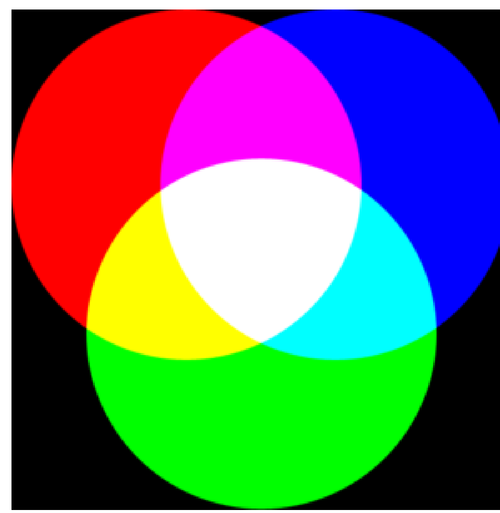
Audio Compression

- Audio data requires too much bandwidth
  - Speech: 64 kbps is too high for a dial-up modem user
  - Stereo music: 1.411 Mbps exceeds most access rates
- Compression to reduce the size
  - Remove redundancy
  - Remove details that human tend not to perceive
- Example audio formats
  - Speech: GSM (13 kbps), G.729 (8 kbps), and G.723.3 (6.4 and 5.3 kbps)
  - Stereo music: MPEG 1 layer 3 (MP3) at 96 kbps, 128 kbps, and 160 kbps

Digital Video

- Sampling the analog signal
  - Sample at some fixed rate (e.g., 24 or 30 times per sec)
  - Each sample is an image
- Quantizing each sample
  - Representing an image as an array of picture elements
  - Each pixel is a mixture of colors (red, green, and blue)
  - E.g., 24 bits, with 8 bits per color






The
2272 x 1704
hand
The
320 x 240
hand

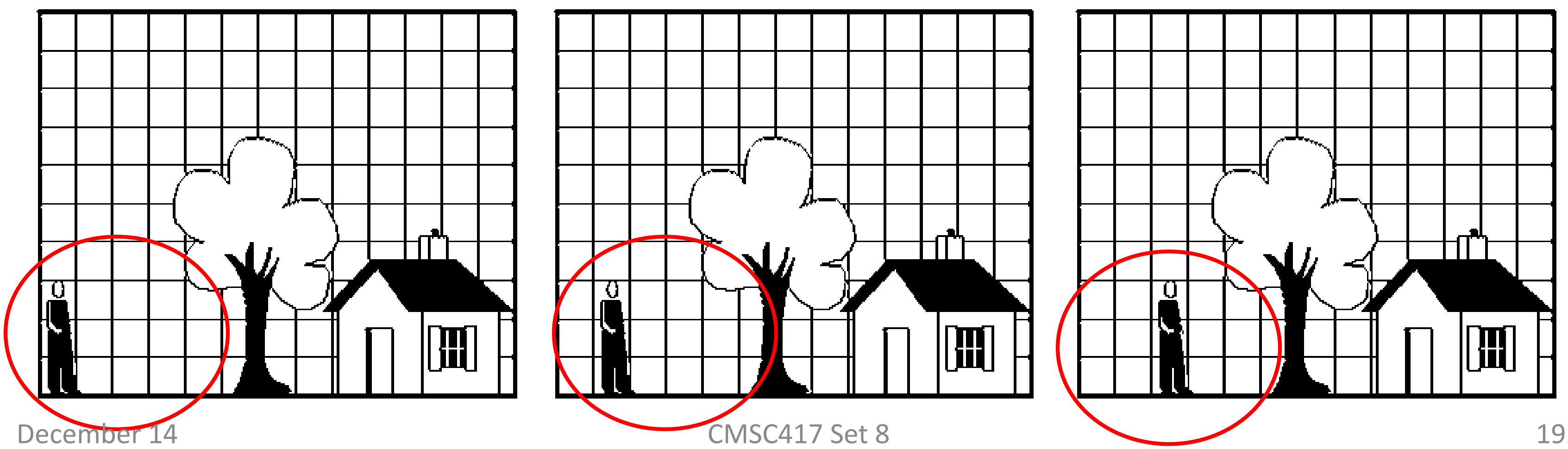
Video Compression: Within an Image

- Image compression
  - Exploit spatial redundancy (e.g., regions of same color)
  - Exploit aspects humans tend not to notice
- Common image compression formats
  - Joint Pictures Expert Group (JPEG)
  - Graphical Interchange Format (GIF)

Uncompressed: 167 KB   Good quality: 46 KB   Poor quality: 9 KB

Video Compression: Across Images

- Compression across images
  - Exploit temporal redundancy across images
- Common video compression formats
  - MPEG 1: CD-ROM quality video (1.5 Mbps)
  - MPEG 2: high-quality DVD video (3-6 Mbps)
  - Proprietary protocols like QuickTime and RealNetworks



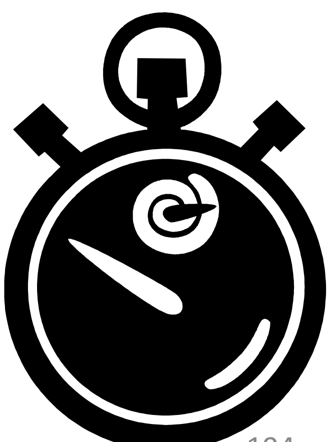
Transferring Audio and Video Data

- Simplest case: just like any other file
  - Audio and video data stored in a file
  - File downloaded using conventional protocol
  - Playback does not overlap with data transfer
- A variety of more interesting scenarios
  - Live vs. pre-recorded content
  - Interactive vs. non-interactive
  - Single receiver vs. multiple receivers

- Examples
  - Streaming audio and video data from a server
  - Interactive audio in a phone call

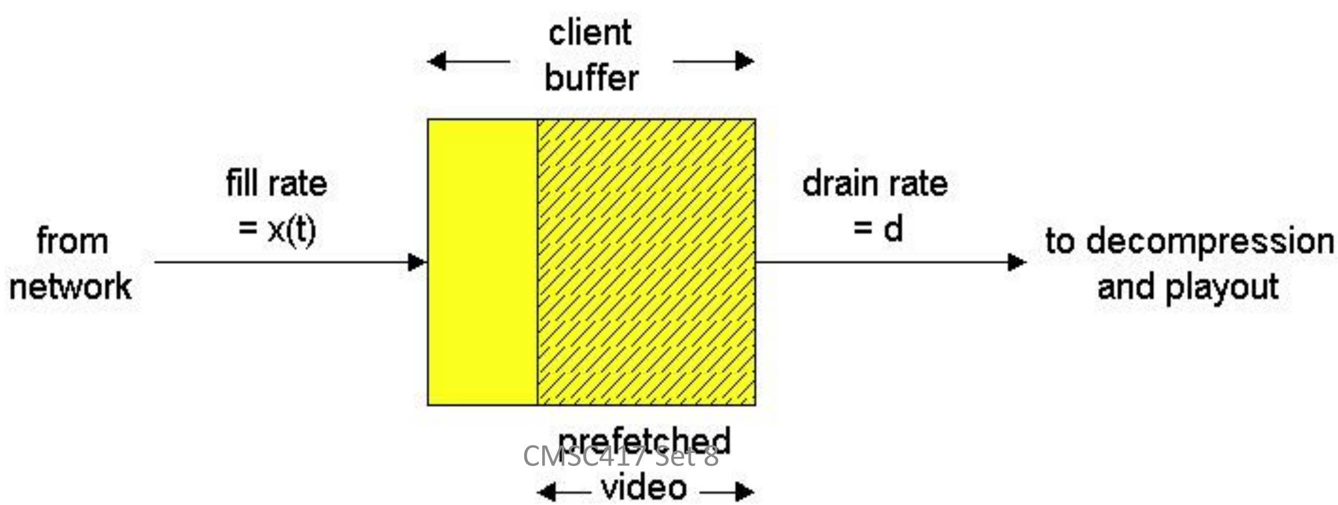
Streaming Stored Audio and Video

- Client-server system
  - Server stores the audio and video files
  - Clients request files, play them as they download, and perform VCR-like functions (e.g., rewind and pause)
- Playing data at the right time
  - Server divides the data into segments
  - … and labels each segment with timestamp or frame id
  - … so the client knows when to play the data
- Avoiding starvation at the client
  - The data must arrive quickly enough
  - … otherwise the client cannot keep playing

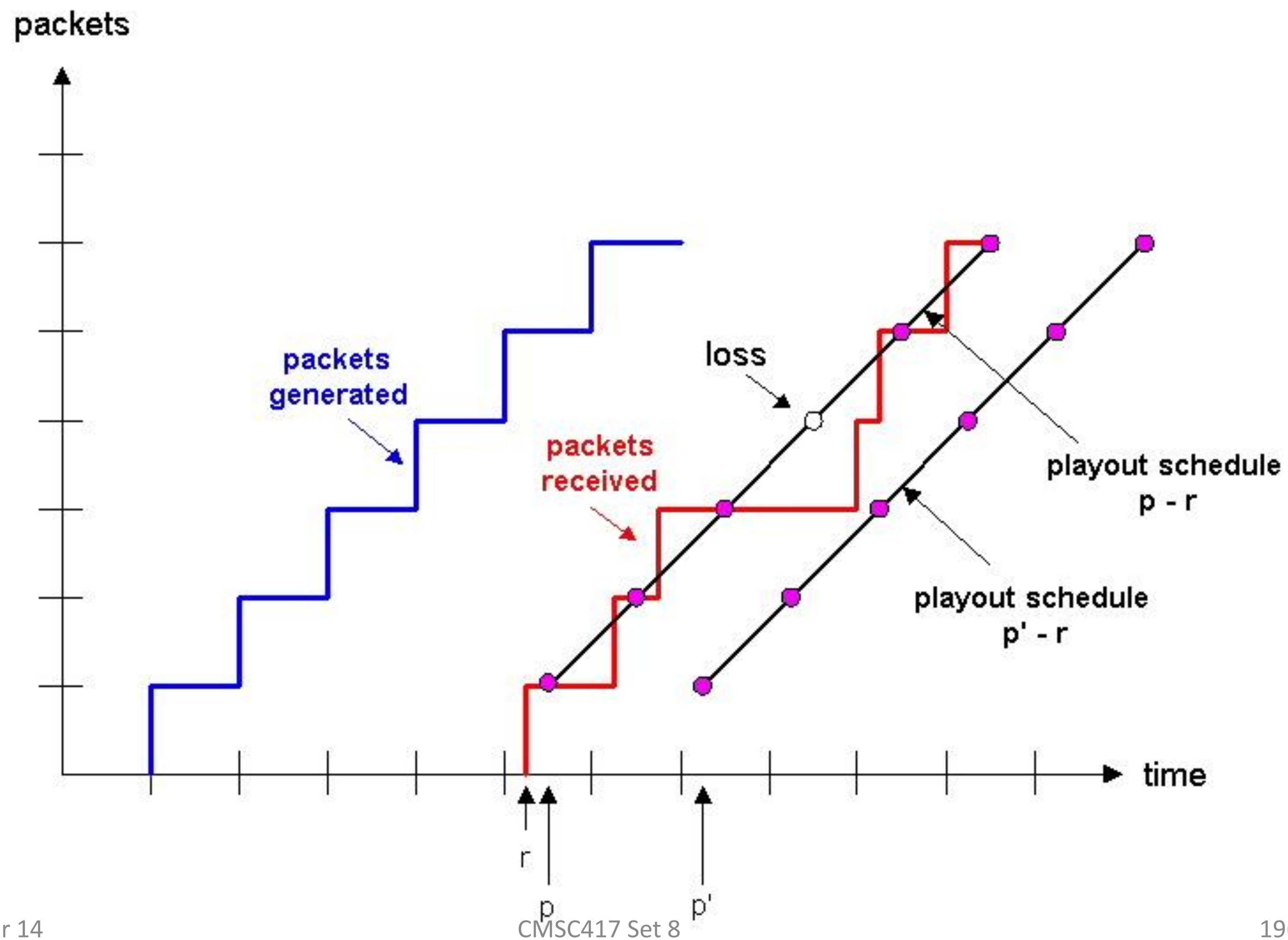


Playout Buffer

- Client buffer
  - Store the data as it arrives from the server
  - Play data for the user in a continuous fashion
- Playout delay
  - Client typically waits a few seconds to start playing
  - … to allow some data to build up in the buffer



Influence of Playout Delay

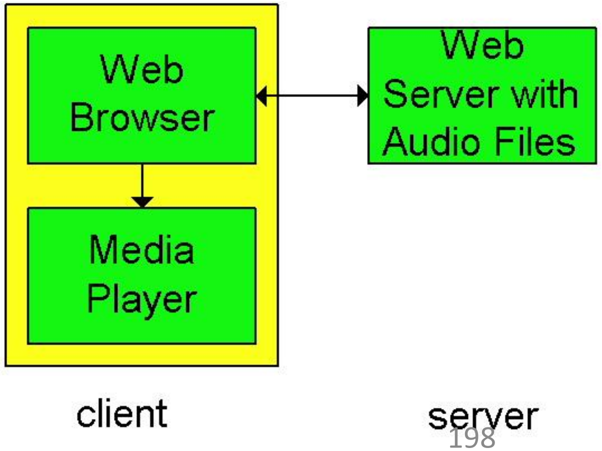


Requirements for Data Transport

- Delay
  - Some small delay at the beginning is acceptable
  - E.g., start-up delays of a few seconds are okay
- Jitter
  - Variability of packet delay within the same packet stream
  - Client cannot tolerate high variation if the buffer starves
- Loss
  - Small amount of missing data does not disrupt playback
  - Retransmitting a lost packet might take too long anyway

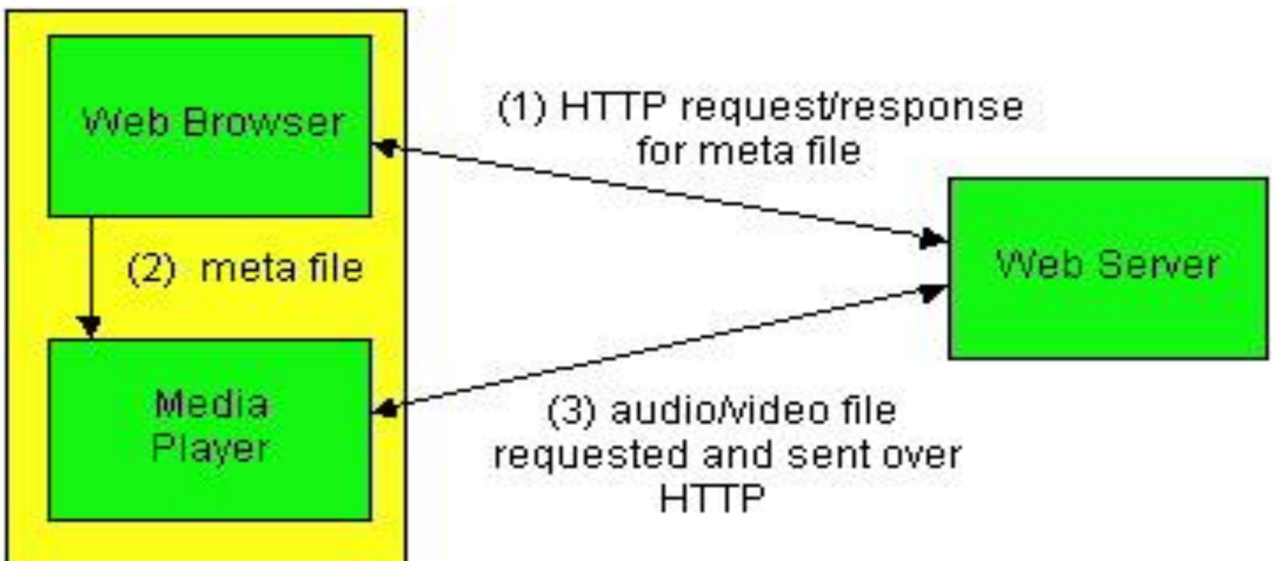
Streaming From Web Servers

- Data stored in a file
  - Audio: an audio file
  - Video: interleaving of audio and images in a single file
- HTTP request-response
  - TCP connection between client and server
  - Client HTTP request and server HTTP response
- Client invokes the media player
  - Content-type indicates the encoding
  - Browser launches the media player
  - Media player then renders the file



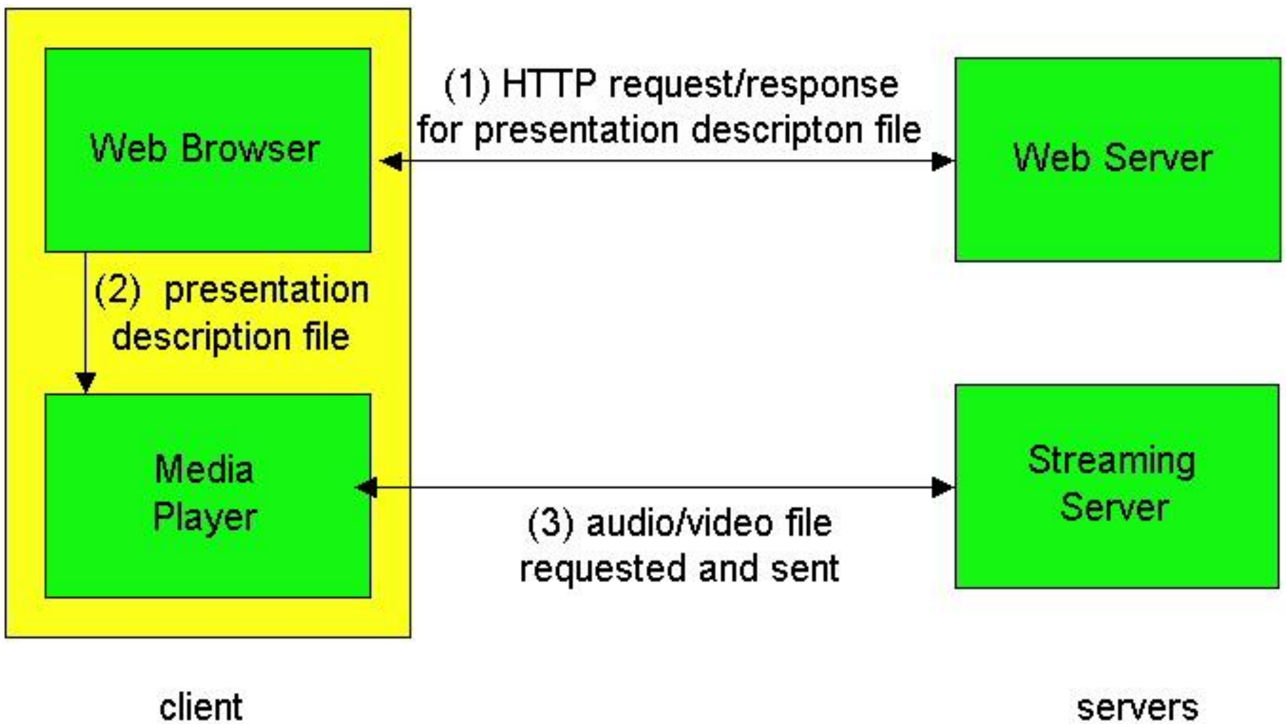
Initiating Streams from Web Servers

- Avoid passing all data through the Web browser
  - Web server returns a meta file describing the object
  - Browser launches media player and passes the meta file
  - The player sets up its own connection to the Web server



Using a Streaming Server

- Avoiding the use of HTTP (and perhaps TCP, too)
  - Web server returns a meta file describing the object
  - Player requests the data using a different protocol



TCP is Not a Good Fit

- Reliable delivery
  - Retransmission of lost packets
  - … even though it may not be useful
- Adapting the sending rate
  - Slowing down after a packet loss
  - … even though it may cause starvation at the client
- Protocol overhead
  - TCP header of 20 bytes in every packet
  - … which is large for sending audio samples
  - Sending ACKs for every other packet
  - … which may be more feedback than needed

Better Ways of Transporting Data

- User Datagram Protocol (UDP)
  - No automatic retransmission of lost packets
  - No automatic adaptation of sending rate
  - Smaller packet header

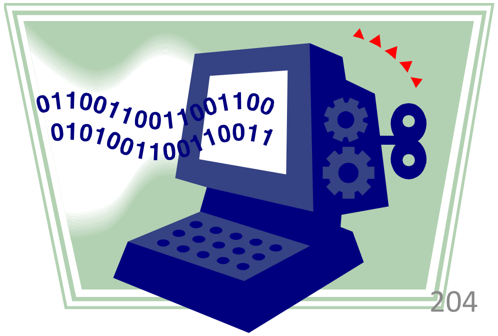
- UDP leaves many things up to the application
  - When to transmit the data
  - How to encapsulate the data
  - Whether to retransmit lost data
  - Whether to adapt the sending rate
  - … or adapt the quality of the audio/video encoding

Recovering From Packet Loss

- Loss is defined in a broader sense
  - Does a packet arrive in time for playback?
  - A packet that arrives late is as good as lost
  - Retransmission is not useful if the deadline has passed
- Selective retransmission
  - Sometimes retransmission is acceptable
  - E.g., if the client has not already started playing the data
  - Data can be retransmitted within the time constraint

Forward Error Correction (FEC)

- Forward error correction
  - Add redundant information to the packet stream
  - So the client can reconstruct data even after a loss
- Send redundant chunk after every n chunks
  - E.g., extra chunk is an XOR of the other n chunks
  - Receiver can recover from losing a single chunk
- Send low-quality version along with high quality
  - E.g., 13 kbps audio along with 64 kbps version
  - Receiver can play low quality version if the high-quality version is lost

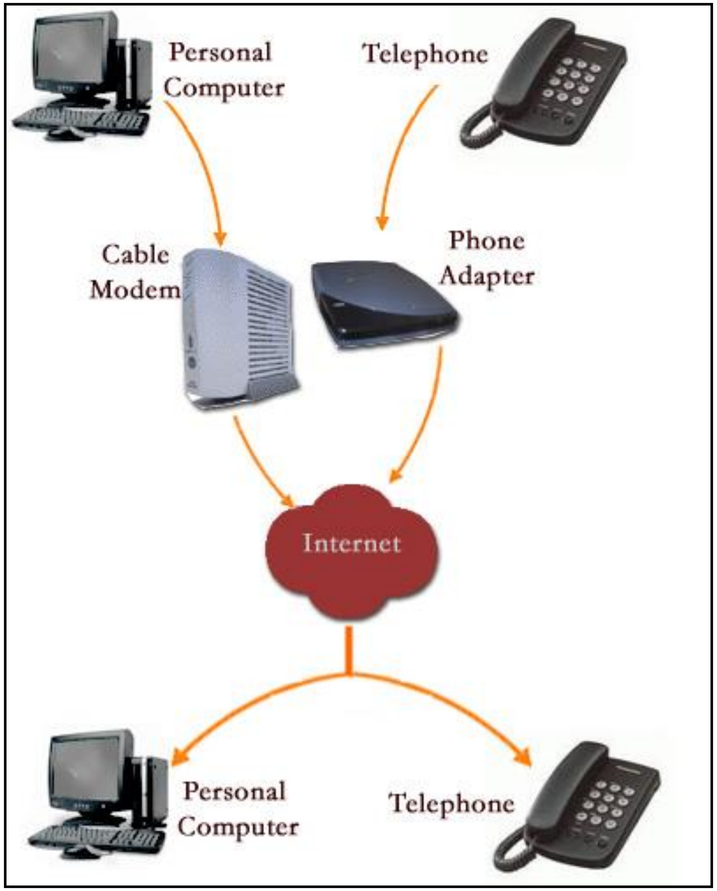


Interactive Audio and Video

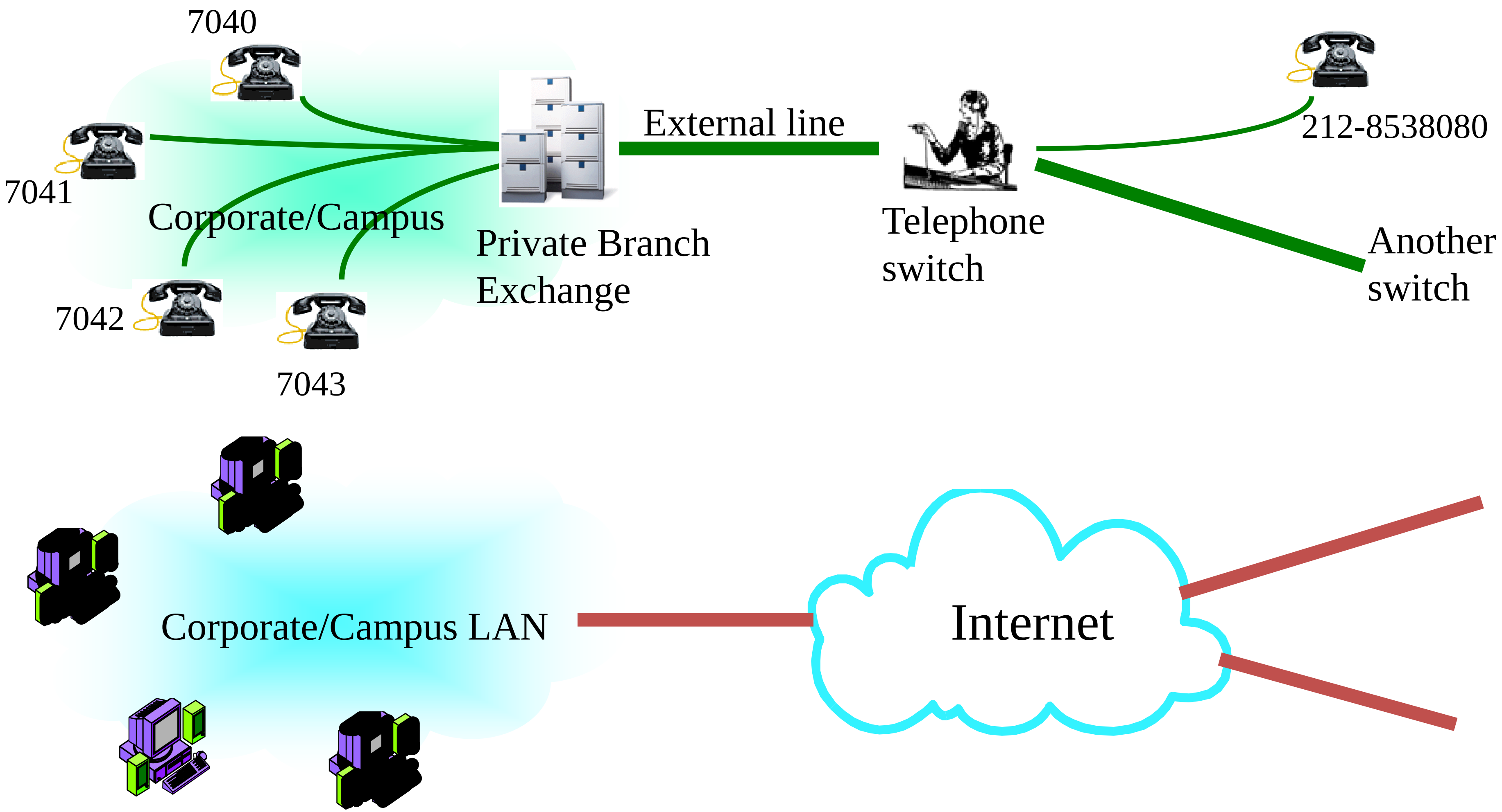
- Two or more users interacting
  - Telephone call
  - Video conference
  - Video game
- Strict delay constraints
  - Delays over 150-200 msec are very noticeable
  - … and delays over 400 msec are a disaster for voice
- Much harder than streaming applications
  - Receiver cannot introduce much playout delay
  - Difficult if the network does not guarantee performance

Voice Over IP (VoIP)

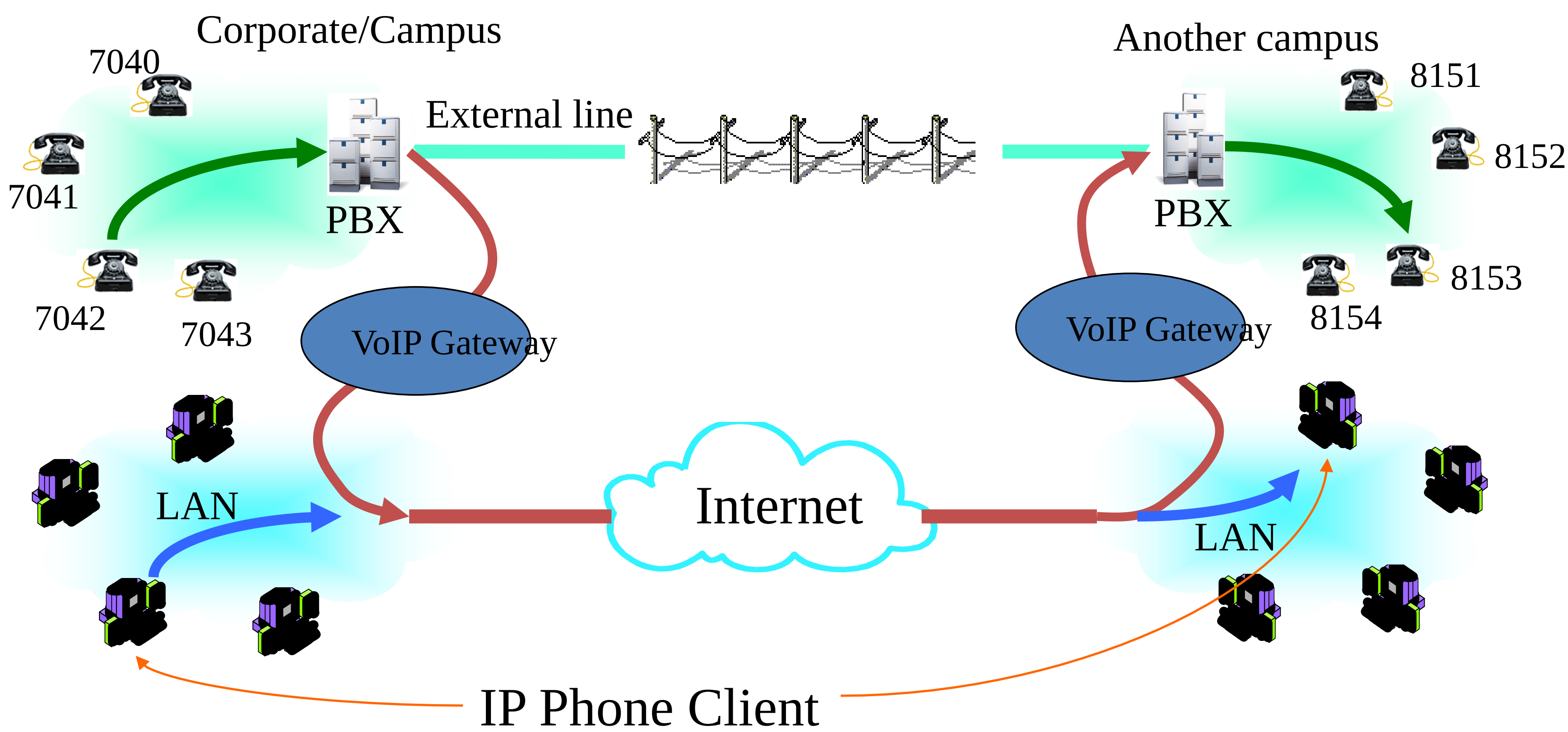
- Delivering phone calls over IP
  - Computer to computer
  - Analog phone to/from computer
  - Analog phone to analog phone
- Motivations for VoIP
  - Cost reduction
  - Simplicity
  - Advanced applications
    - Web-enabled call centers
    - Collaborative white boarding
    - Do Not Disturb, Locate Me, etc.
    - Voicemail sent as e-mail



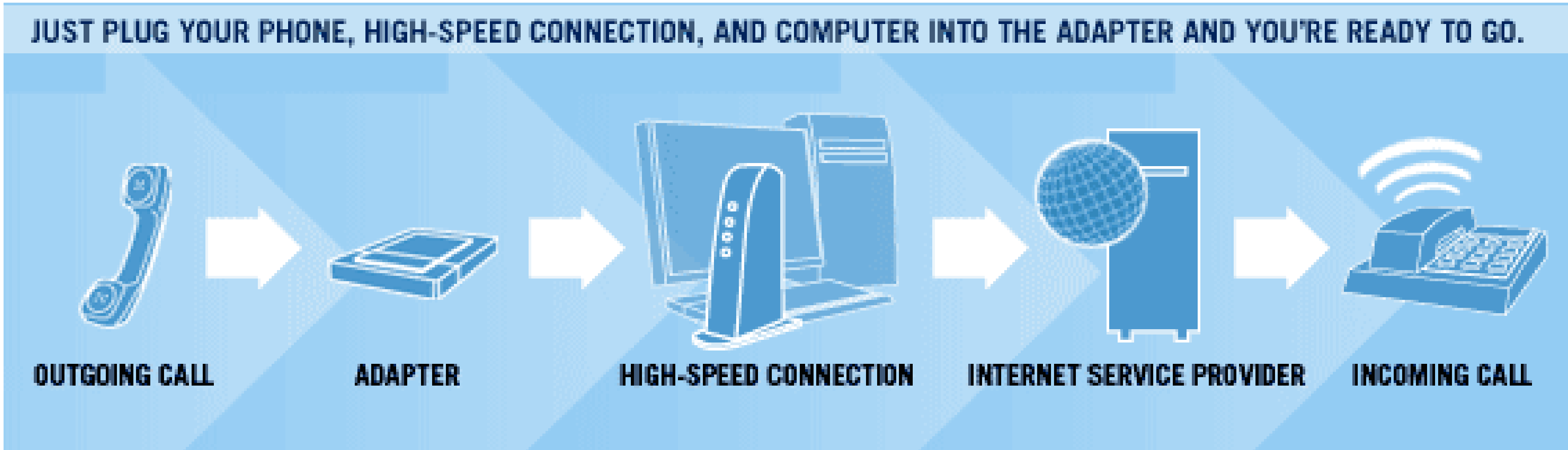
Traditional Telecom Infrastructure



VoIP Gateways



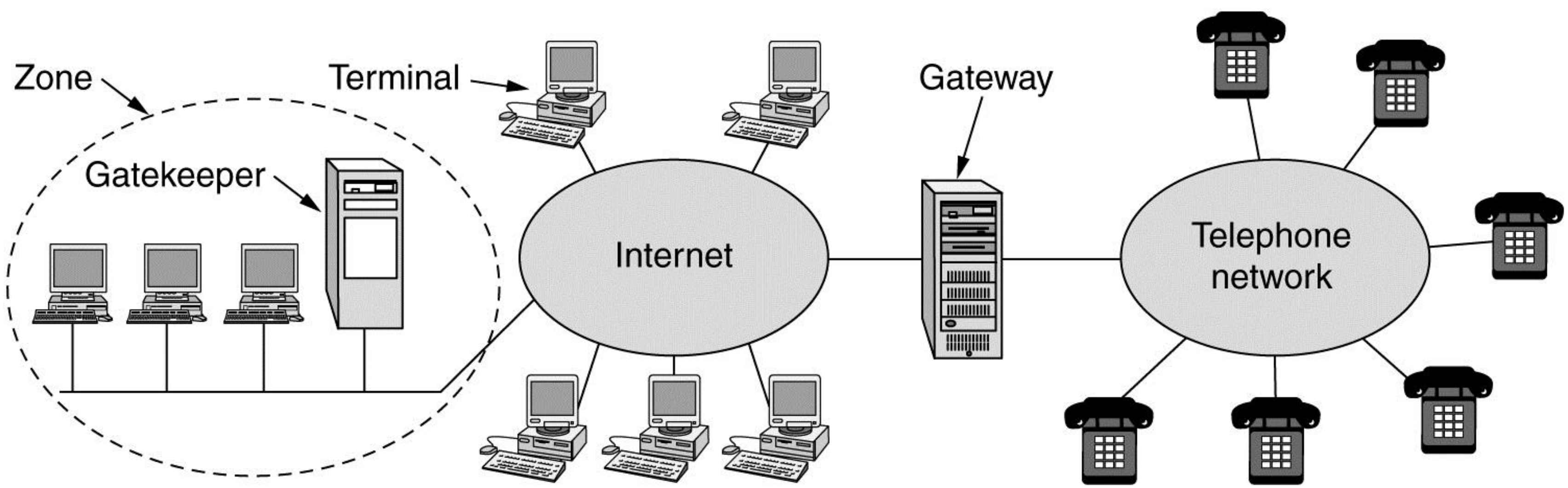
VoIP With an Analog Phone



- Adapter
  - Converts between analog and digital
  - Sends and receives data packets
  - Communicates with the phone in standard way

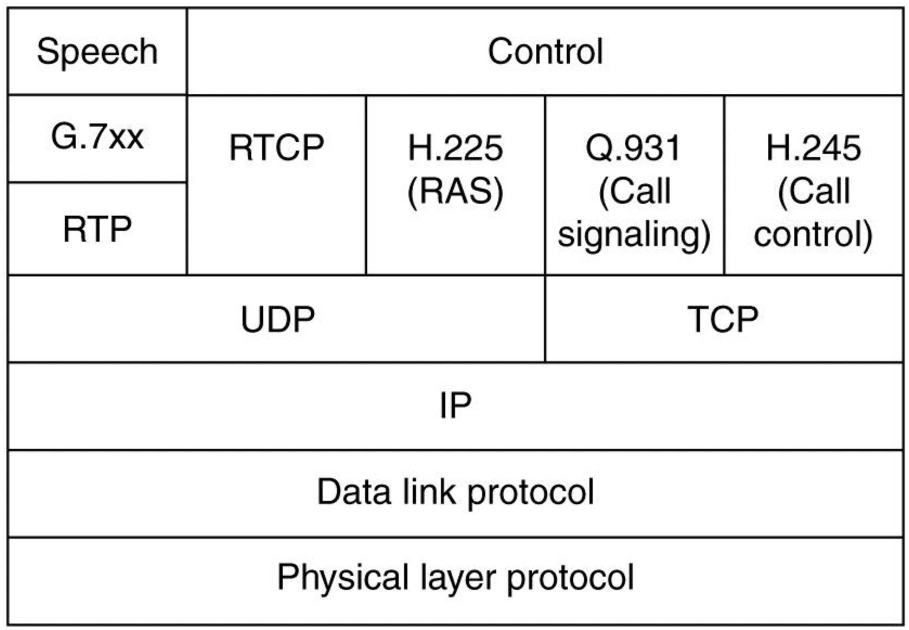
Voice over IP

The H323 architectural model for Internet telephony.



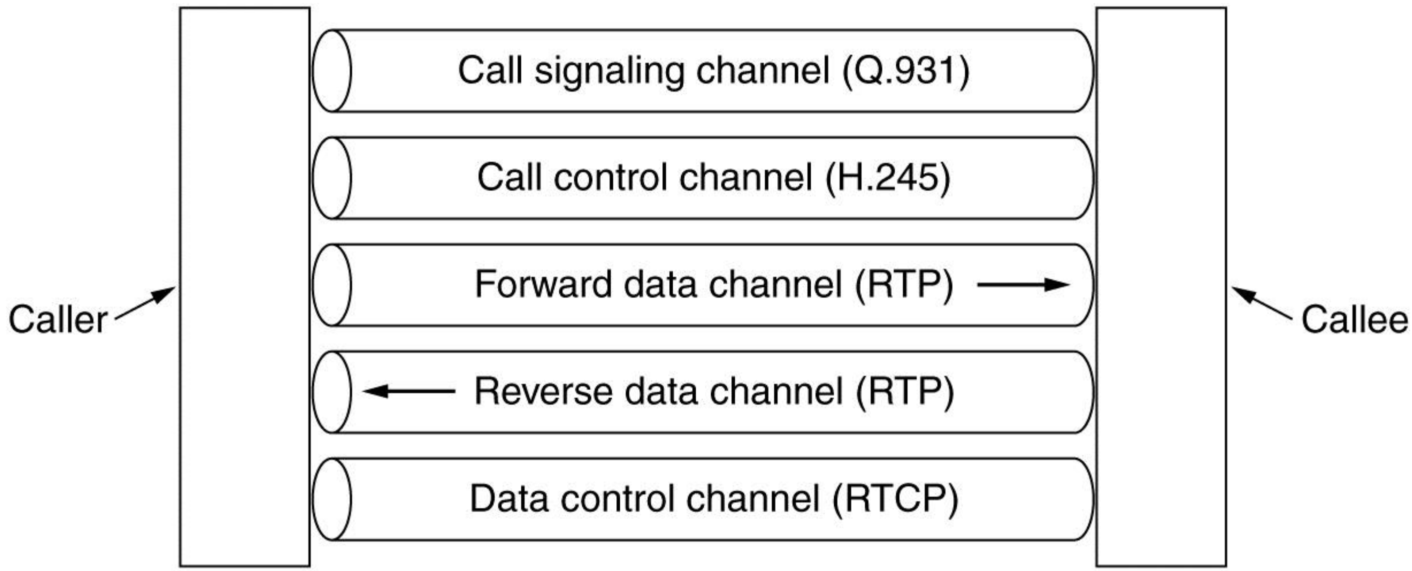
Voice over IP (2)

The H323 protocol stack.



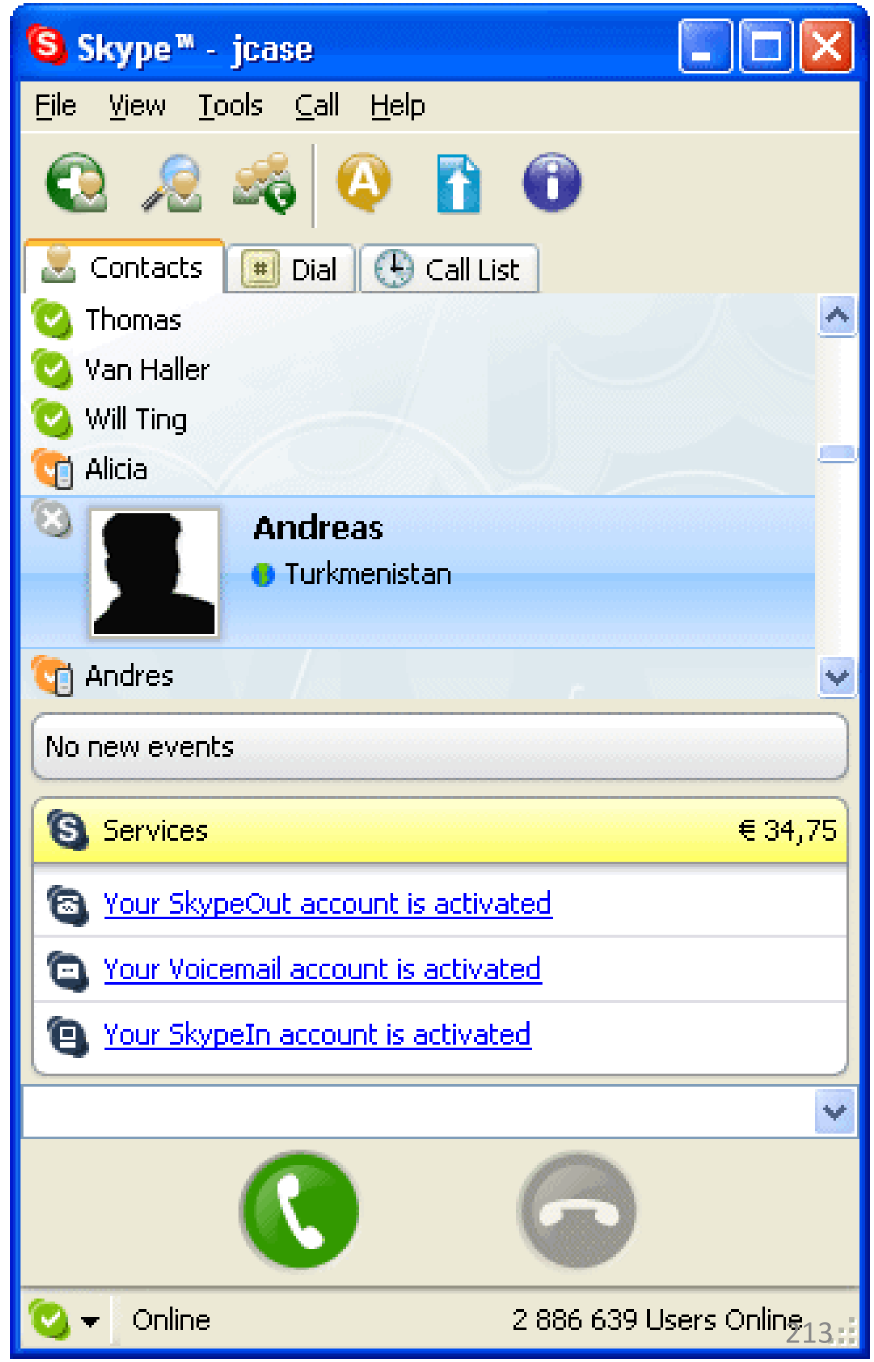
Voice over IP (3)

Logical channels between the caller and callee during a call.



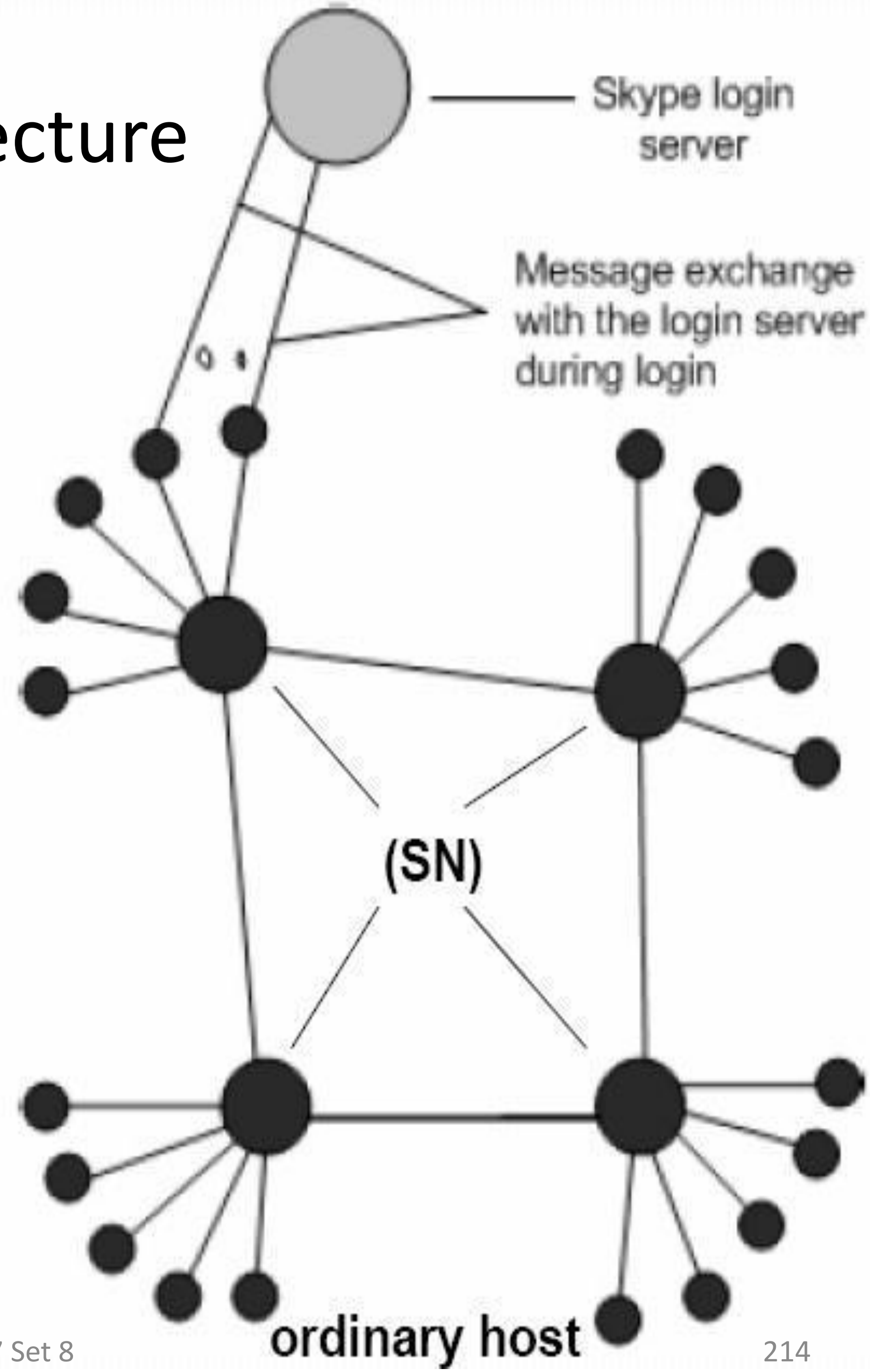
Skype

- Niklas Zennström and Janus Friis in 2003
- Developed by KaZaA
- Instant Messenger (IM) with voice support
- Based on peer-to-peer (P2P) networking technology



Skype Network Architecture

- Login server is the only central server (consisting of multiple machines)
- Both ordinary host and super nodes are Skype clients
- Any node with a public IP address and having sufficient resources can become a super node
- Skype maintains their own super nodes



Challenges of Firewalls and NATs

- Firewalls
  - Often block UDP traffic
  - Usually allow hosts to initiate connections on port 80 (HTTP) and 443 (HTTPS)
- NAT
  - Cannot easily initiate traffic to a host behind a NAT
  - … since there is no unique address for the host
- Skype must deal with these problems
  - Discovery: client exchanges messages with super node
  - Traversal: sending data through an intermediate peer

Data Transfer

- UDP directly between the two hosts
  - Both hosts have public IP address
  - Neither host's network blocks UDP traffic
  - Easy: the hosts can exchange UDP packets directly
- UDP between an intermediate peer
  - One or both hosts with a NAT
  - Neither host's network blocks UDP traffic
  - Solution: direct UDP packets through another node
- TCP between an intermediate peer
  - Hosts behind NAT and UDP-restricted firewall
  - Solution: direct TCP connections through another node

Silence Suppression

- What to transfer during quiet periods?
  - Could save bandwidth by reducing transmissions
  - E.g., send nothing during silence periods
- Skype does not appear to do silence suppression
  - Maintain the UDP bindings in the NAT boxes
  - Provide background noise to play at the receiver
  - Avoid drop in the TCP window size
- Skype sends data when call is “on hold”
  - Send periodic messages as a sort of heartbeat
  - Maintain the UDP bindings in the NAT boxes
  - Detect connectivity problems on the network path

Skype Data Transfer

- Audio compression
  - Voice packets around 67 bytes
  - Up to 140 packets per second
  - Around 5 KB/sec (40 kbps) in each direction
- Encryption
  - Data packets are encrypted in both directions
  - To prevent snooping on the phone call
  - … by someone snooping on the network
  - … or by the intermediate peers forwarding data

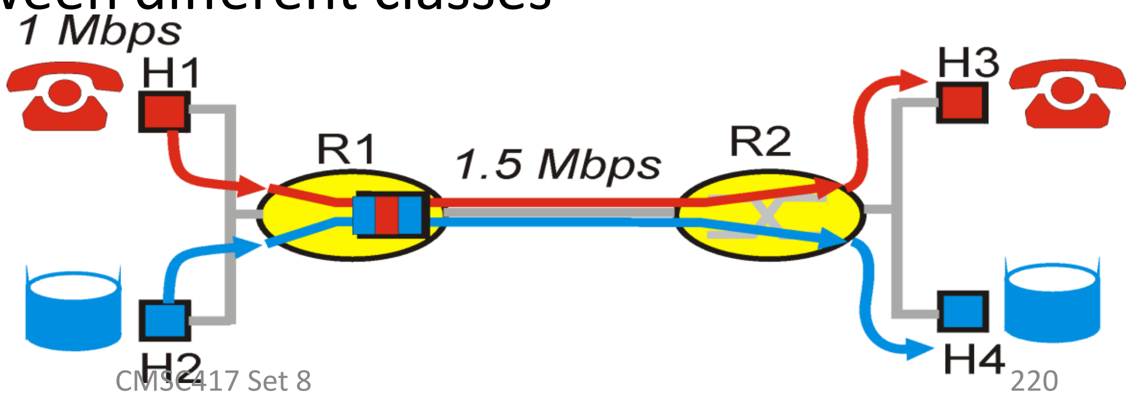
VoIP Quality

- The application can help
  - Good audio compression algorithms
  - Avoiding hops through far-away hosts
  - Forward error correction
  - Adaptation to the available bandwidth
- But, ultimately the network is a major factor
  - Long propagation delay?
  - High congestion?
  - Disruptions during routing changes?

- Leads to an interest in Quality of Service

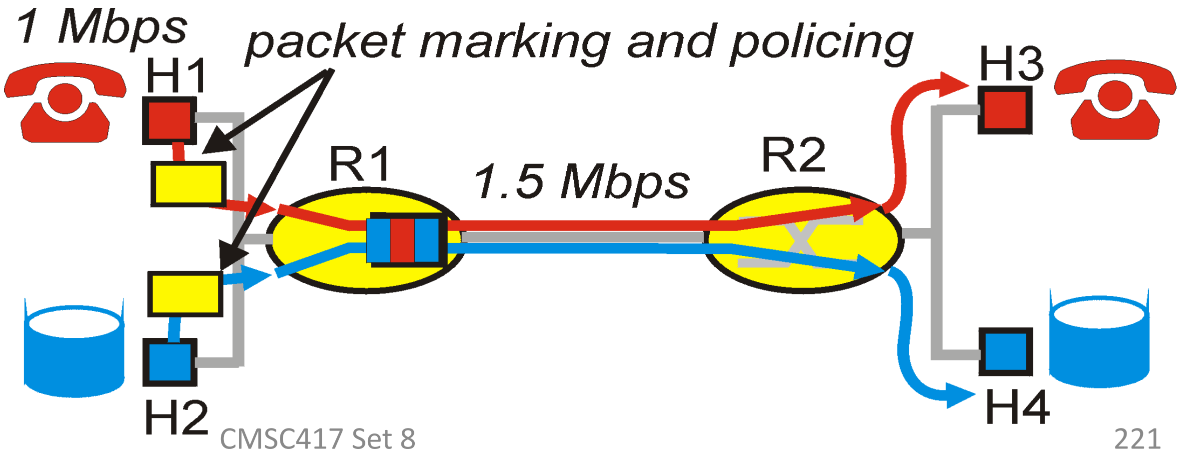
Principles for QoS Guarantees

- Applications compete for bandwidth
  - Consider a 1 Mbps VoIP application and an FTP transfer sharing a single 1.5 Mbps link
  - Bursts of FTP traffic can cause congestion and losses
  - We want to give priority to the audio packets over FTP
- Principle 1: Packet marking
  - Marking of packets is needed for the router
  - To distinguish between different classes



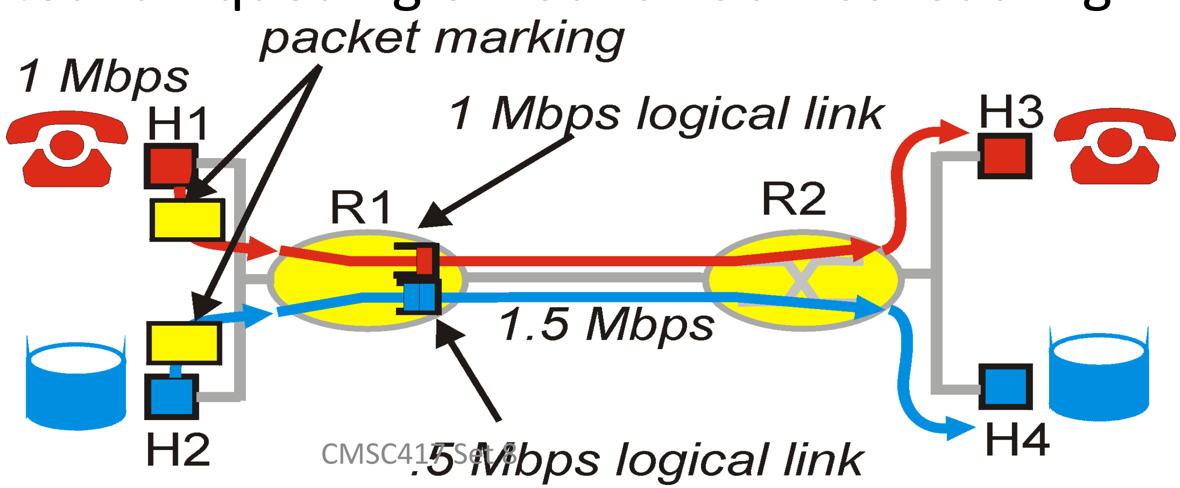
Principles for QoS Guarantees

- Applications misbehave
  - Audio sends packets at a rate higher than 1 Mbps
- Principle 2: Policing
  - Provide protection for one class from other classes
  - Ensure sources adhere to bandwidth restrictions
  - Marking and policing need to be done at the edge



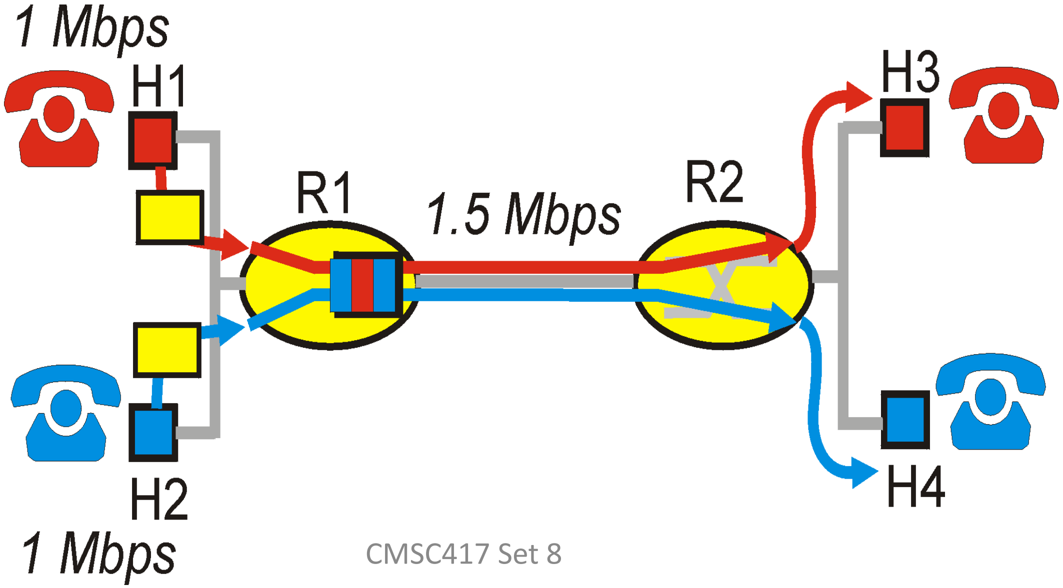
Principles for QoS Guarantees

- Alternative to marking and policing
  - Allocate fixed bandwidth to each application flow
  - But, this can lead to inefficient use of bandwidth
  - … if one of the flows does not use its allocation
- Principle 3: Link scheduling
  - While providing isolation, it is desirable to use resources as efficiently as possible
  - E.g., weighted fair queuing or round-robin scheduling



Principles for QoS Guarantees

- Cannot support traffic beyond link capacity
  - If total traffic exceeds capacity, you are out of luck
  - Degrade the service for all, or deny someone access
- Principle 4: Admission control
  - Application flow declares its needs in advance
  - The network may block call if it cannot satisfy the needs



Quality of Service

- Significant change to Internet architecture
  - Guaranteed service rather than best effort
  - Routers keeping state about the traffic
- A variety of new protocols and mechanisms
  - Reserving resources along a path
  - Identifying paths with sufficient resources
  - Link scheduling and buffer management
  - Packet marking with the Type-of-Service bits
  - Packet classifiers to map packets to ToS classes
  - …
- Seeing some deployment within individual ASes
  - E.g., corporate/campus networks, and within an ISP

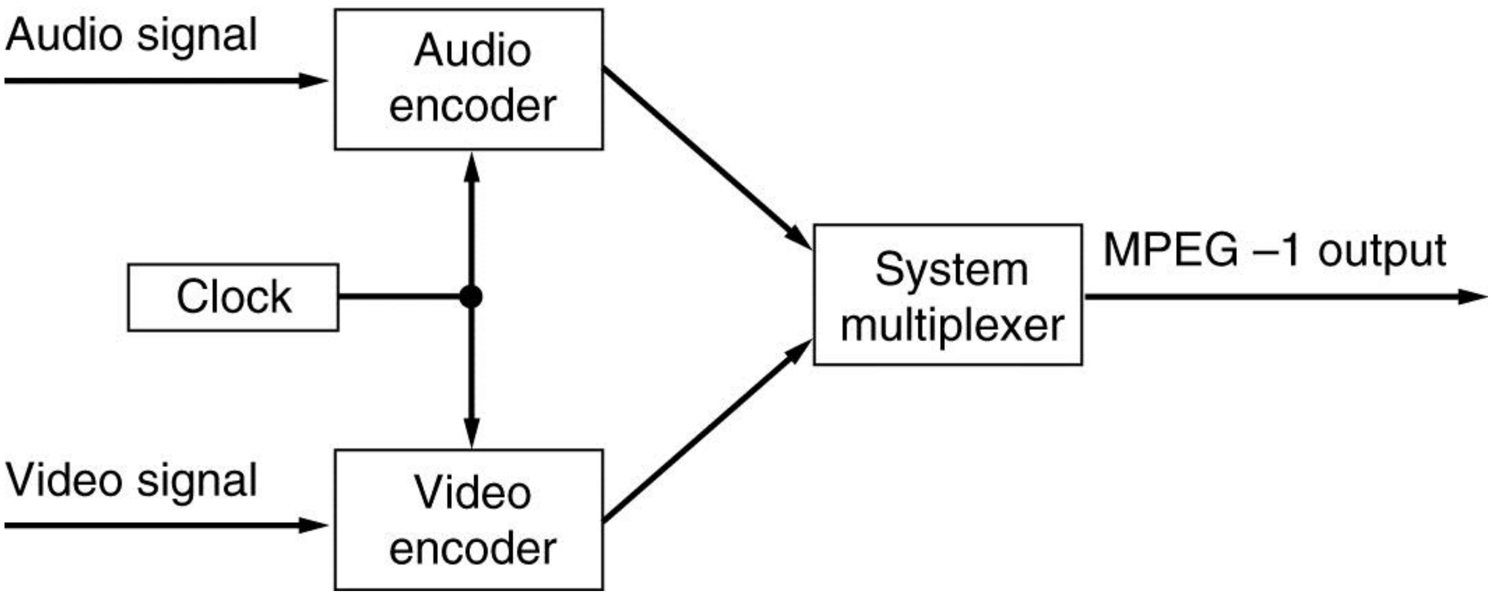
Introduction to MPEG Video Coding

MPEG -1

- Moving Pictures Experts Group, established in 1988
- Approved by ISO/IEC MPEG Group in November'91
  - Coding moving pictures & associated audio up to 1.5Mbit/s
  - Up to 1.2Mbit/s for video & 256kbps for audio
  - Supports only non-interlaced video

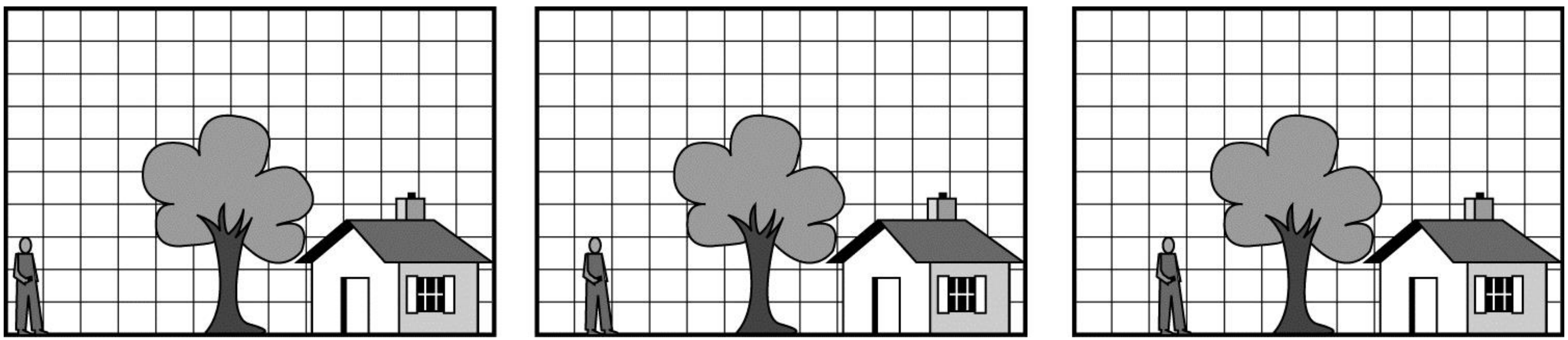
The MPEG Standard

Synchronization of the audio and video streams in MPEG-1.

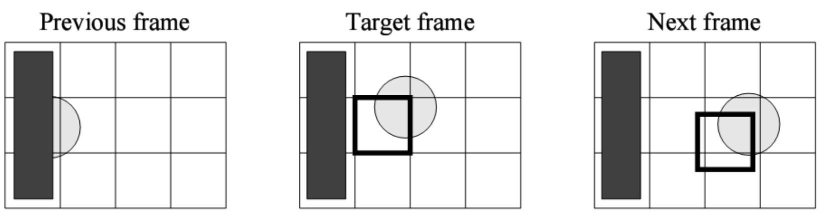


The MPEG Standard (2)

Three consecutive frames.



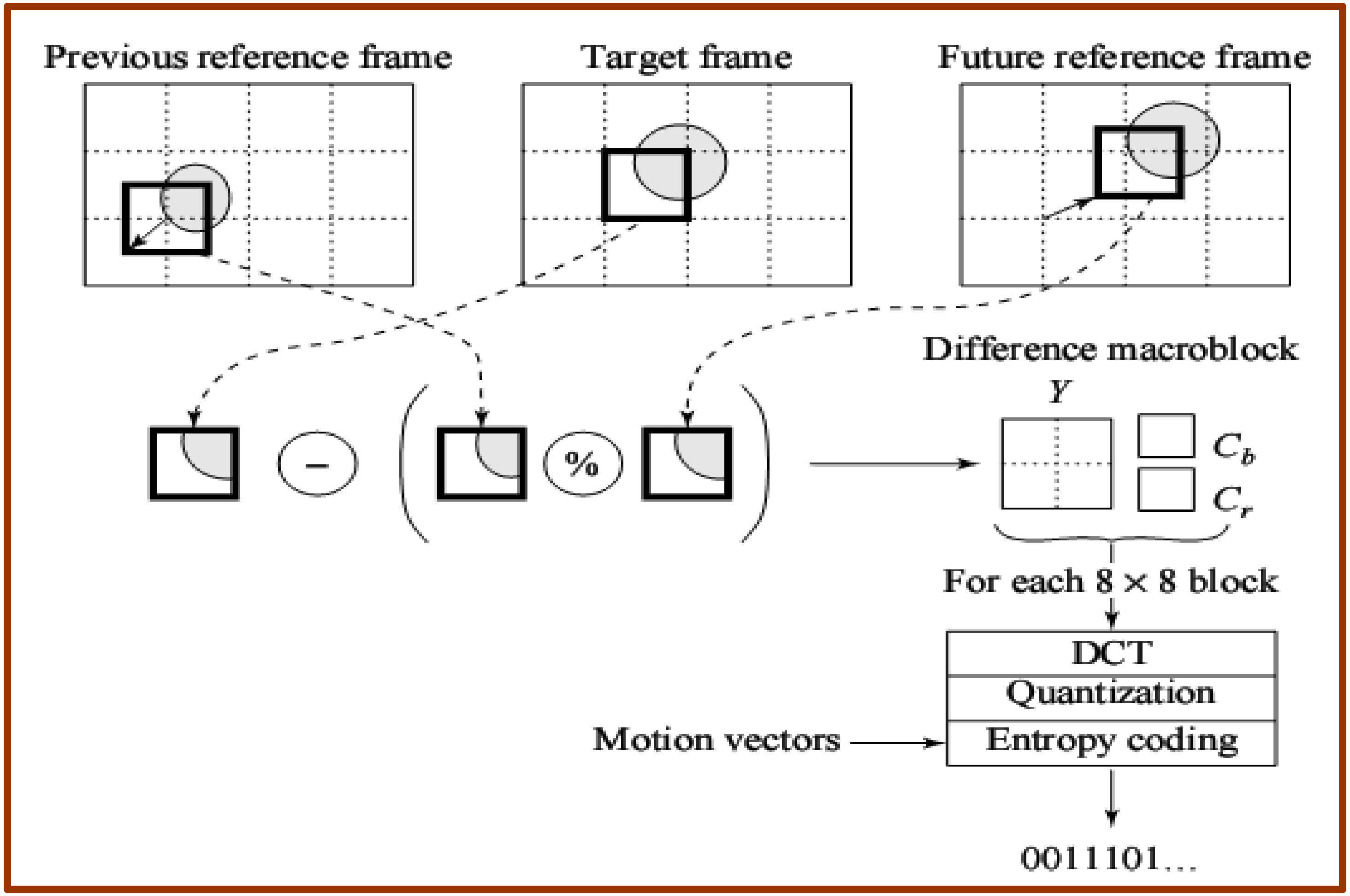
MPEG-1 Motion Compensation



Need for bidirectional Motion compensation

- Introduces third frame (in addition to I-Frame, P-Frame) *B-Frame*
- B-Frame uses backward prediction & forward prediction
  - Each MB in the B-Frame will have 2 motion vectors

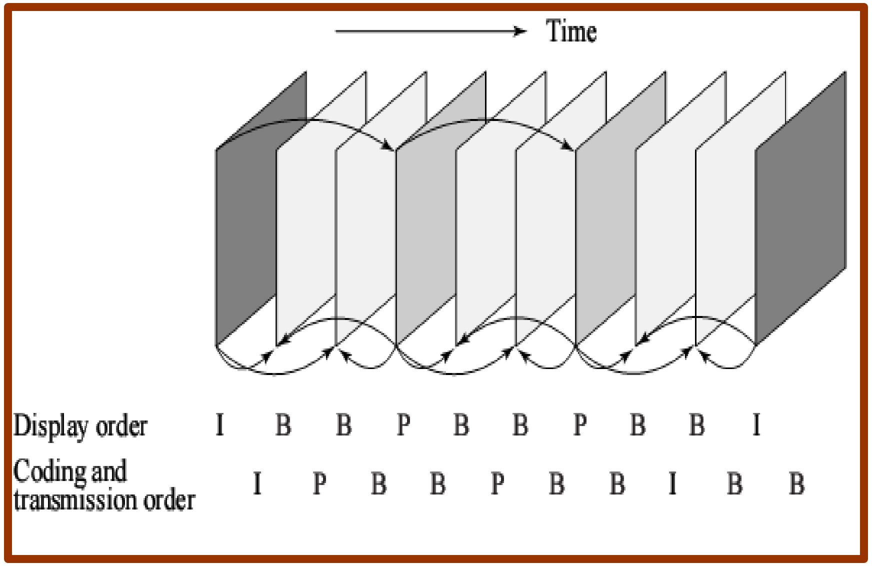
MPEG-1 Motion Compensation [2]



B-Frame Coding based on bidirectional motion compensation

MPEG-1 Motion Compensation [3]

- Notation
  - M ← interval between P-Frame and preceding I or P Frame
  - N ← interval between two consecutive I-Frames
  - In Fig M=3 & N=9



MPEG Frame Sequence

MPEG-1 Major differences from H.261

- Source Formats
  - Supports any formats that meet the constrained parameter set

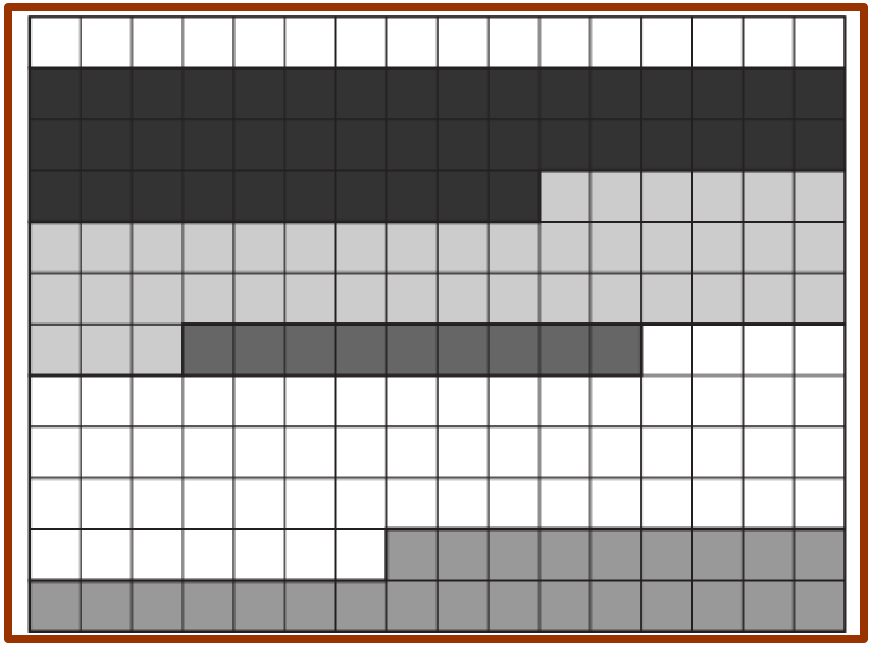
Parameter	Value
Horizontal size of picture	≤ 768
Vertical size of picture	≤ 576
No. of MBs / picture	≤ 396
No. of MBs / second	$\leq 9,900$
Frame rate	≤ 30 fps
Bit-rate	$\leq 1,856$ kbps

Constrained Parameter Set

MPEG-1 Major differences from H.261

[2]

- Slices
  - MPEG-1 picture is divided into slices
  - Variable number of macro-blocks
  - Each slice is coded separately
  - Unique start code



Slices in MPEG-1 Picture

MPEG-1 Major differences from H.261

- Quantization
  - Different quantization tables for *intra* & *inter* coding
  - For intra-coding

$$QDCT[i,j] = round\left(\frac{8 \times DCT[i,j]}{step\_size[i,j]}\right) = round\left(\frac{8 \times DCT[i,j]}{Q_1[i,j] * scale}\right)$$

  - For inter-coding

$$QDCT[i,j] = \left\lfloor \frac{8 \times DCT[i,j]}{step\_size[i,j]} \right\rfloor = \left\lfloor \frac{8 \times DCT[i,j]}{Q_2[i,j] * scale} \right\rfloor$$

[3]

8	16	19	22	26	27	29	34
16	16	22	24	27	29	34	37
19	22	26	27	29	34	34	38
22	22	26	27	29	34	37	40
22	26	27	29	32	35	40	48
26	27	29	32	35	40	48	58
26	27	29	34	38	46	56	69
27	29	35	38	46	56	69	83

Intra - Coding

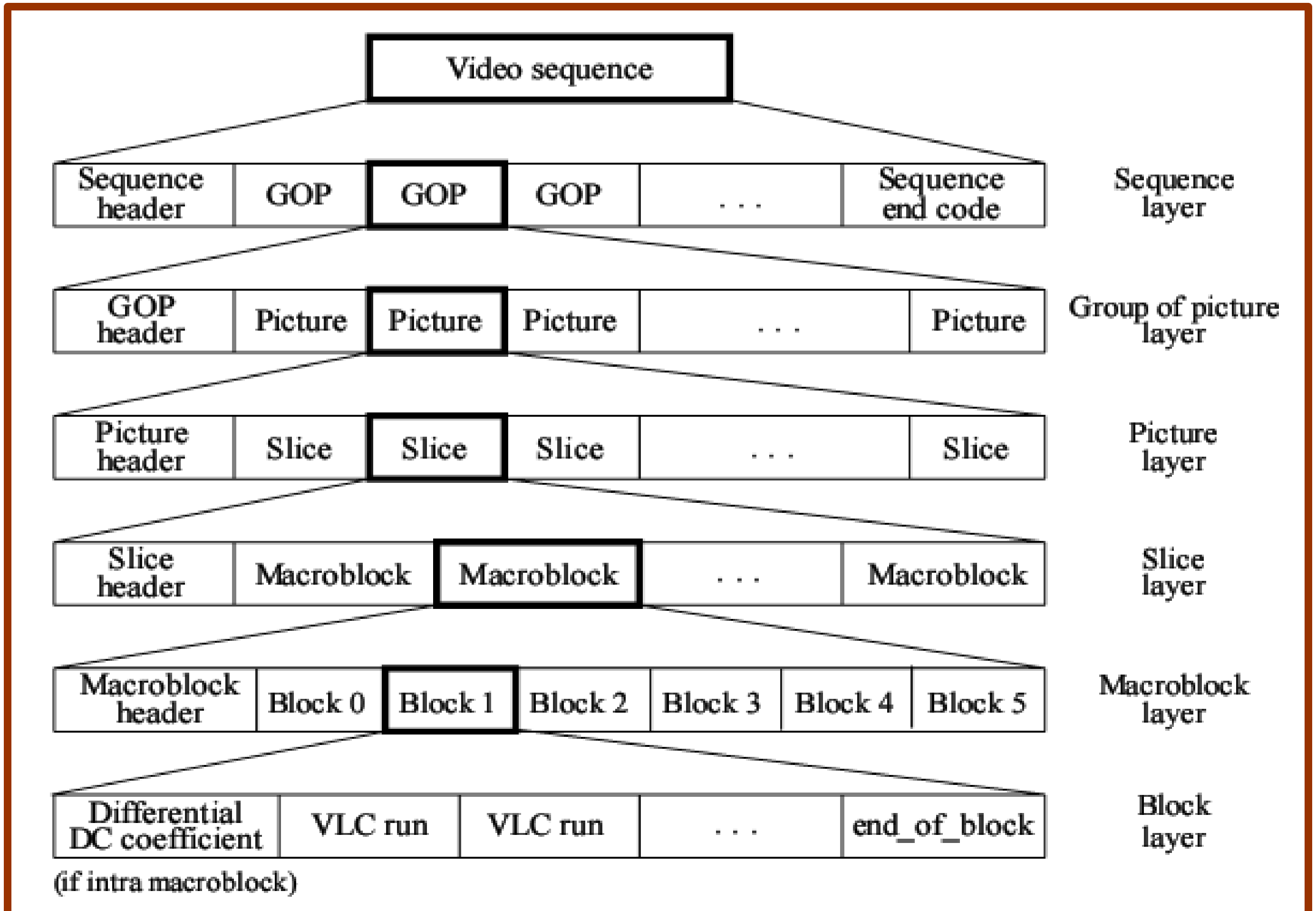
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16
16	16	16	16	16	16	16	16

Inter - Coding

MPEG-1 Video Stream

- Sequence layer
  - Header contains the info about the picture *horizontal size, vertical size, pixel aspect ratio, frame rate, bitrate, buffer size* etc
- GOP layer
  - One picture must be I-picture
  - Header contains time code to indicate hour-min-sec-frame
- Picture layer
  - I, P, B, D pictures
- Slice layer
- MB layer
  - 6 blocks of 8x8
- Block layer
  - Dc co-eff is sent first followed by AC co-effs

MPEG-1 Video Stream [2]



MPEG-2 Introduction

- Focused at higher bitrates, more than 4Mbps
- Development of standard started in 1990
- Approved by ISO/IEC Moving Picture Experts Group in 1994
- Standard adopted for DVDs
- Defines 7 profiles aimed at different applications
  - Simple , Main, SNR Scalability, Spatially Scalable, High, 4:2:2 and Multiview
  - Each profile defines 4 levels

MPEG-2 Introduction [2]

Level	Simple Profile	Main Profile	SNR Scalable Profile	Spatially Scalable Profile	High Profile	4:2:2 Profile	Multiview Profile
High		*			*		
High 1440		*		*	*		
Main	*	*	*		*	*	*
Low		*	*				

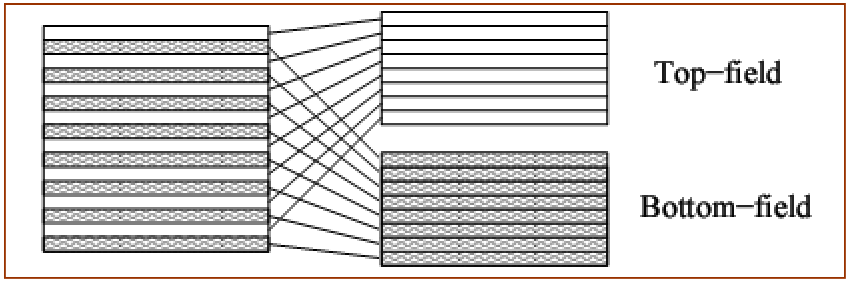
Profiles & Levels in MPEG - 2

Level	Max Resolution	Max fps	Max Pixels/sec	Max coded Data Rate (Mbps)	Application
High	$1,920 \times 1,152$	60	62.7×10^6	80	film production
High 1440	$1,440 \times 1,152$	60	47.0×10^6	60	consumer HDTV
Main	720×576	30	10.4×10^6	15	studio TV
Low	352×288	30	3.0×10^6	4	consumer tape equiv.

Four Levels in the main profile

MPEG-2 Support for Interlaced Video

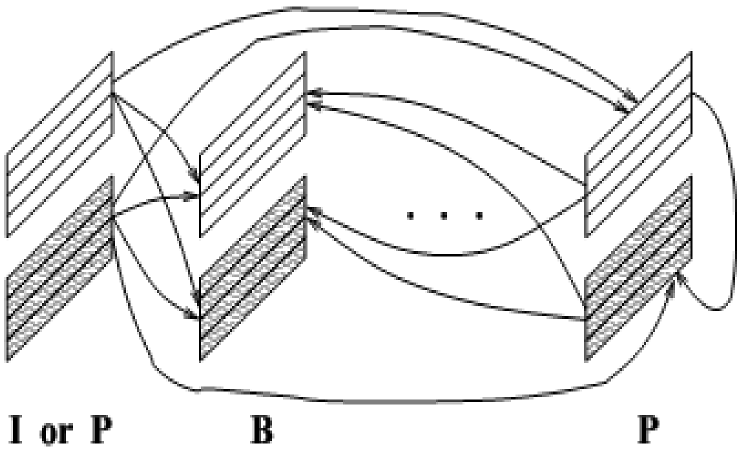
- MPEG-2 is adopted by broadcast TV's, need to support interlaced mode
- Each frame consists of 2 fields
  - Top-field
  - Bottom-field
- All scanlines are interleaved to form the frame picture, then macroblock is formed & coding proceeds
- Each field is treated as separate picture, *field-picture*



Frame Picture Vs Field Picture

MPEG-2 Prediction Modes

1. Frame prediction for frame pictures
2. Field prediction for field pictures
   - Uses macroblock 16 x 16 from field pictures
3. Field prediction for frame pictures
4. 16 x 8 MC for field-pictures
5. Dual-prime for P-pictures



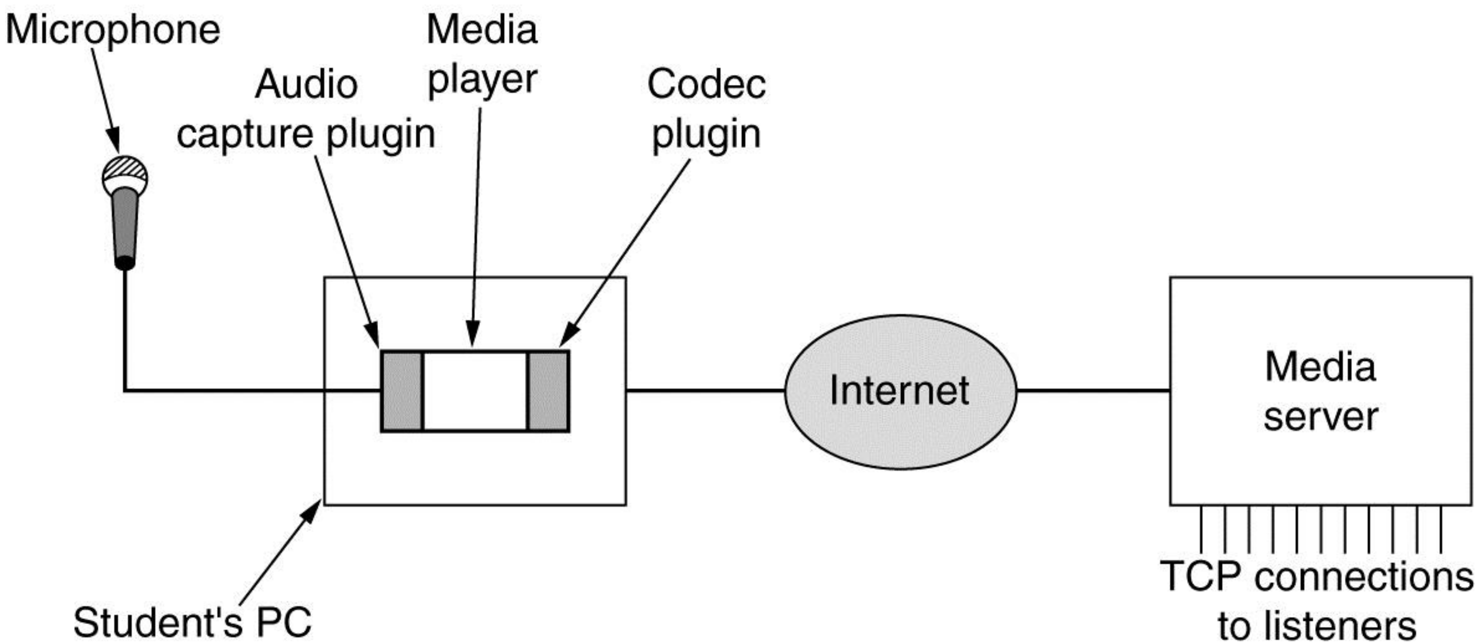
Field Prediction for Field pictures

MPEG-2 Scalabilities

- MPEG-2 is designed for various applications , digital TV, HDTV& the video will be transmitted over networks with very different characteristics
- Single coded MPEG-2 bitstream should be scalable for various bitrates
- Coded as base layer and enhancement layers
- Following scalabilities are supported
  1. SNR Scalability
  2. Spatial Scalability
  3. Temporal Scalability
  4. Hybrid scalability
  5. Data Partitioning

Internet Radio

A student radio station.



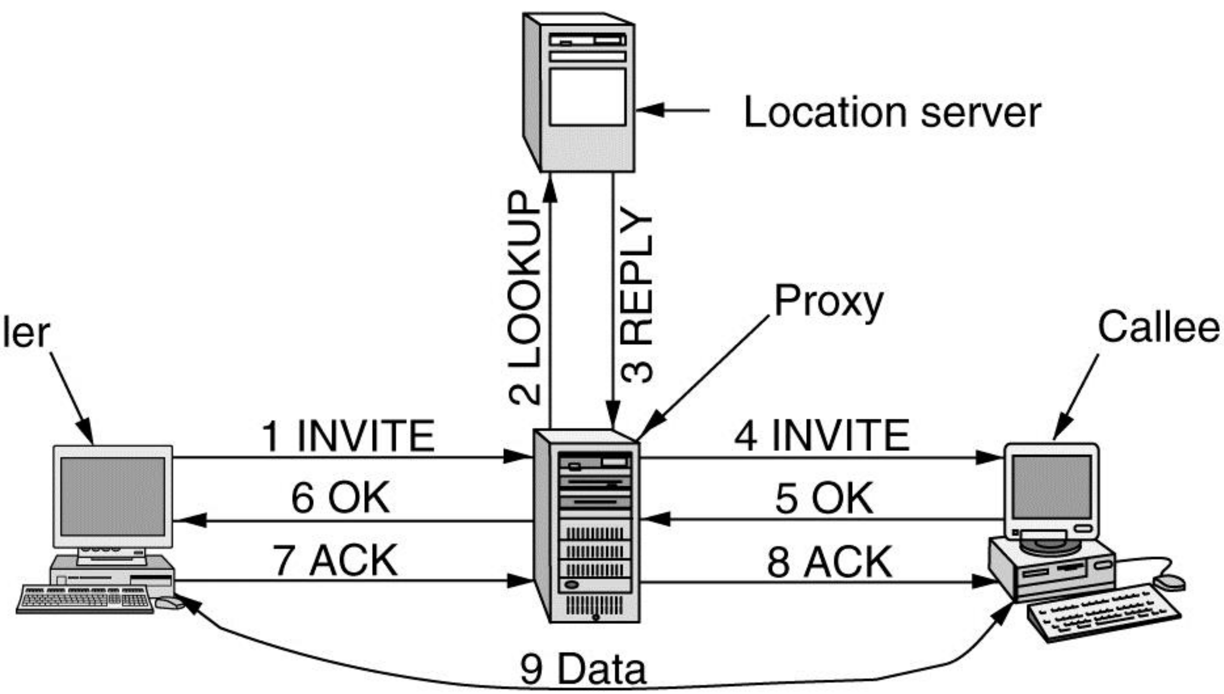
SIP – The Session Initiation Protocol

The SIP methods defined in the core specification.

Method	Description
INVITE	Request initiation of a session
ACK	Confirm that a session has been initiated
BYE	Request termination of a session
OPTIONS	Query a host about its capabilities
CANCEL	Cancel a pending request
REGISTER	Inform a redirection server about the user's current location

SIP (2)

Use a proxy and redirection servers with SIP.

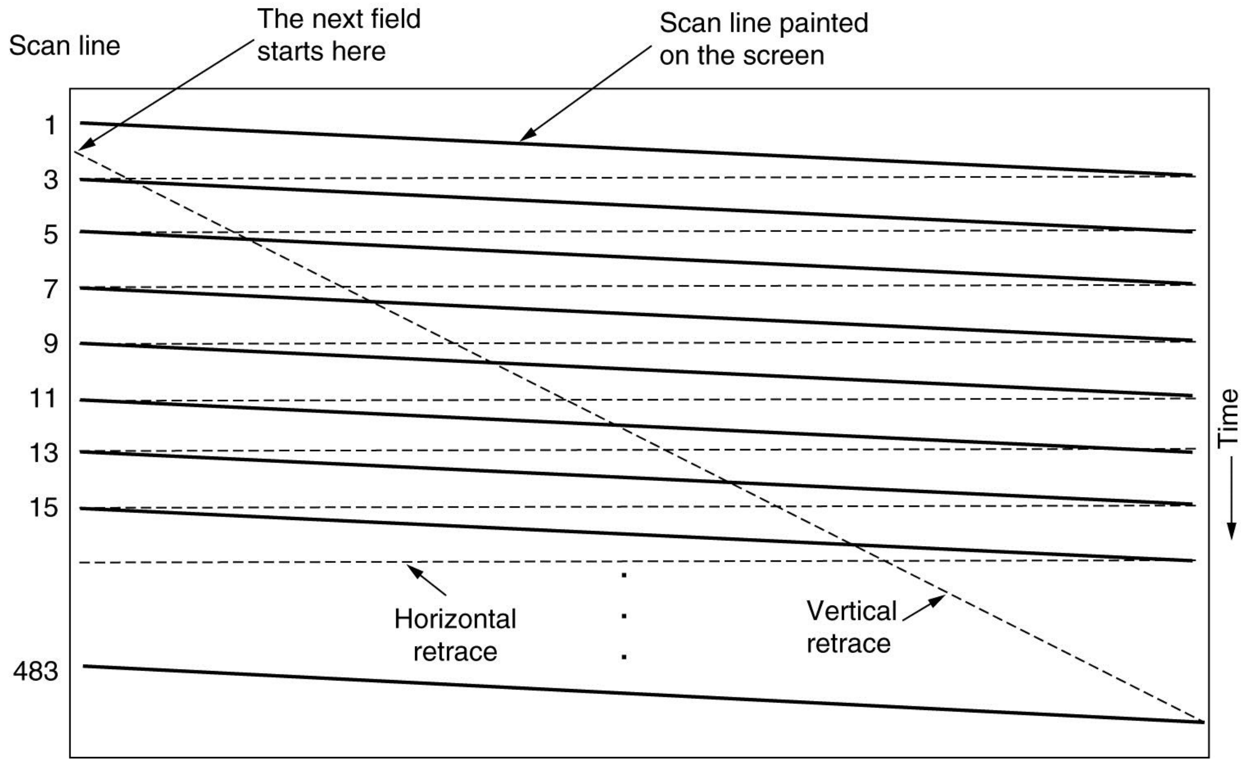


Comparison of H.323 and SIP

Item	H.323	SIP
Designed by	ITU	IETF
Compatibility with PSTN	Yes	Largely
Compatibility with Internet	No	Yes
Architecture	Monolithic	Modular
Completeness	Full protocol stack	SIP just handles setup
Parameter negotiation	Yes	Yes
Call signaling	Q.931 over TCP	SIP over TCP or UDP
Message format	Binary	ASCII
Media transport	RTP/RTCP	RTP/RTCP
Multiparty calls	Yes	Yes
Multimedia conferences	Yes	No
Addressing	Host or telephone number	URL
Call termination	Explicit or TCP release	Explicit or timeout
Instant messaging	No	Yes
Encryption	Yes	Yes
Size of standards	1400 pages	250 pages
Implementation	Large and complex	Moderate
Status	Widely deployed	Up and coming

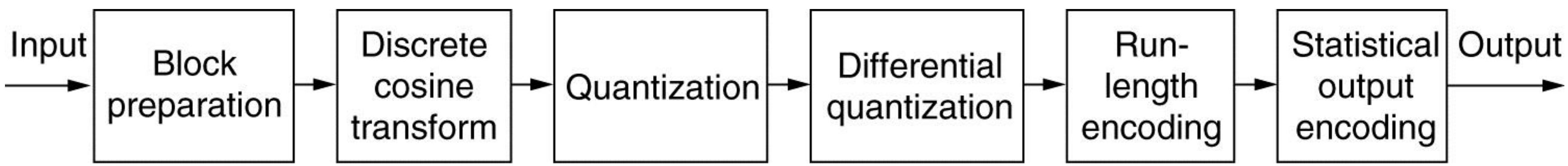
Video Analog Systems

The scanning pattern used for NTSC video and television.

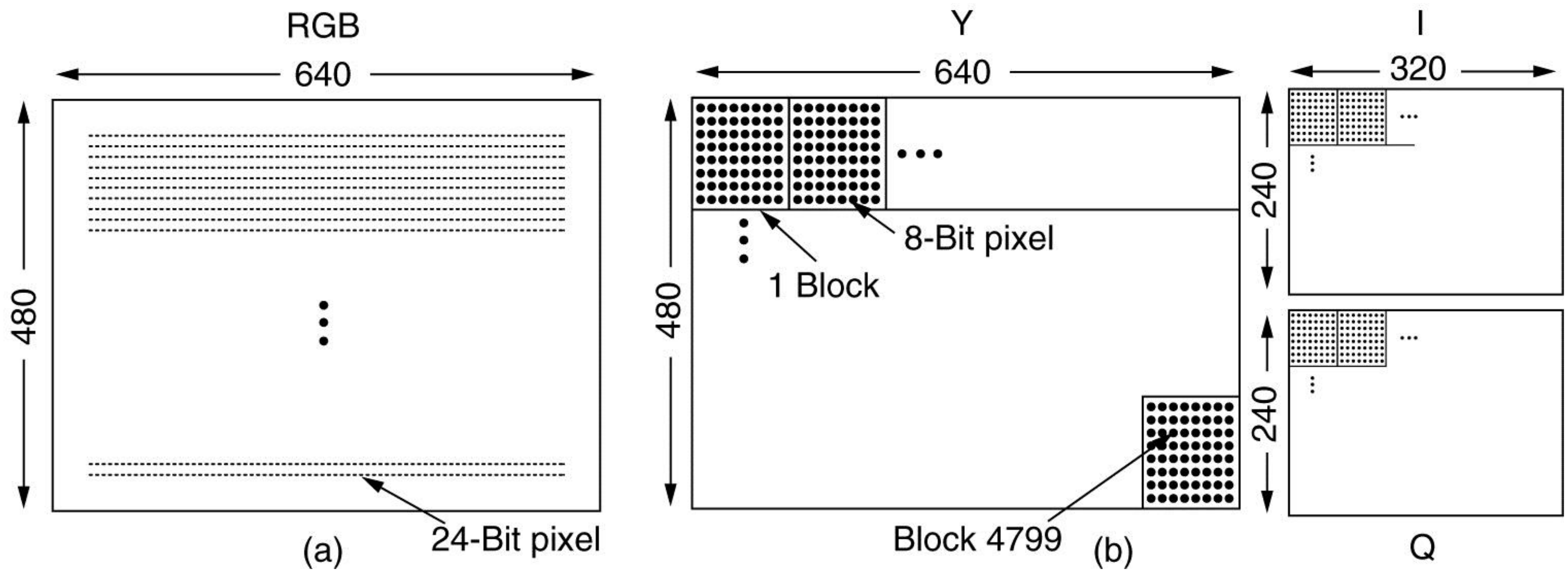


The JPEG Standard

The operation of JPEG in lossy sequential mode.



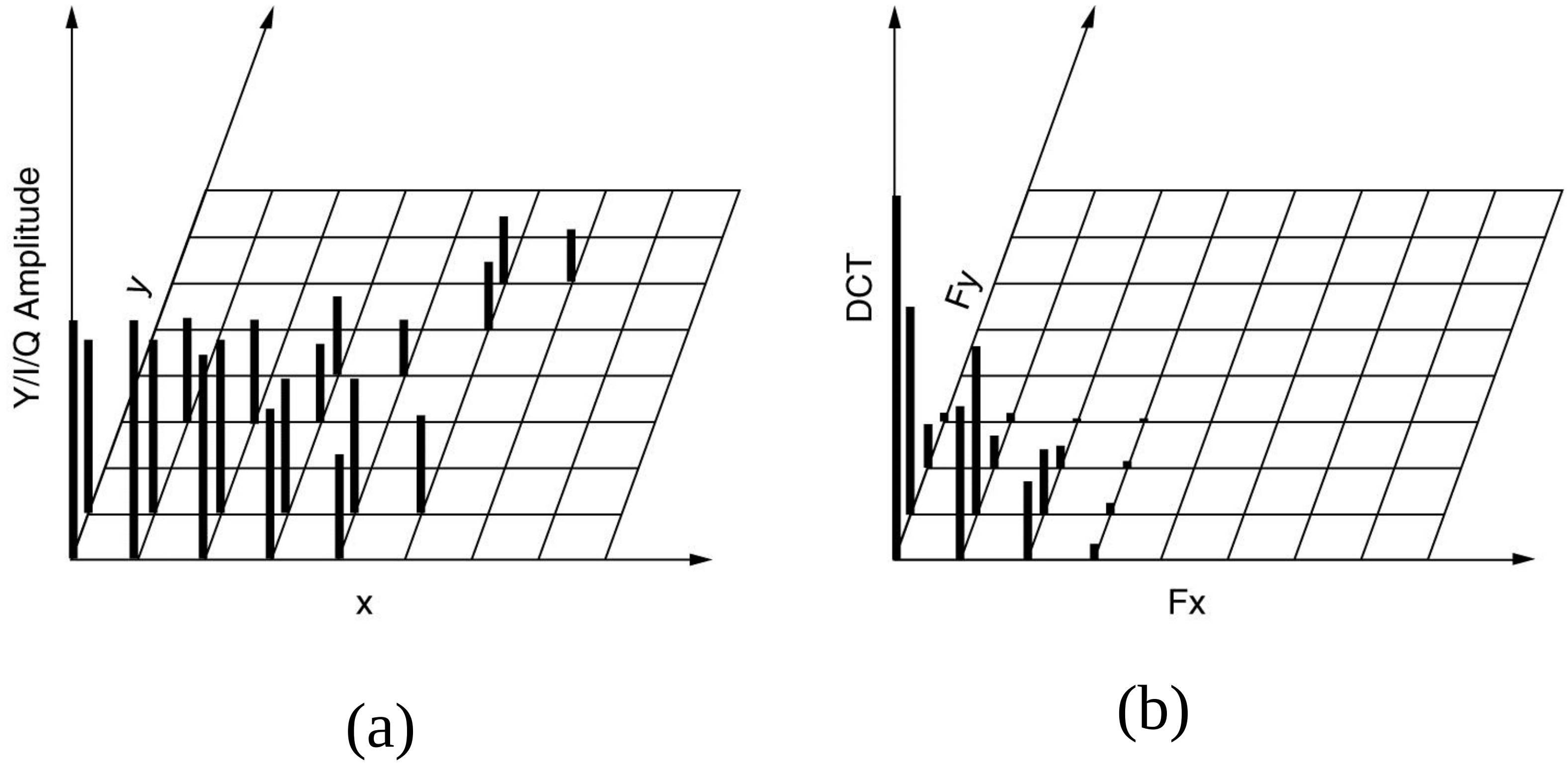
The JPEG Standard (2)



(a) RGB input data.

(b) After block preparation.

The JPEG Standard (3)



(a) One block of the *Y* matrix.
(b) The DTC coefficients.

The JPEG Standard (4)

Computation of the quantized DTC coefficients.

DCT Coefficients

150	80	40	14	4	2	1	0
92	75	36	10	6	1	0	0
52	38	26	8	7	4	0	0
12	8	6	4	2	1	0	0
4	3	2	0	0	0	0	0
2	2	1	1	0	0	0	0
1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Quantization table

1	1	2	4	8	16	32	64
1	1	2	4	8	16	32	64
2	2	2	4	8	16	32	64
4	4	4	4	8	16	32	64
8	8	8	8	8	16	32	64
16	16	16	16	16	16	32	64
32	32	32	32	32	32	32	64
64	64	64	64	64	64	64	64

Quantized coefficients

150	80	20	4	1	0	0	0
92	75	18	3	1	0	0	0
26	19	13	2	1	0	0	0
3	2	2	1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

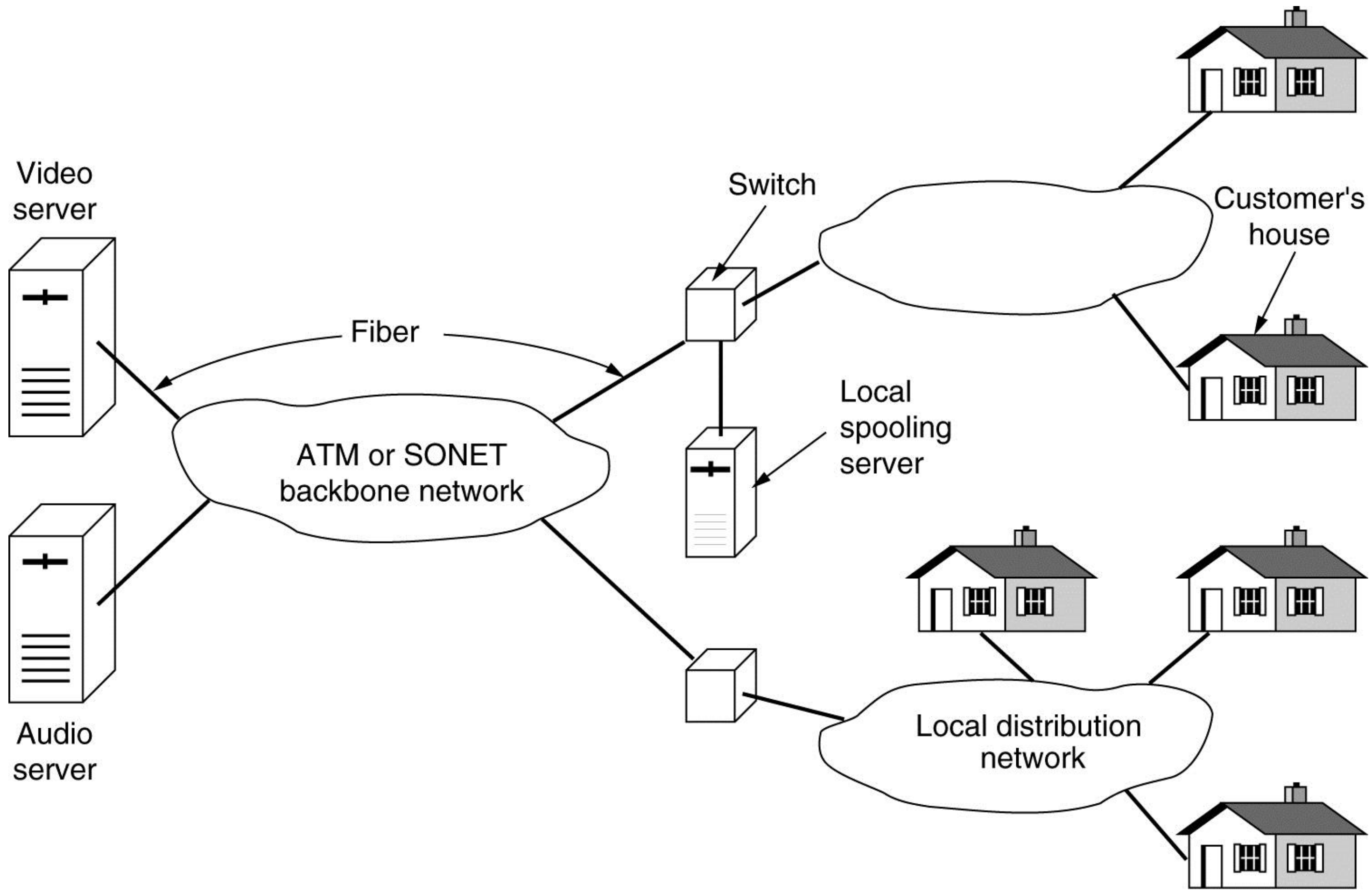
The JPEG Standard (5)

The order in which the quantized values are transmitted.

150	80	20	4	1	0	0	0
92	75	18	3	1	0	0	0
26	19	13	2	1	0	0	0
3	2	2	1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

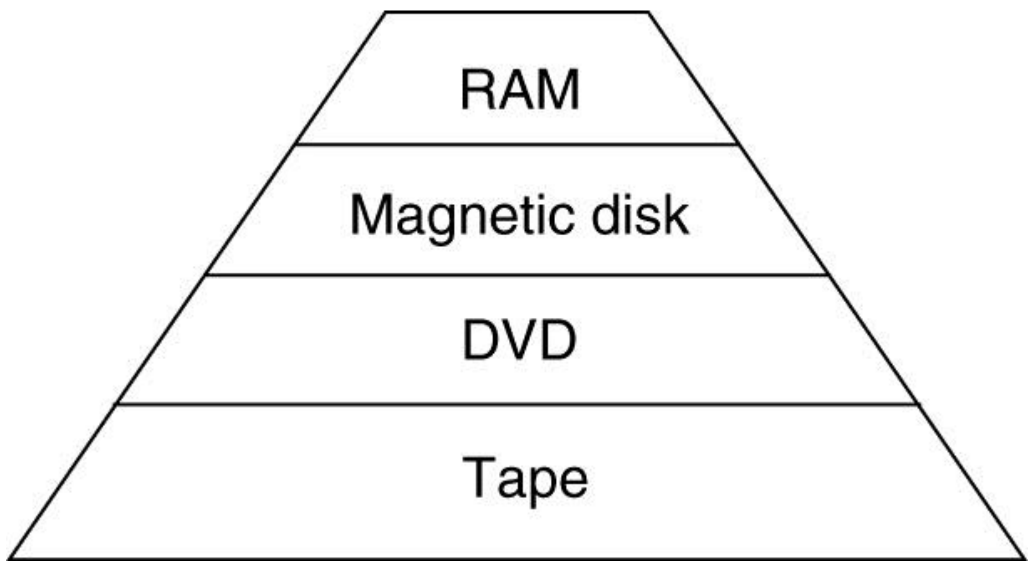
Video on Demand

Overview of a video-on-demand system.



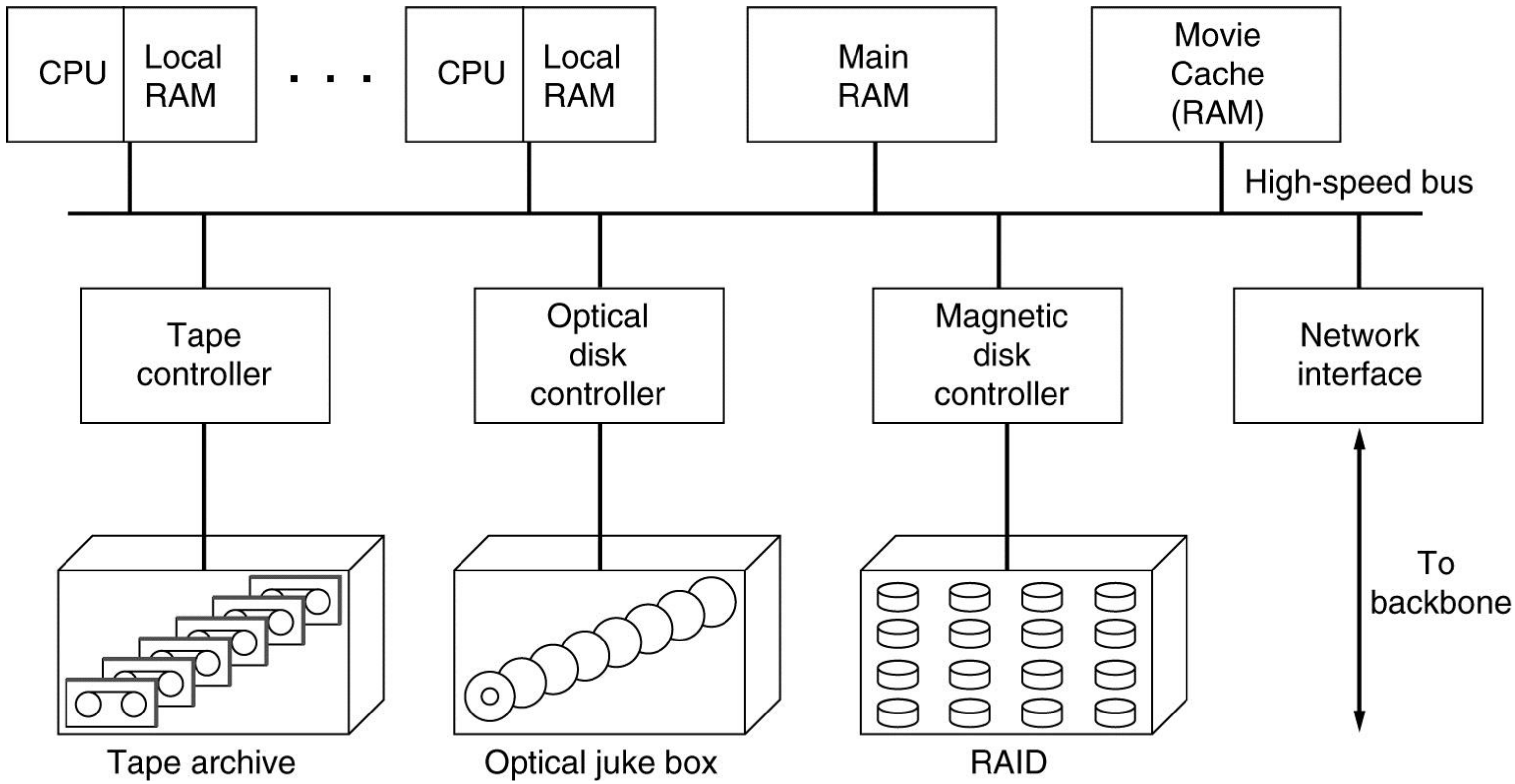
Video Servers

A video server storage hierarchy.



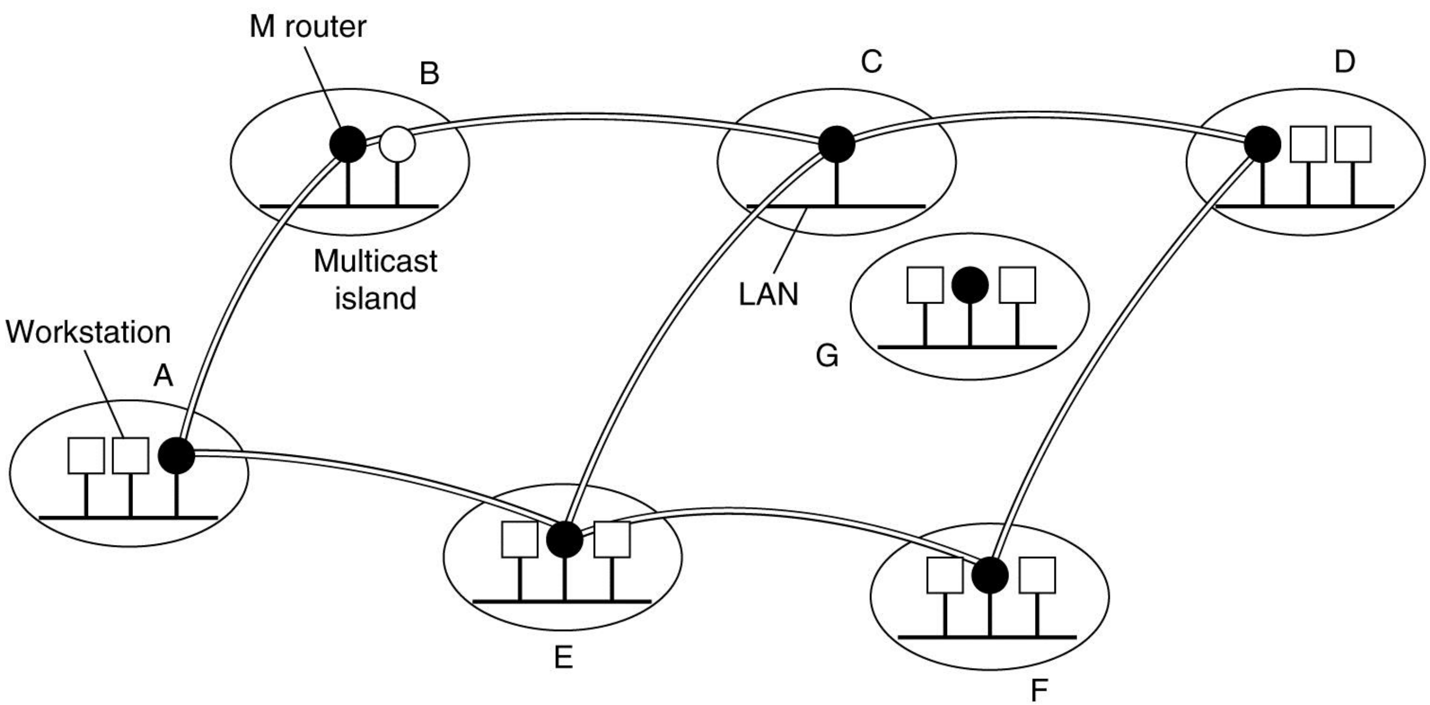
Video Servers (2)

The hardware architecture of a typical video server.



The MBone – The Multicast Backbone

MBone consists of multicast islands connected by tunnels.



Streaming Audio and Video

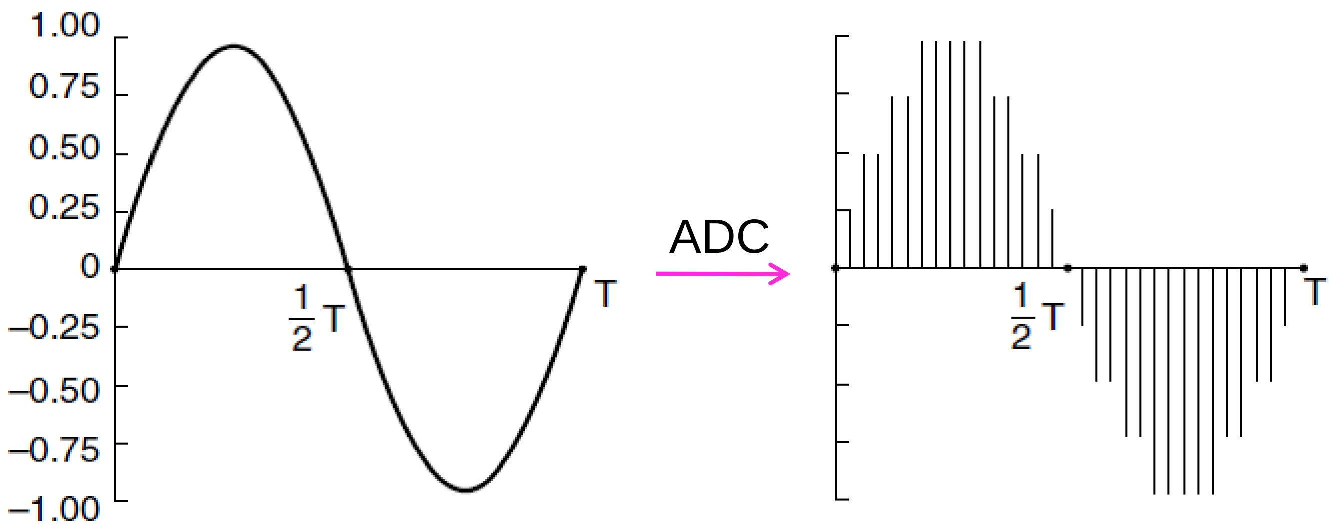
Audio and video have become key types of traffic, e.g., voice over IP, and video streaming.

- Digital audio »
- Digital video »
- Streaming stored media »
- Streaming live media »
- Real-time conferencing »

Digital Audio (1)

ADC (Analog-to-Digital Converter) produces digital audio from a microphone

- Telephone: 8000 8-bit samples/second (64 Kbps); computer audio is usually better quality



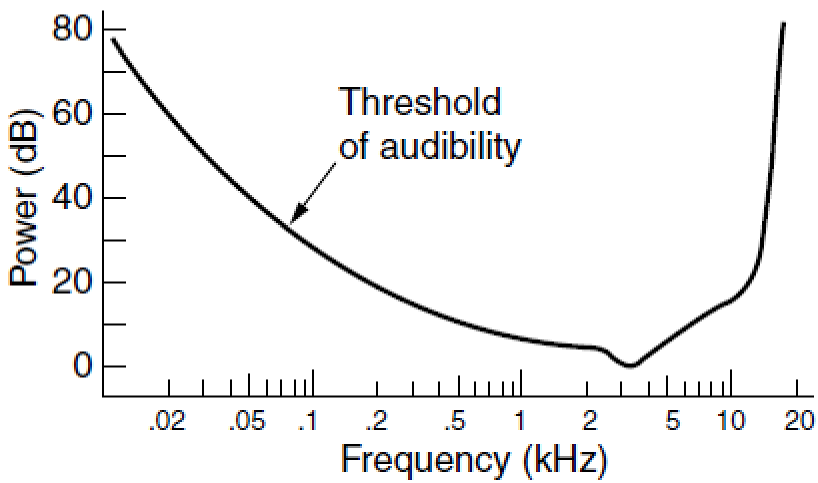
Continuous audio (sine wave)

Digital audio (sampled, 4-bit quantized)

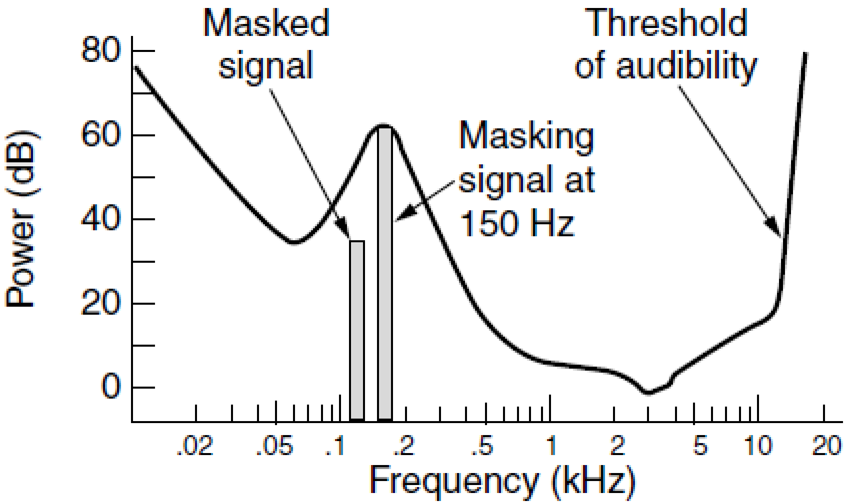
Digital Audio (2)

Digital audio is typically compressed before it is sent

- Lossy encoders (like AAC) exploit human



Sensitivity of the ear varies with frequency



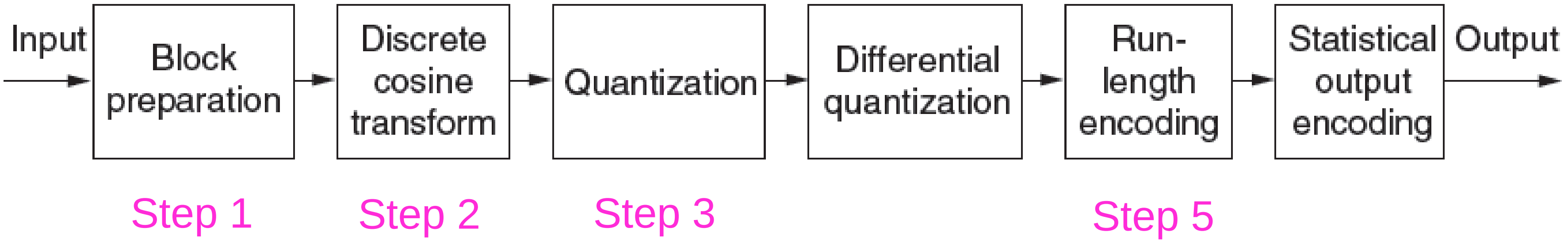
A loud tone can mask nearby tones

Digital Video (1)

- Video is digitized as pixels (sampled, quantized)
  - TV quality: 640x480 pixels, 24-bit color, 30 times/sec
- Video is sent compressed due to its large bandwidth
  - 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

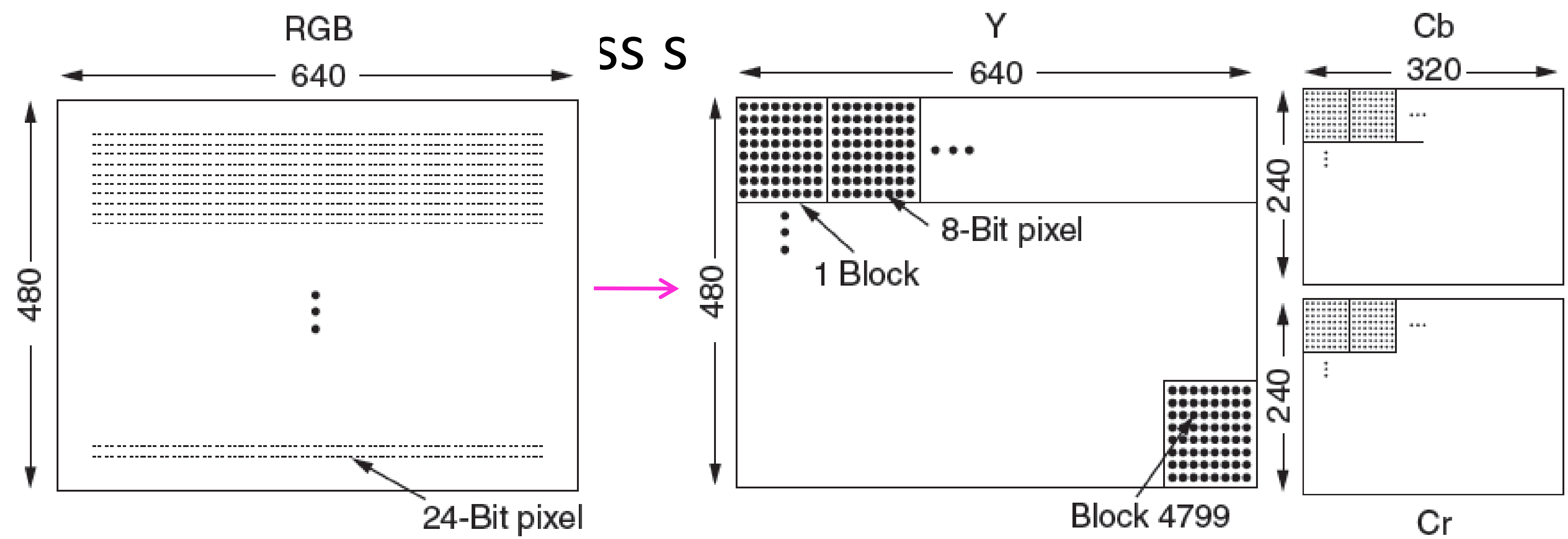
Digital Video (2)

JPEG lossy compression sequence for one image:



Digital Video (3)

Step 1: Pixels are mapped to luminance/chrominance (YCbCr) color space and chrominance is sub-sampled



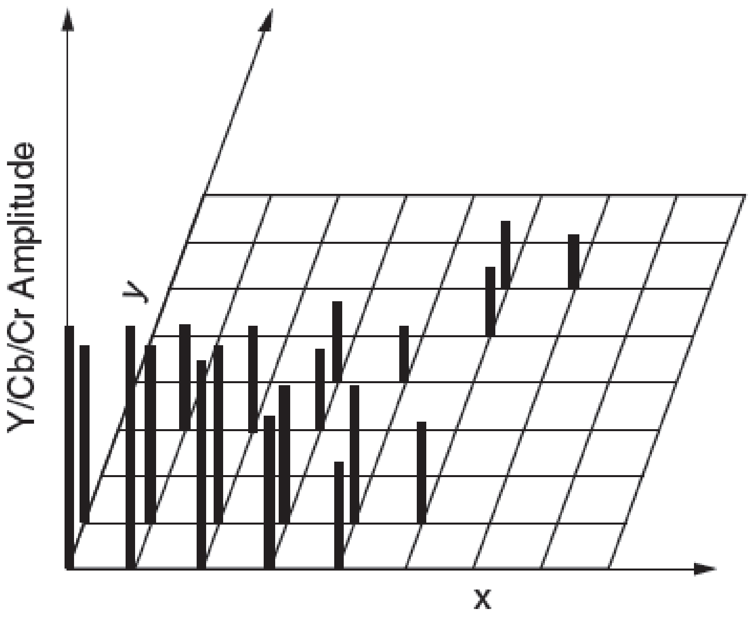
Input 24-bit RGB pixels

8-bit luminance pixels

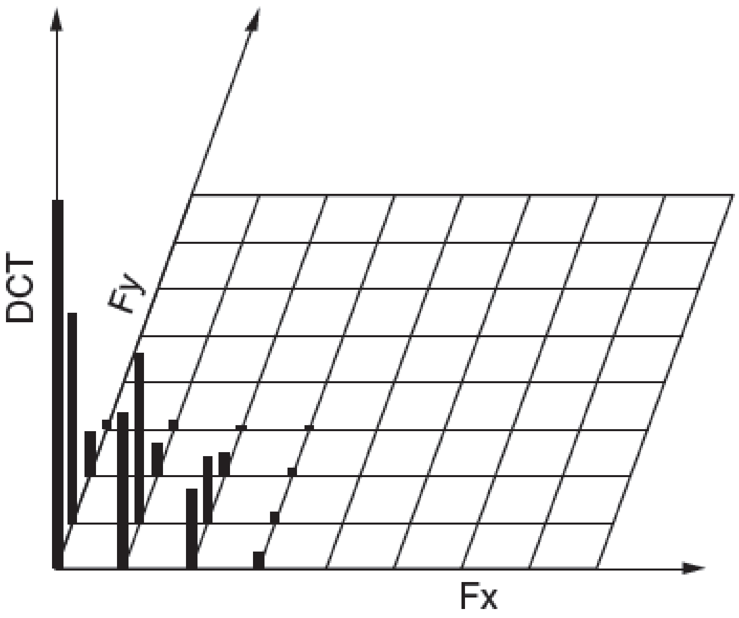
8-bit chrominances for every 4 pixels

Digital Video (4)

Step 2: Each component block is transformed to spatial frequencies with DCT (Discrete Cosine Transformation)



One component block



Transformed block

Digital Video (4)

DCT coefficients

150	80	40	14	4	2	1	0
92	75	36	10	6	1	0	0
52	38	26	8	7	4	0	0
12	8	6	4	2	1	0	0
4	3	2	0	0	0	0	0
2	2	1	1	0	0	0	0
1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Quantization table

1	1	2	4	8	16	32	64
1	1	2	4	8	16	32	64
2	2	2	4	8	16	32	64
4	4	4	4	8	16	32	64
8	8	8	8	8	16	32	64
16	16	16	16	16	16	32	64
32	32	32	32	32	32	32	64
64	64	64	64	64	64	64	64

Quantized coefficients

150	80	20	4	1	0	0	0
92	75	18	3	1	0	0	0
26	19	13	2	1	0	0	0
3	2	2	1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Computation of the quantized DCT coefficients.

Digital Video (5)

150	80	20	4	1	0	0	0
92	75	18	3	1	0	0	0
26	19	13	2	1	0	0	0
3	2	2	1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

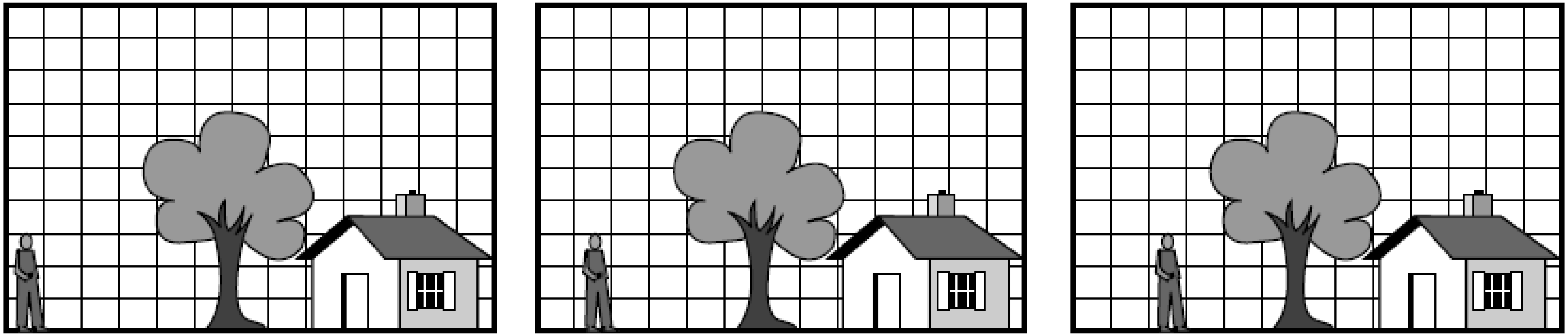
The order in which the quantized values are transmitted.

Digital Video (6)

MPEG output consists of three kinds of frames:

- I- (Intracoded) :
  Self-contained compressed still pictures.
- P- (Predictive) : Block-by-block difference with previous frames.
- B- (Bidirectional) : block-by-block differences between previous and future frames.

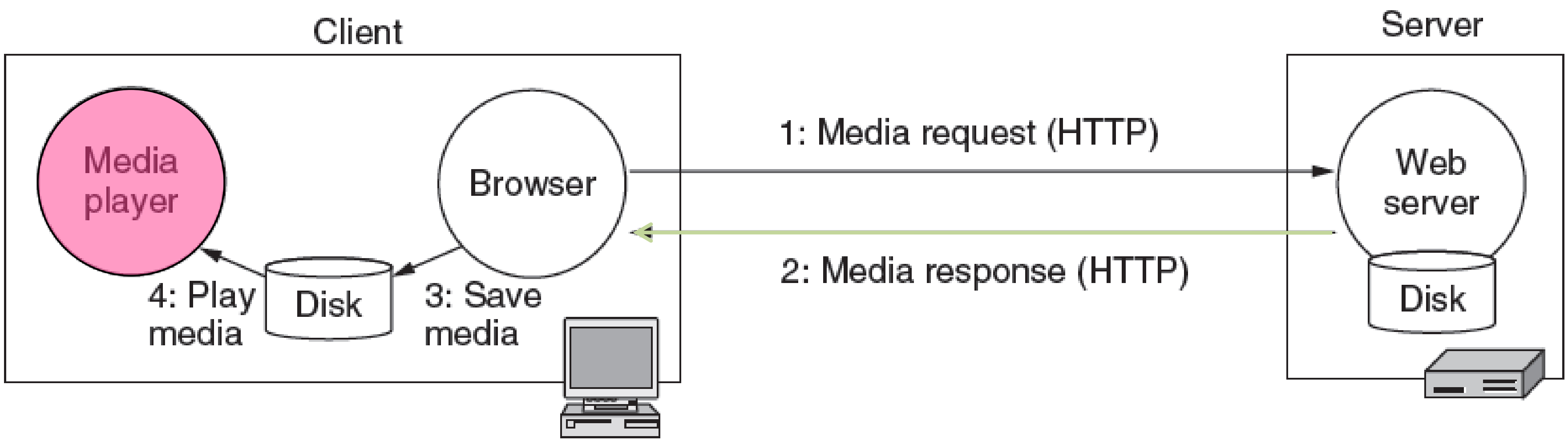
Digital Video (7)



Three consecutive frames

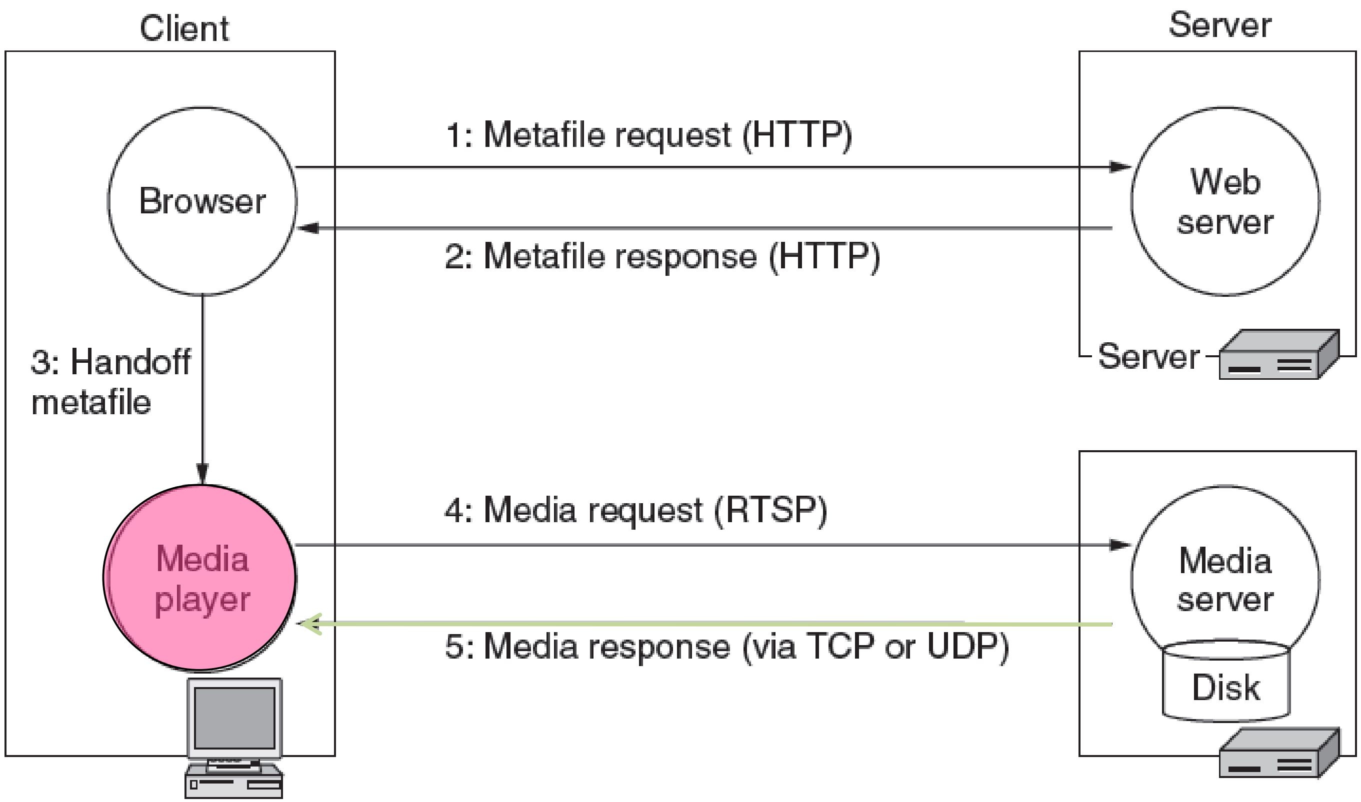
Streaming Stored Media (1)

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



Streaming Stored Media (2)

Effective streaming starts the playout during transport



Streaming Stored Media (3)

Major tasks of the media player:

1. Manage the user interface.
2. Handle transmission errors.
3. Decompress the content.
4. Eliminate jitter.

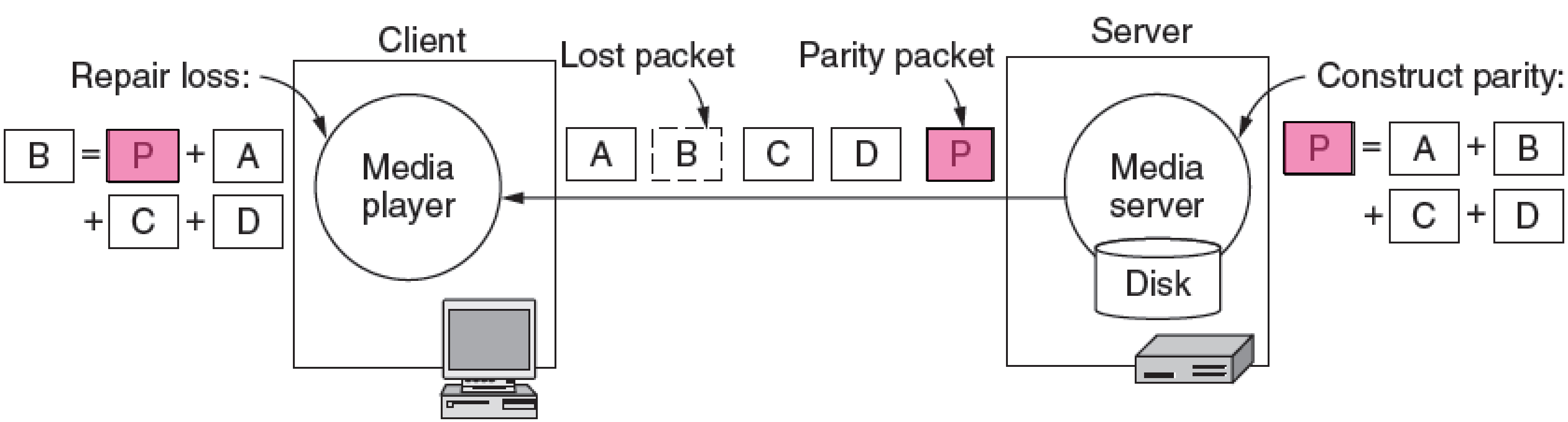
Streaming Stored Media (3)

Key problem: how to handle transmission errors

Strategy	Advantage	Disadvantage
Use reliable transport (TCP)	Repairs all errors	Increases jitter significantly
Add FEC (e.g., parity)	Repairs most errors	Increases overhead, decoding complexity and jitter
Interleave media	Masks most errors	Slightly increases decoding complexity and jitter

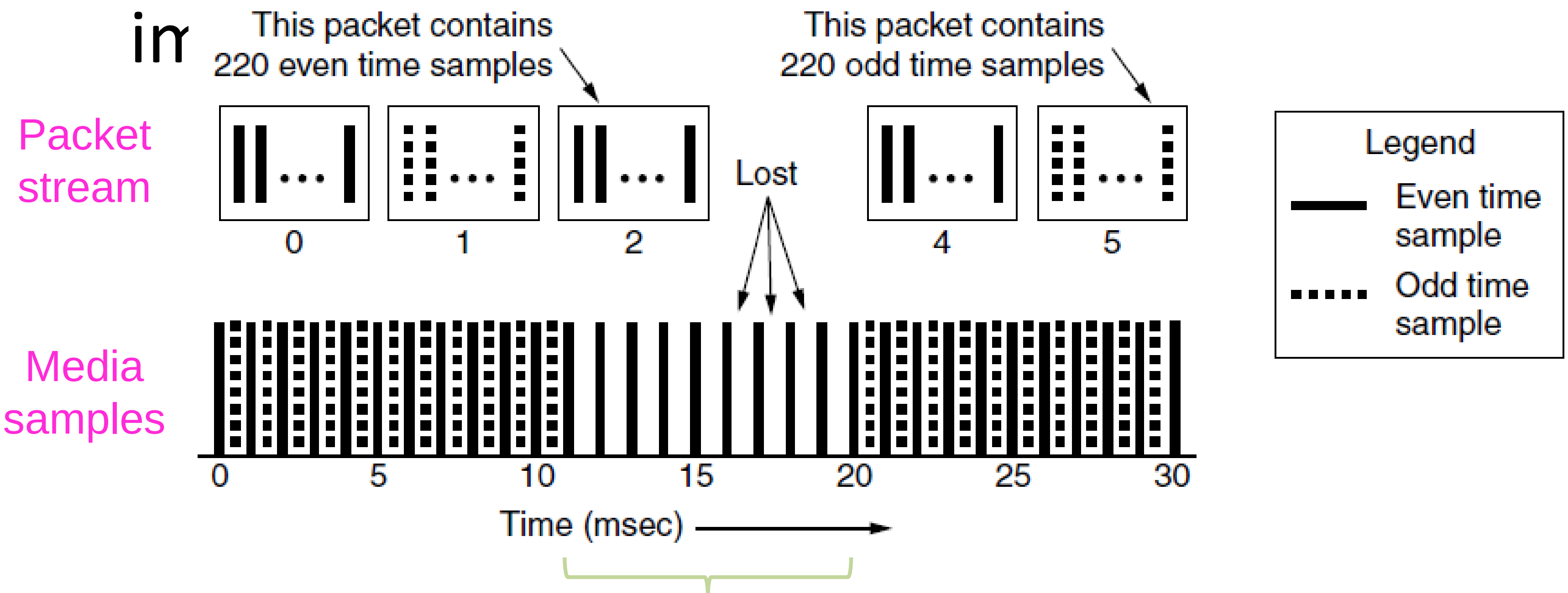
Streaming Stored Media (4)

Parity packet can repair one lost packet in a group of N



Streaming Stored Media (5)

Interleaving spreads nearby media samples over different transmissions to reduce the im

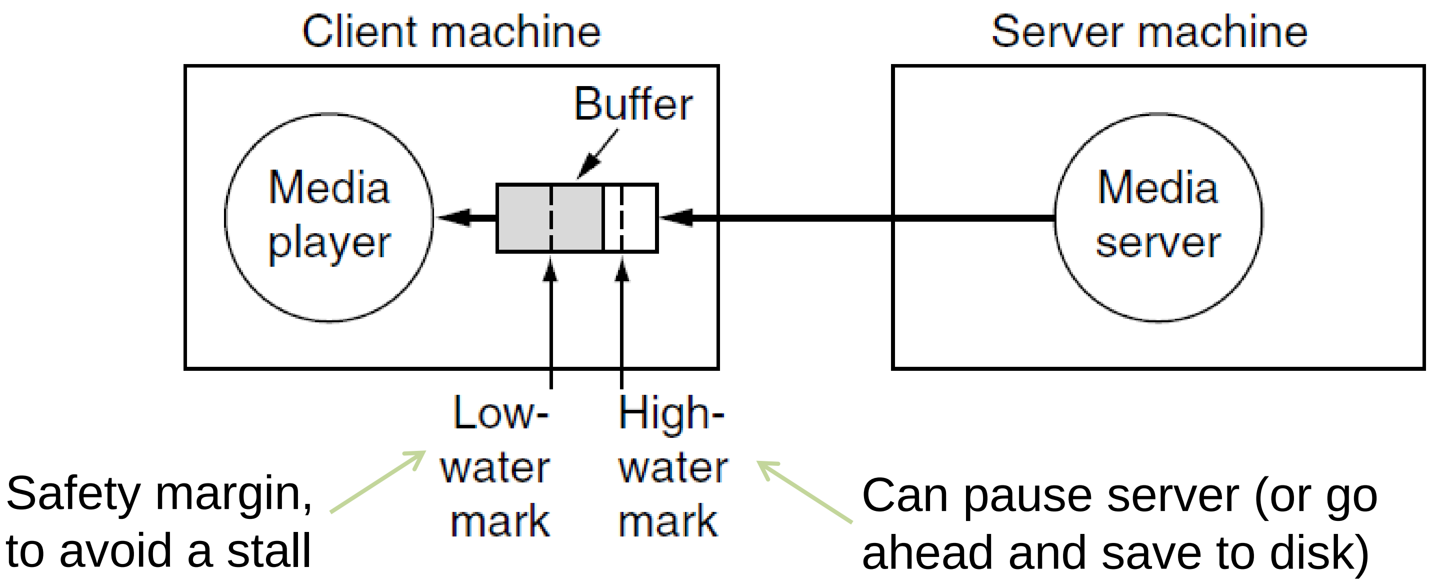


Loss reduces temporal resolution; doesn't leave a gap

Streaming Stored Media (6)

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

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



Streaming Stored Media (7)

RTSP commands

– Sent from player to server to adjust streaming

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 (1)

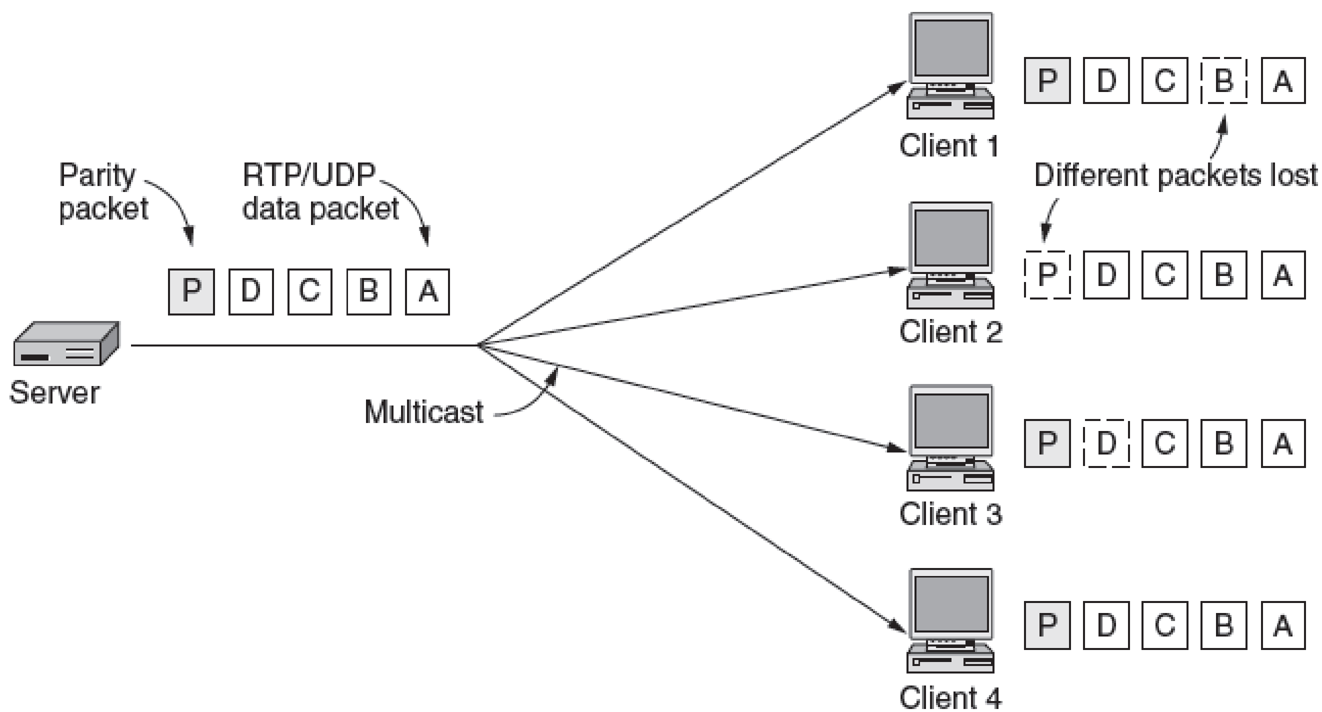
Streaming live media is similar to the stored case plus:

- Can’t stream faster than “live rate” to get ahead
  - Usually need larger buffer to absorb jitter
- Often have many users viewing at the same time
  - UDP with multicast greatly improves efficiency. It is rarely available, so many TCP connections are used.
  - For very many users, content distribution is used [later]

Streaming Live Media (2)

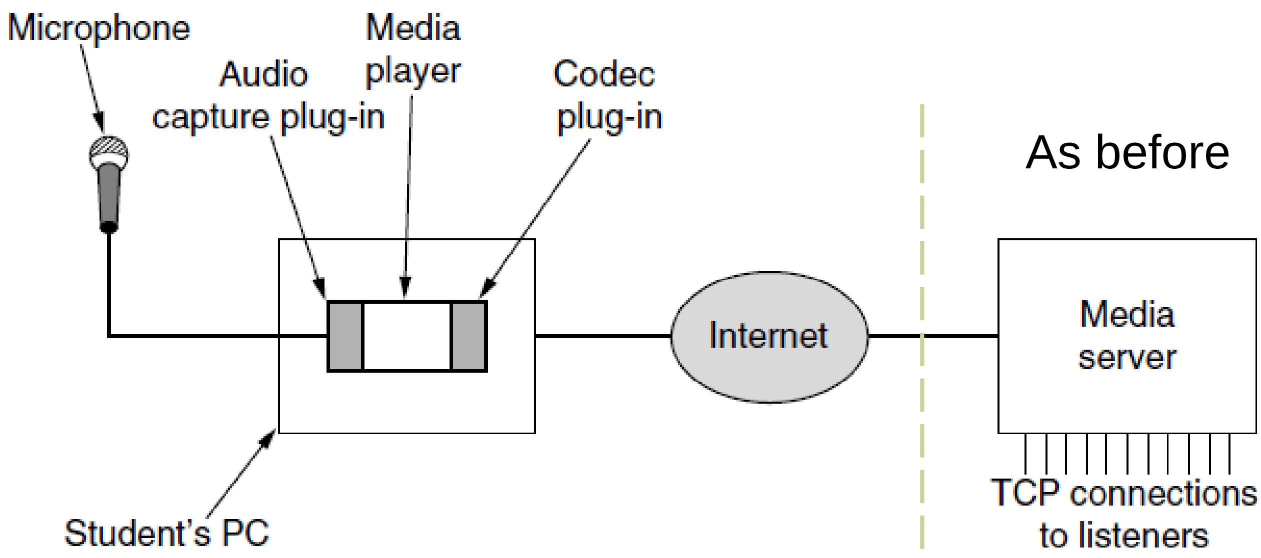
With multicast streaming media, parity is effective

d



Streaming Live Media (2)

Production side of a student radio station.



Real-Time Conferencing (1)

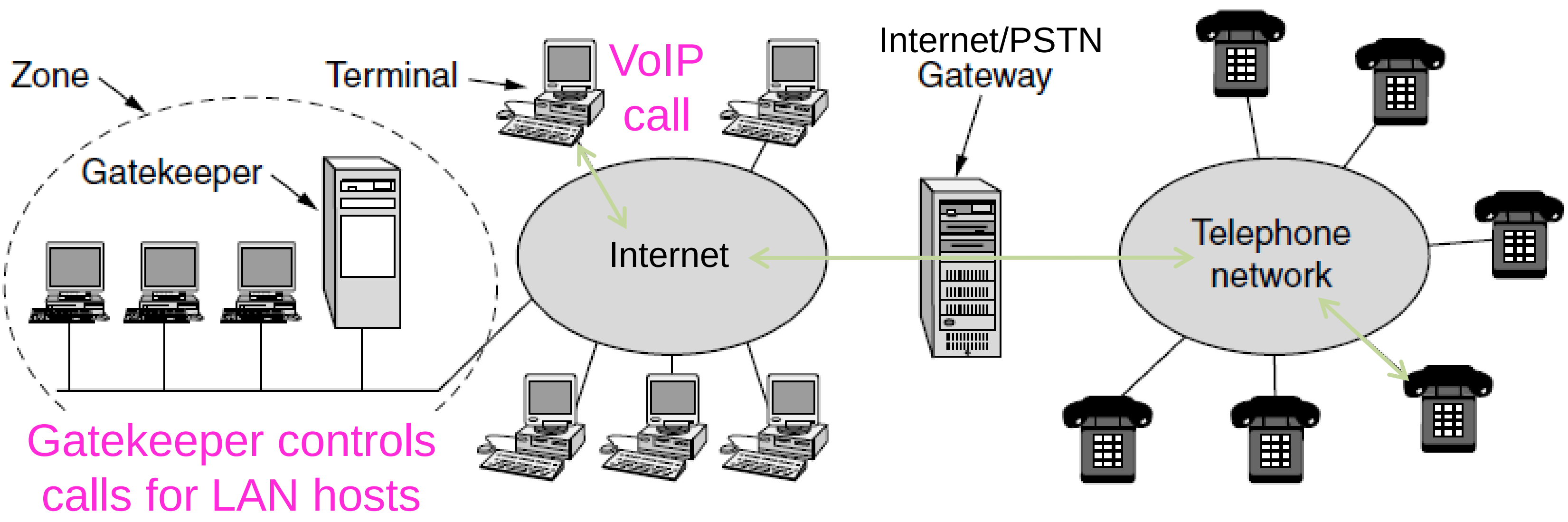
Real-time conferencing has two or more connected live media streams, e.g., voice over IP, Skype video call

Key issue over live streaming is low (interactive) latency

- Want to reduce delay to near propagation
- Benefits from network support, e.g., QoS
- Or, benefits from ample bandwidth (no congestion)

Real-Time Conferencing (2)

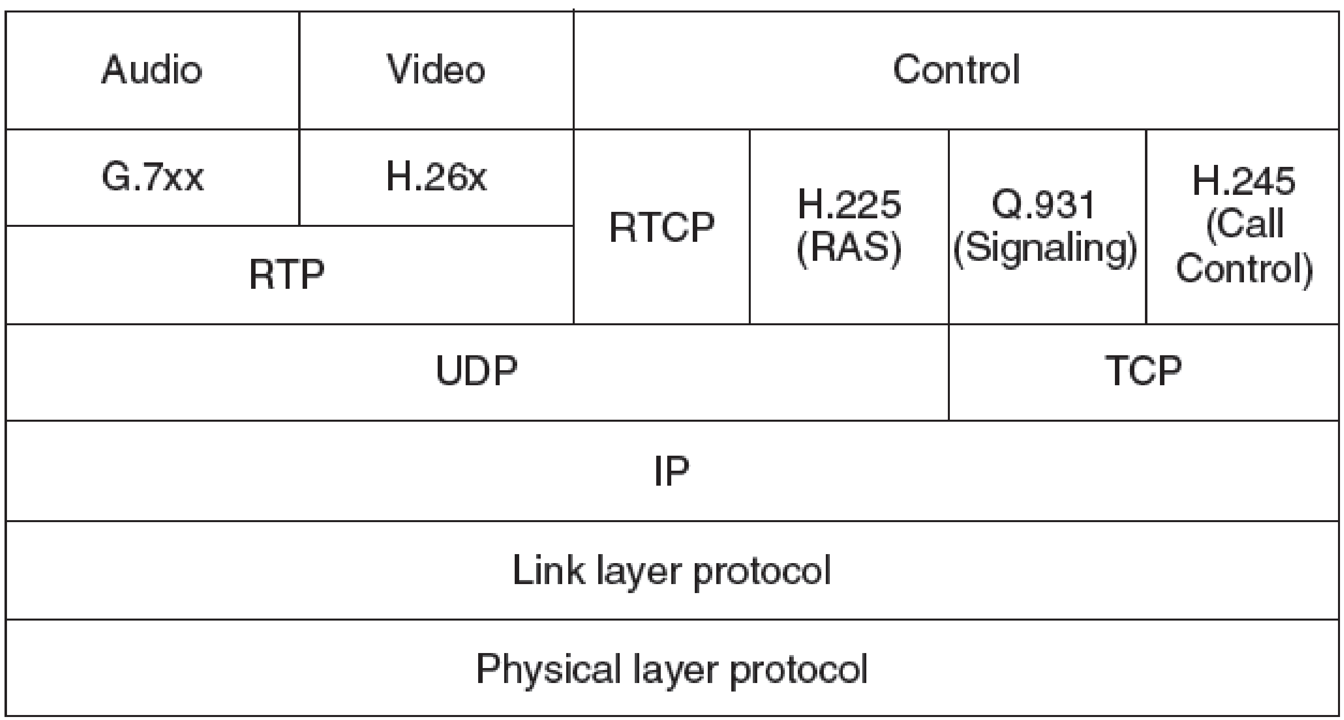
H.323 architecture for Internet telephony supports calls between Internet computers and PSTN phones.



Real-Time Conferencing (3)

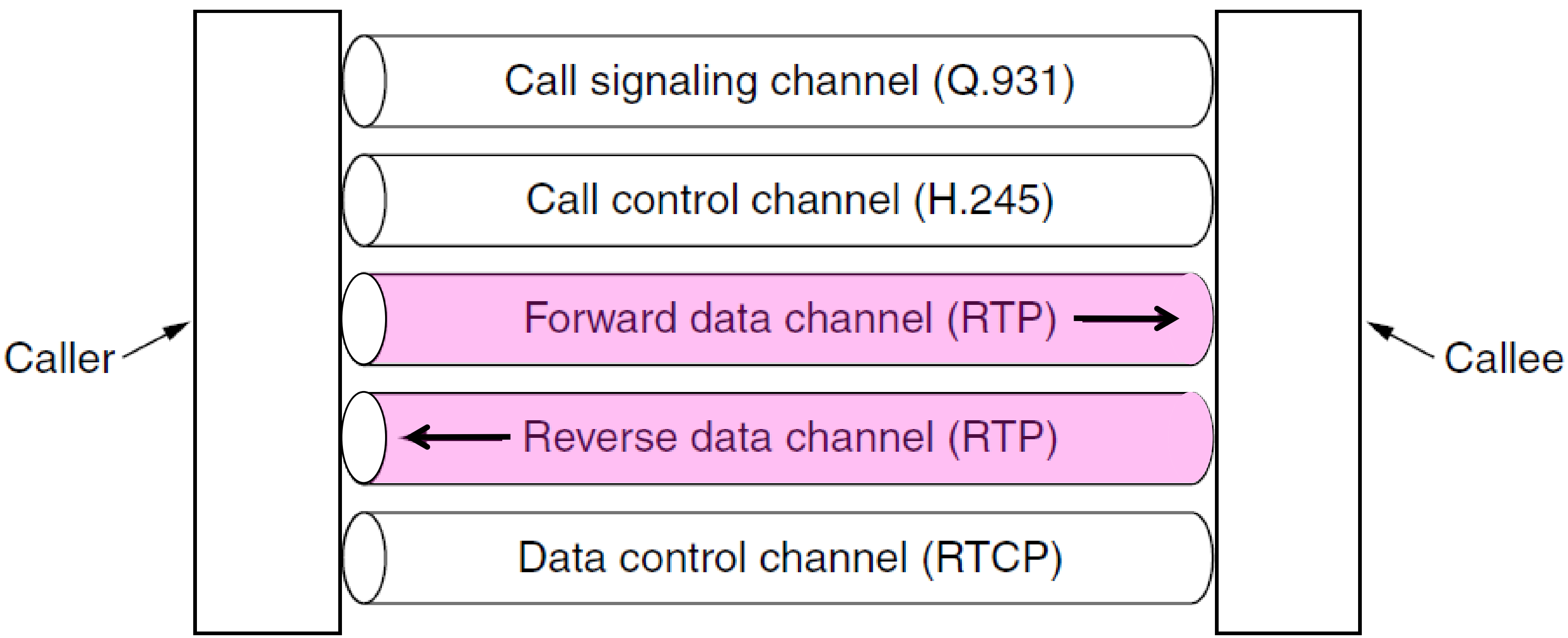
H.323 protocol stack:

- Call is digital audio/video over RTP/UDP/IP
- Call setup is handled by other protocols (Q.931 etc.)



Real-Time Conferencing (4)

Logical channels that make up an H.323 call



Real-Time Conferencing (5)

SIP (Session Initiation Protocol) is an alternative to H.323 to set up real-time calls

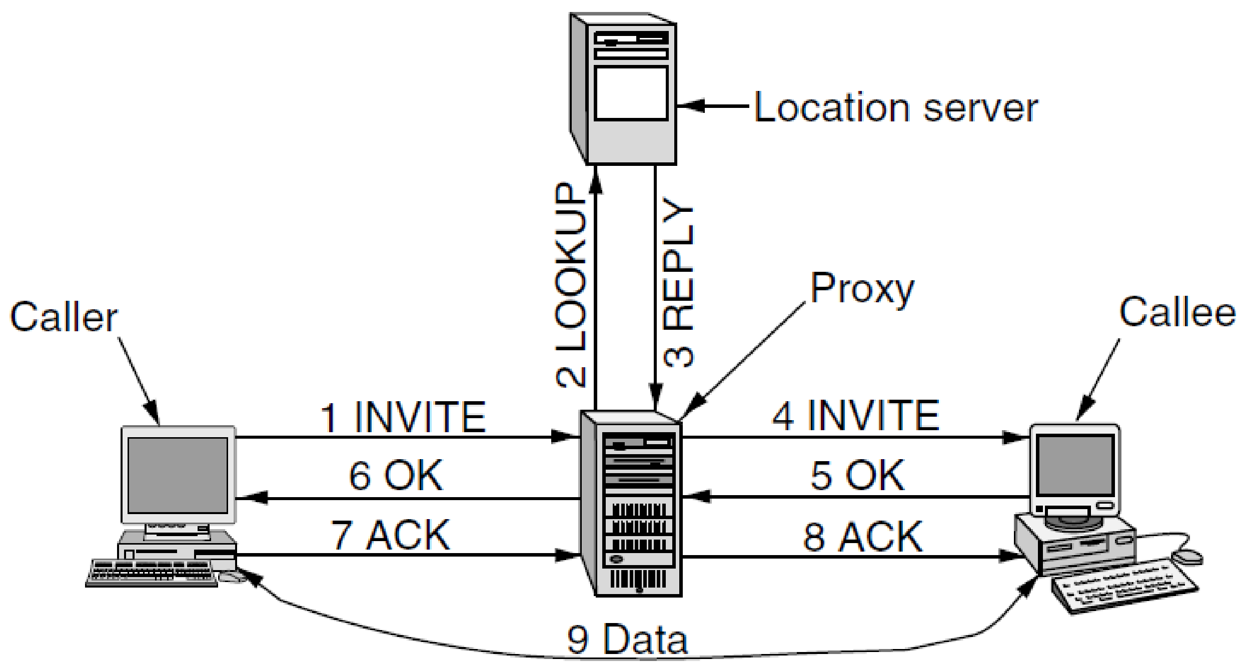
- Simple, text-based protocol with URLs for addresses

Method	Description
INVITE	Request initiation of a session
ACK	Confirm that a session has been initiated
BYE	Request termination of a session
OPTIONS	Query a host about its capabilities
CANCEL	Cancel a pending request
REGISTER	Inform a redirection server about the user's current location

Real-Time Conferencing (6)

Setting up a call with the SIP three-way handshake

- Proxy server lets a remote callee be connected
- Call data flows directly between caller/callee



Real-Time Conferencing (7)

Item	H.323	SIP
Designed by	ITU	IETF
Compatibility with PSTN	Yes	Largely
Compatibility with Internet	Yes, over time	Yes
Architecture	Monolithic	Modular
Completeness	Full protocol stack	SIP just handles setup
Parameter negotiation	Yes	Yes
Call signaling	Q.931 over TCP	SIP over TCP or UDP
Message format	Binary	ASCII
Media transport	RTP/RTCP	RTP/RTCP
Multiparty calls	Yes	Yes
Multimedia conferences	Yes	No
Addressing	URL or phone number	URL
Call termination	Explicit or TCP release	Explicit or timeout
Instant messaging	No	Yes
Encryption	Yes	Yes
Size of standards	1400 pages	250 pages
Implementation	Large and complex	Moderate, but issues
Status	Widespread, esp. video	Alternative, esp. voice

Comparison of H.323 and SIP.