CSc 450/550 Computer Networks Network Architectures & Client-Server Model

Jianping Pan

Summer 2007

Last lectures

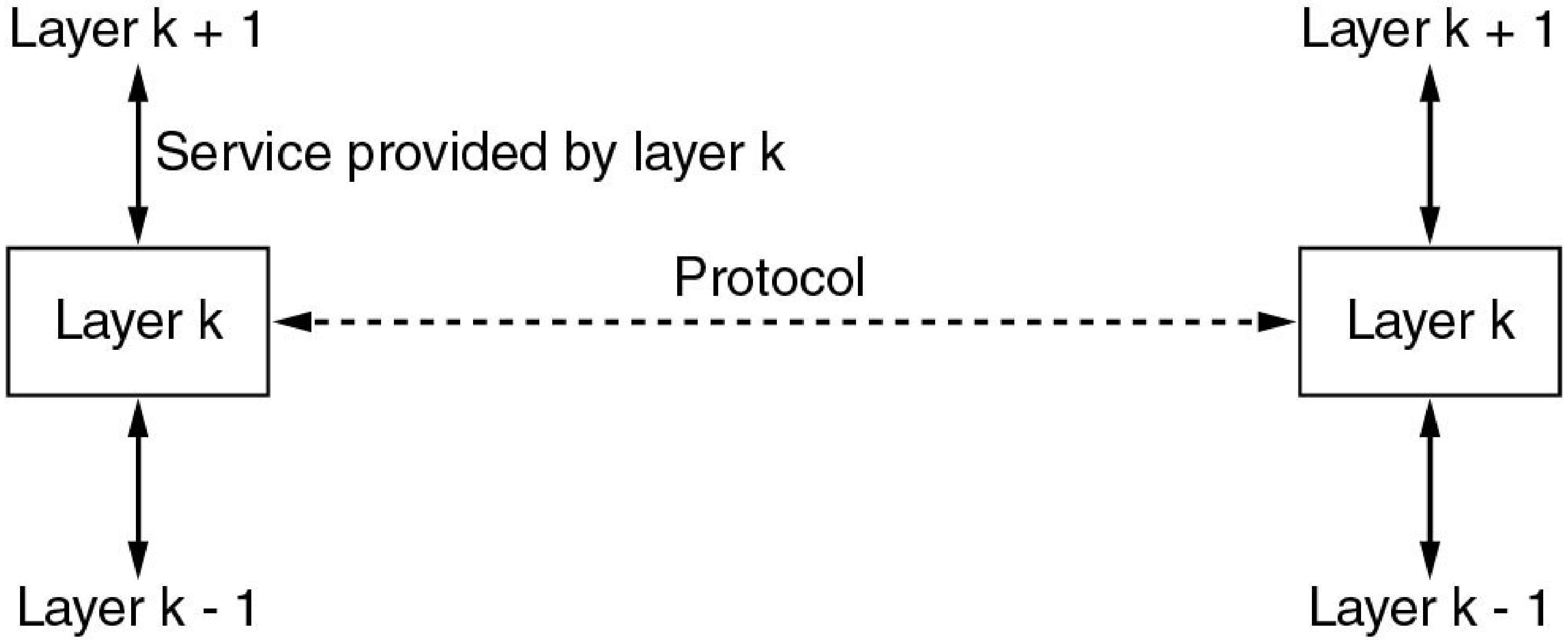
- So far, “nuts and bolts” views of the Internet
  - Internet evolution and state-of-the-art
  - UVicNet, BCNET, CA*Net4
  - Internet access technologies
  - Internet backbone technologies
- How does the Internet work indeed?
  - **network protocol** design and implementation
    - network protocol: machine-to-machine language
    - syntax, semantics, synchronization

Today's topics

- Network architectures
  - why do we need an architecture?
  - layers, services, protocols
- Service models
  - client-server model
  - client-server programming
- HTTP
  - a client-server application-layer protocol
  - HTML and simple HTTP request-reply

Network architectures

- Layered architecture (Q: why layered?)
  - service vs protocol



Network services

- Connection-oriented vs connectionless
  - connection establishment
  - data transfer
  - connection release
- Reliable vs unreliable
  - error checking
  - error correction
  - error recovery

Q: reliable services always connection-oriented?

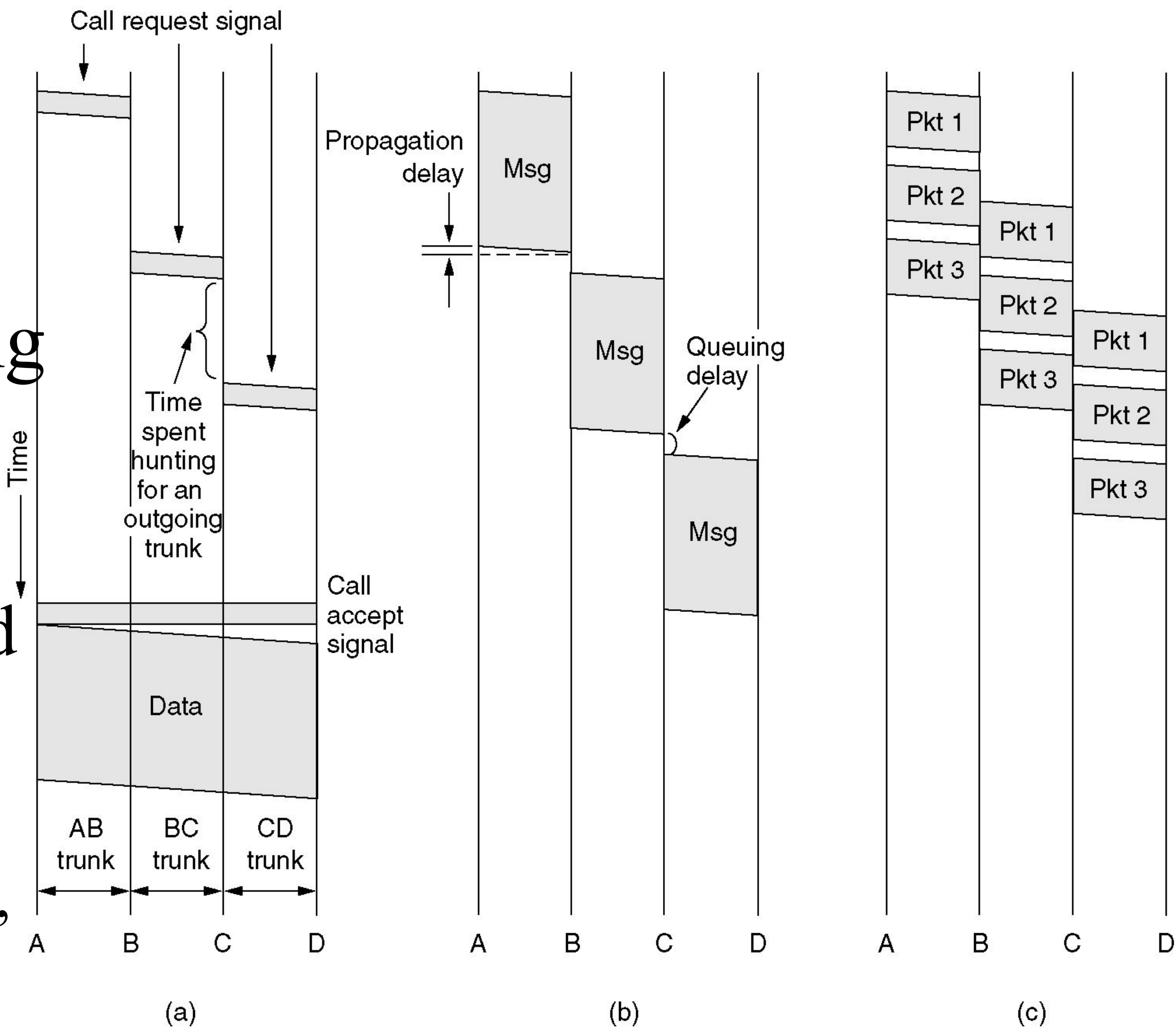
Switching technologies

- Circuit switching
  - e.g., telephone network
- Packet switching
  - virtual circuit
    - e.g., ATM
  - datagram
    - e.g., the Internet

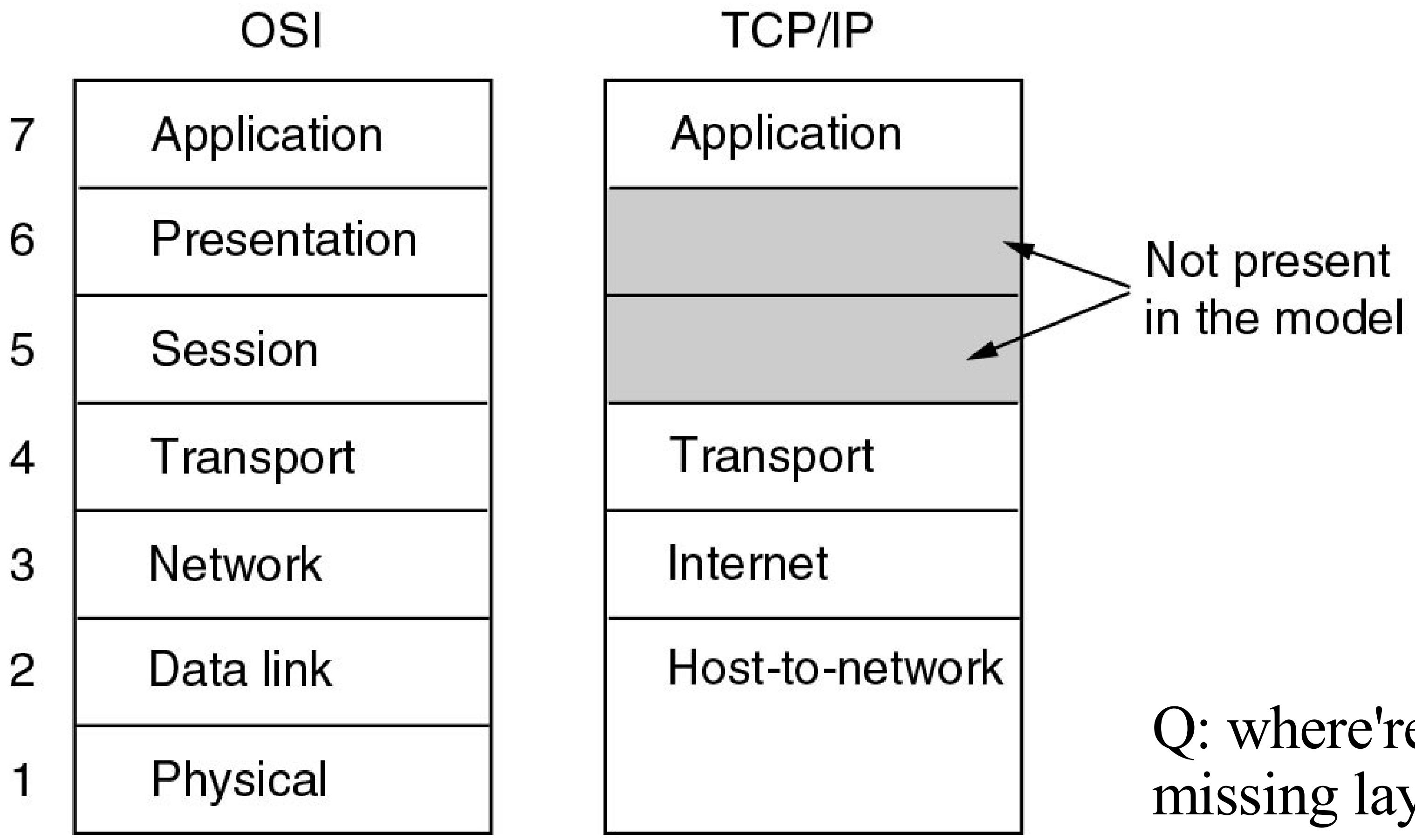
Q: IP/ATM/SONET/WDM?

- Circuit switching
- Message switching
- Packet switching
  - Internet: store-and-forward packet switching

Q: transmission, propagation, processing, queuing delay?



OSI and TCP/IP models

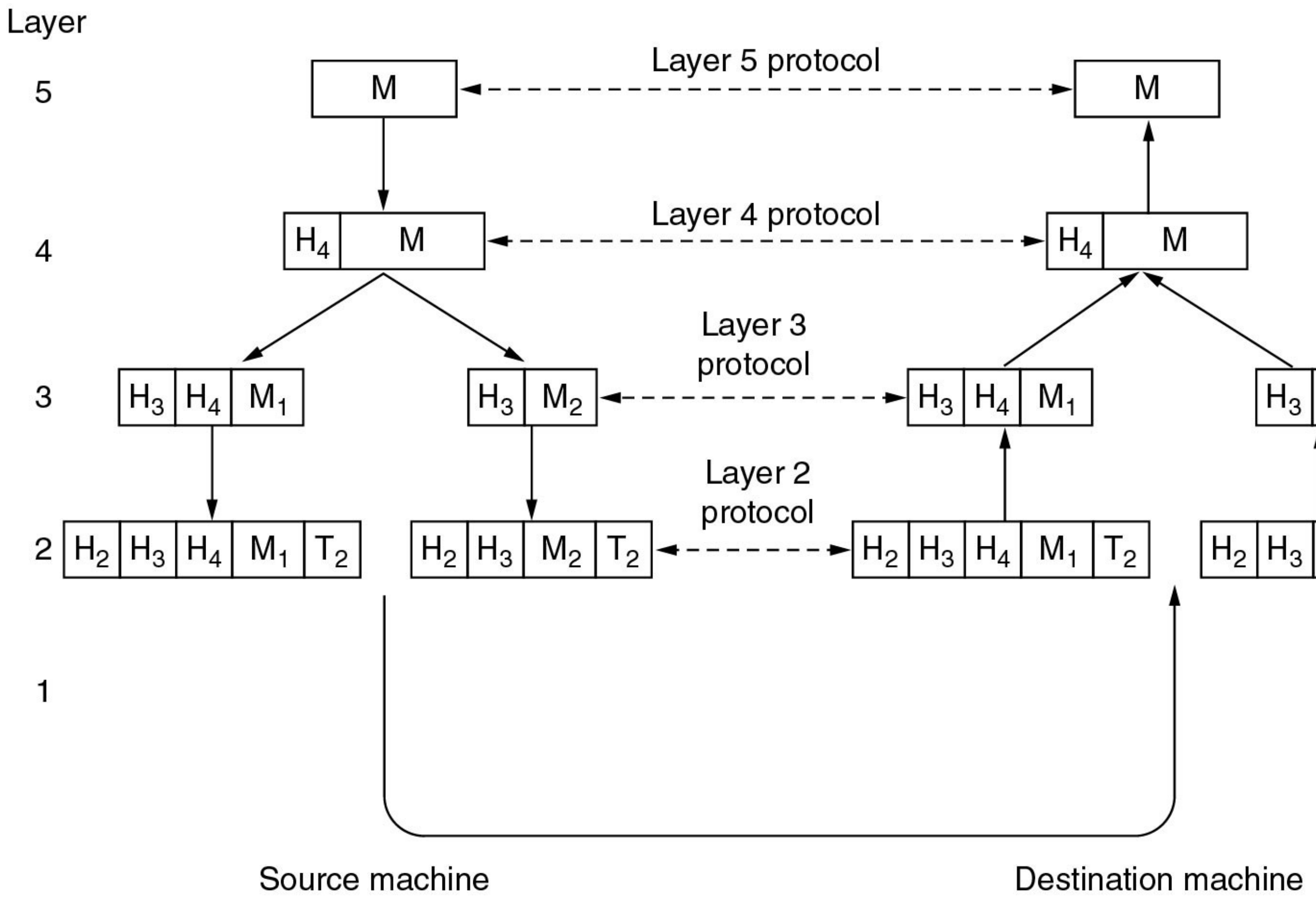


Q: where're the missing layers?

Protocol hierarchies

- HTTP message
- TCP segment
- IP packet
  - ➔ $M_1+M_2=M$
- Ethernet frame
- Bit stream

H: header; T: tail



Internet Protocol Suite

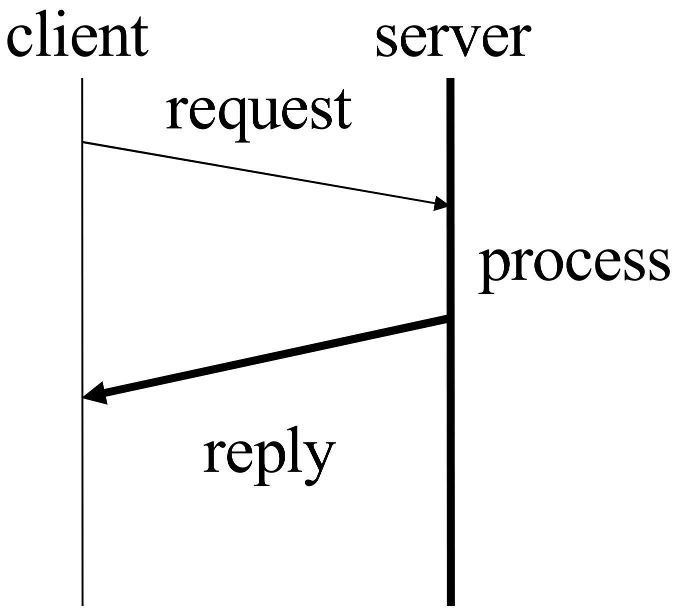
- "Hour-glass" model
  - application: telnet, ftp, email, Web, VoIP, ...
    - Web/HTTP: a client-server application layer protocol
  - transport: TCP, UDP, RTP, SCTP
  - network: IP
  - subnetwork: Ethernet, ATM, FDDI, PPP, FR, ...
- "Everything over IP"
- "IP over everything"

Service models

- Client-server model
  - server: services at well-known socket (WKS)
  - client: request services from anywhere!
  - client-server: request-reply transactions
- Later, client-*intermediary*-server model
  - web caching and content distribution
- In csc485b, peer-to-peer model
  - client/server-server/client

Client-server model

- Server
  - a process (running program)
  - on a (server) computer
  - (hosted in a server farm)
  - waiting for incoming requests
    - process and reply
- Client
  - a process on a client computer making requests



Q: many clients?

Client-server programming

- E.g., socket API
  - Client
    - socket()
    - connect()
    - send()
    - recv()
    - close()
  - Server
    - socket()
    - bind()
    - listen()
    - accept()
    - recv()
    - send()
    - close()

Socket API

- int socket(int *domain*, int *type*, int *protocol*);
  - domain
    - PF_INET (Internet protocol family), and others
  - type
    - **SOCK_STREAM (supported by TCP)**
    - SOCK_DGRAM (supported by UDP)
    - and others …
  - protocol
    - normally implied by socket type

Service offered by TCP

- Service offered by TCP
  - reliable
  - in-sequence
  - stream-like
  - data transfer
- TCP protocol mechanisms (stay tuned!)
  - connection management
  - flow, error, congestion control

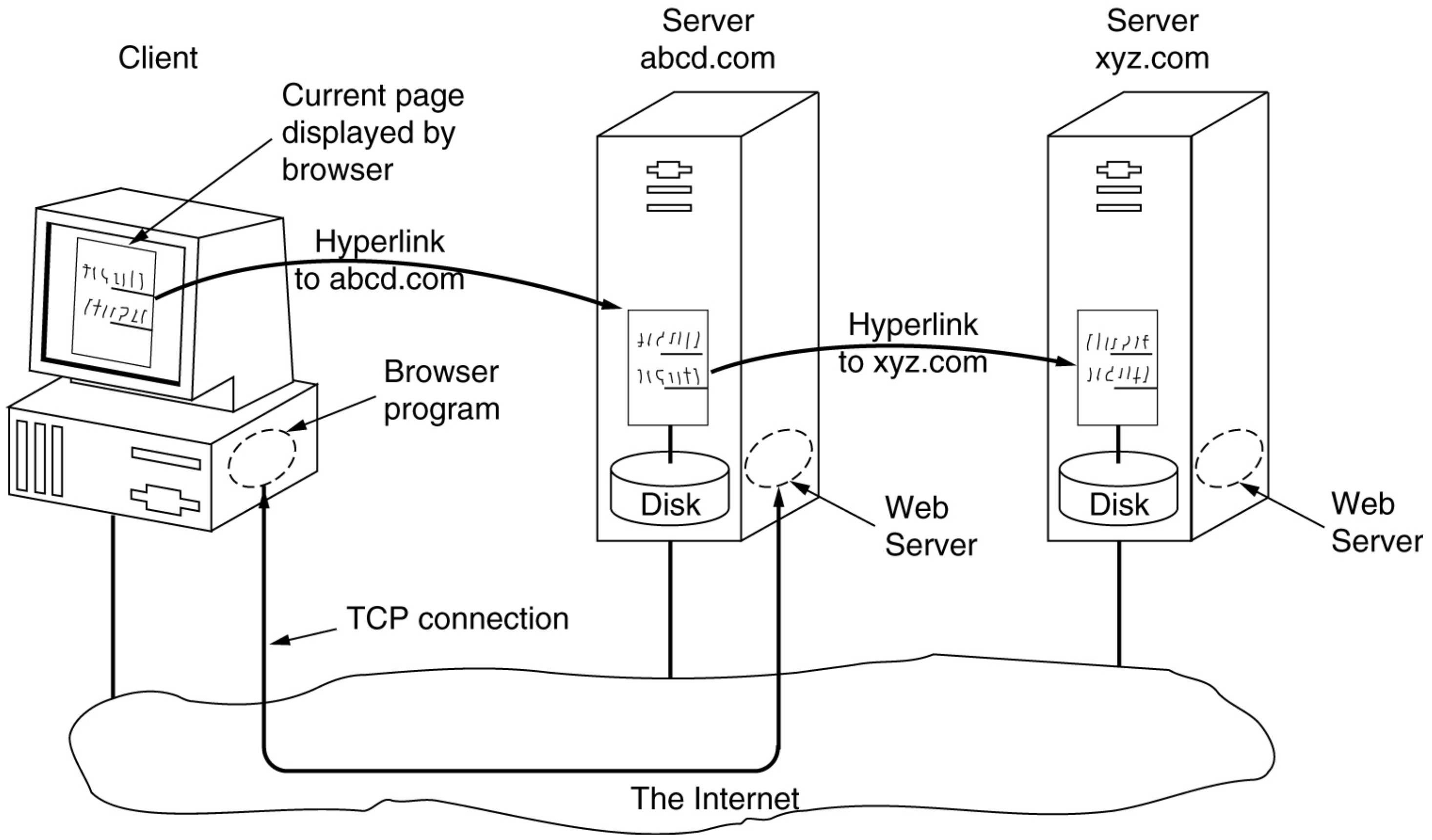
Socket, IP address, port number

- int **bind** (int *sockfd*,
  struct sockaddr *my_addr*,
  socklen_t *addrlen*);
  - struct sockaddr_in {short int *sin_family*;
    unsigned short int *sin_port*; //16-bit port#
    struct in_addr *sin_addr*; // 32-bit IP address
    unsigned char *sin_zero*[8];};
  - struct in_addr {unsigned long *s_addr*;};
- /etc/services, /etc/hosts, DNS

Worldwide web

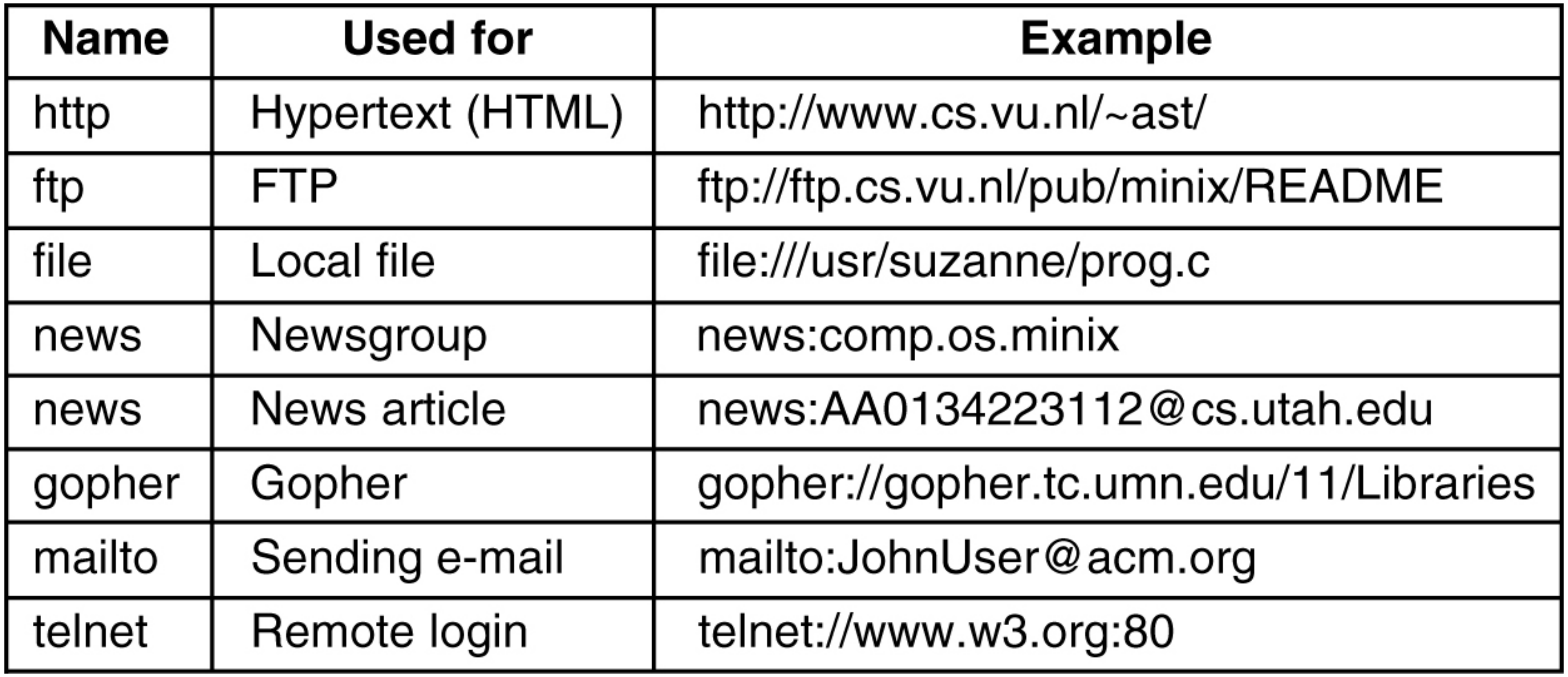
- Tim Berners-Lee
  - 1989, CERN
- Hypertext and hypermedia
  - linked documents
- Marc Andreessen
  - 1993, Mosaic, NCSA@UIUC
- Netscape Comm
  - Netscape navigator vs MS Internet explorer

Web overview



Uniform Resource Locator

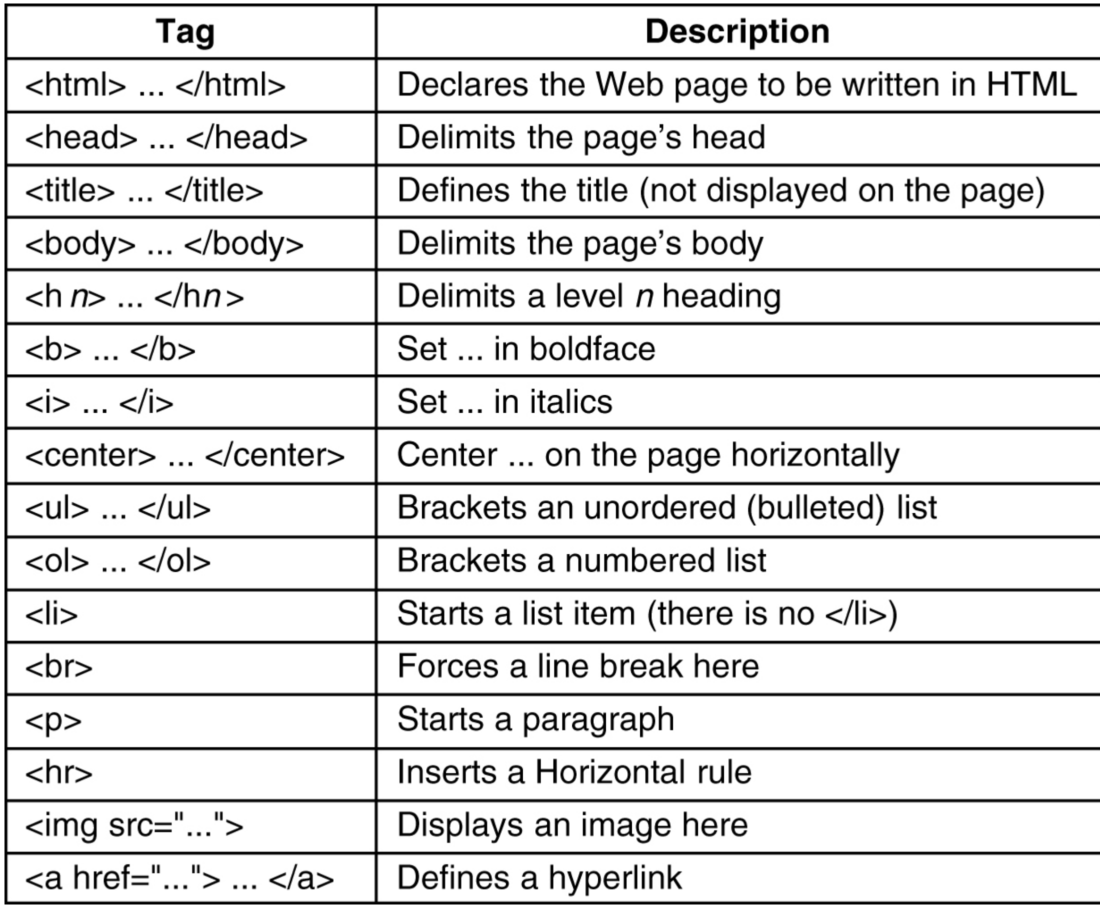
- http://user:pass@host:port/path/file?input



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

HTML tags

- Anchors
  - <a href="...">...</a>
- Objects
  - <img src="...">



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
<b> ... </b>	Set ... in boldface
<i> ... </i>	Set ... in italics
<center> ... </center>	Center ... on the page horizontally
<ul> ... </ul>	Brackets an unordered (bulleted) list
<ol> ... </ol>	Brackets a numbered list
<li>	Starts a list item (there is no </li>)
<br>	Forces a line break here
<p>	Starts a paragraph
<hr>	Inserts a Horizontal rule
<img src="...">	Displays an image here
<a href="..."> ... </a>	Defines a hyperlink

HTTP

- Hyper text transfer protocol
  - application layer protocol, ASCII format
    - HTTP/1.0: RFC1945 (1996); 1.1: RFC2068 (1997)
  - typical client-server model: request-reply
    - client (browser): Navigator, Mozilla, Opera, IE
    - server (web server)
      - Apache, Microsoft Internet information server (IIS)
  - normally uses service offered by TCP
    - http: 80; https: 443 (HTTP over SSL over TCP)

HTTP requests

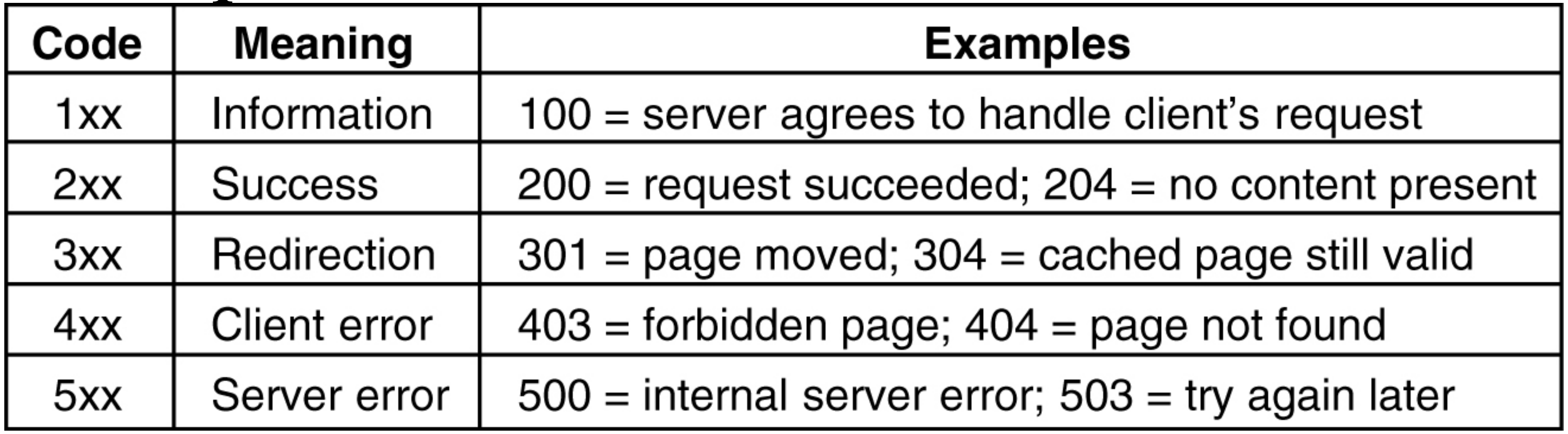
- Request methods

Method	Description
GET	Request to read a Web page
HEAD	Request to read a Web page's header
PUT	Request to store a Web page
POST	Append to a named resource (e.g., a Web page)
DELETE	Remove the Web page
TRACE	Echo the incoming request
CONNECT	Reserved for future use
OPTIONS	Query certain options

- Request parameters (control headers)

HTTP responses

- Response codes



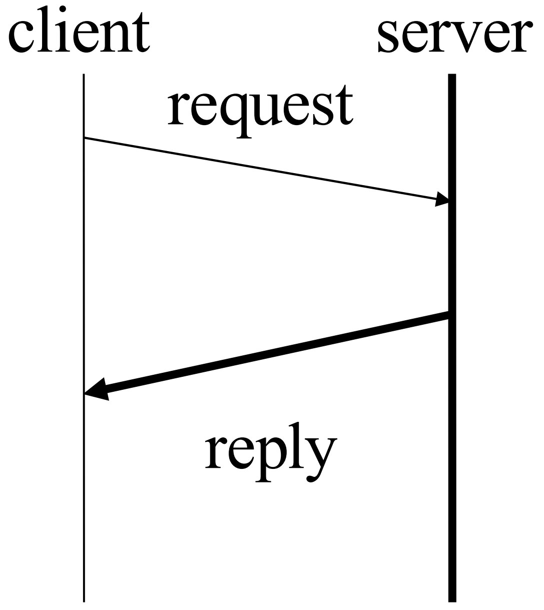
Code	Meaning	Examples
1xx	Information	100 = server agrees to handle client's request
2xx	Success	200 = request succeeded; 204 = no content present
3xx	Redirection	301 = page moved; 304 = cached page still valid
4xx	Client error	403 = forbidden page; 404 = page not found
5xx	Server error	500 = internal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

  - 400: bad request
- Response parameters
- Response data

HTTP examples

- # wget -d www.google.com

```
Connecting to www.google.com:80... Caching www.google.com <-> 66.102.7.104
Created fd 3.
connected!
---request begin---
GET / HTTP/1.0
User-Agent: Wget/1.7
Host: www.google.com
Accept: */*
Connection: Keep-Alive
---request end---
HTTP request sent, awaiting response...
HTTP/1.0 302 Found
Location: http://www.google.ca/
Cache-Control: private
Content-Type: text/html
Server: GWS/2.1
Content-Length: 218
```



Q: syntax, semantics, synchronization?

Web browsing examples

- http://www.a.com/index.html
```
<html>
<img src="http://www.a.com/x.gif">
<a ref="http://www.b.com/y.gif">
<img src="http://www.b.com/z.gif"></a>
</html>
```
- In your favorite web browser
  - URL: http://www.a.com
  - how many HTTP requests?

This lecture

- Internet architecture
- Client-server model
- HTTP
  - HTML basics: anchors and objects
  - simple HTTP request and response
- Explore further
  - how efficient is HTTP?
    - Ethereal lab next Wednesday!

Project 1: multi-thread web server

- Specification
- Specification walk-through
- Design assistance
- Implementation assistance
- Demo
- Submission

For CSc 550 students

- Course project (15%)
  - course project topics and ideas
    - due: May 18
  - 1-page course project proposal
    - due: June 1
  - midterm checkpoint
    - due: June 29
  - final deliverables
    - course project report and prototype
    - due: the end of the term

Next lectures

- May 24: HTTP (2)
  - HTTP connections
    - how HTTP and TCP can fit better
  - Web caching
- May 28: DNS
- May 31: 1st midterm exam