CSc 450/550
Computer Networks
Worldwide Web

Jianping Pan

Summer 2006

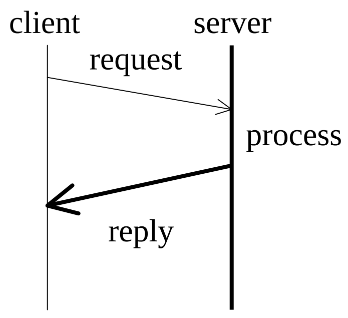
Internet applications

- Traditionally
  - remote login: e.g., telnet
  - file transfer: e.g., FTP
  - electronic mail: e.g., email
- More recently
  - worldwide web: the Web!
  - multimedia streaming
  - voice over IP (VoIP)

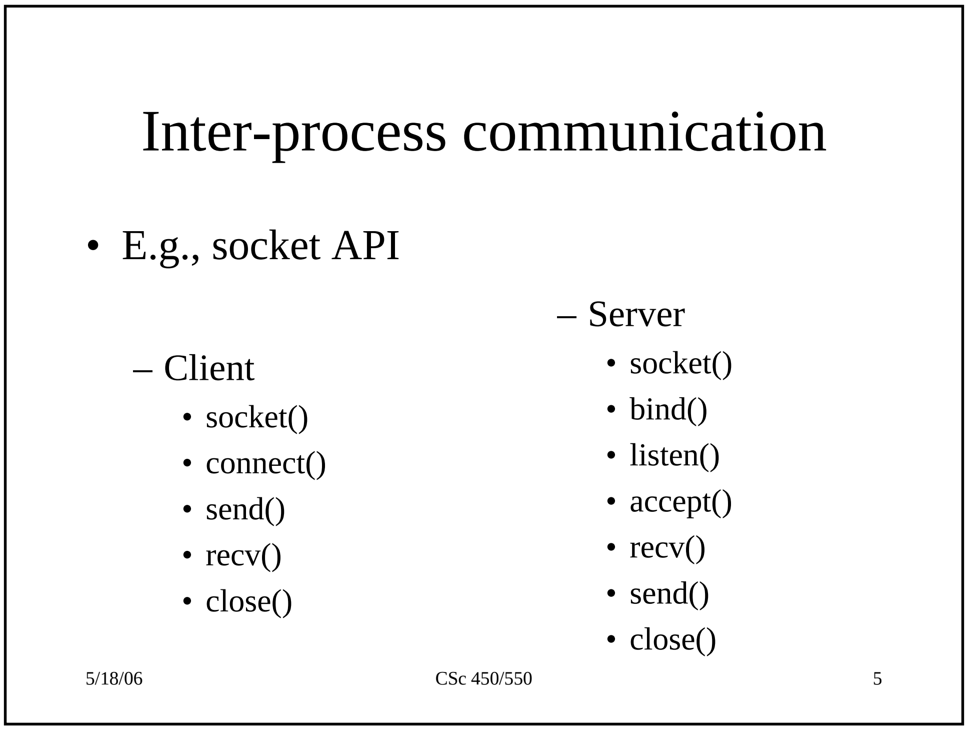
Service models

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

Client-server model



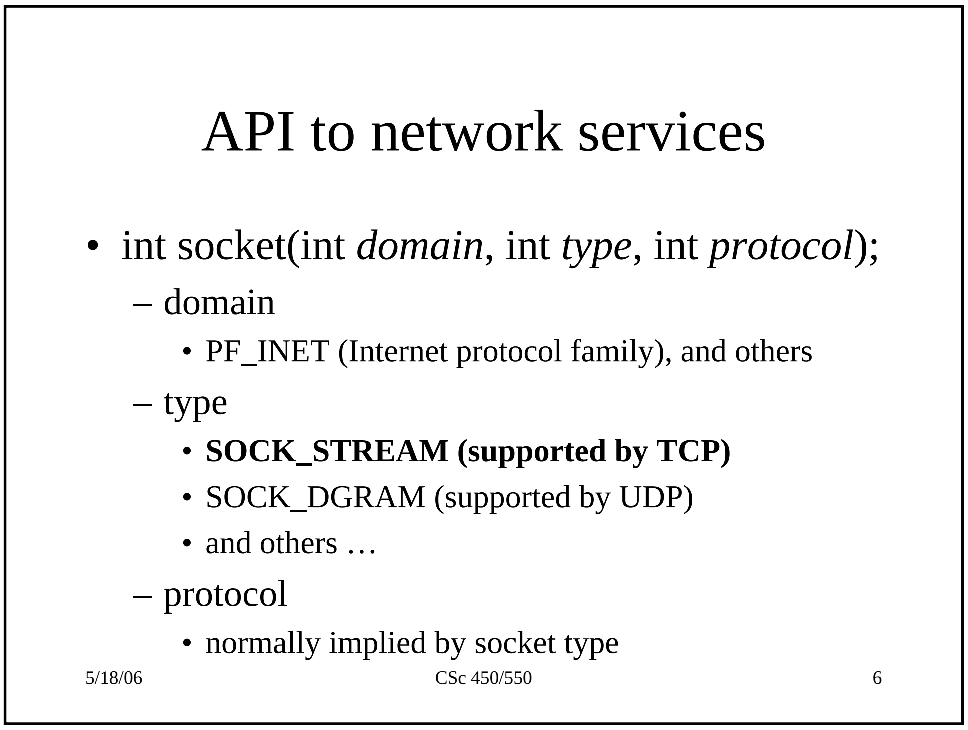
- Server
  - a process (running program)
  - on a (server) computer
  - (hosted in a server farm)
  - waiting for incoming requests
    - process and reply
- Client
  - a process on a client computer making requests



Inter-process communication

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

5/18/06 CSc 450/550 5



API to network services

- 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

5/18/06 CSc 450/550 6

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

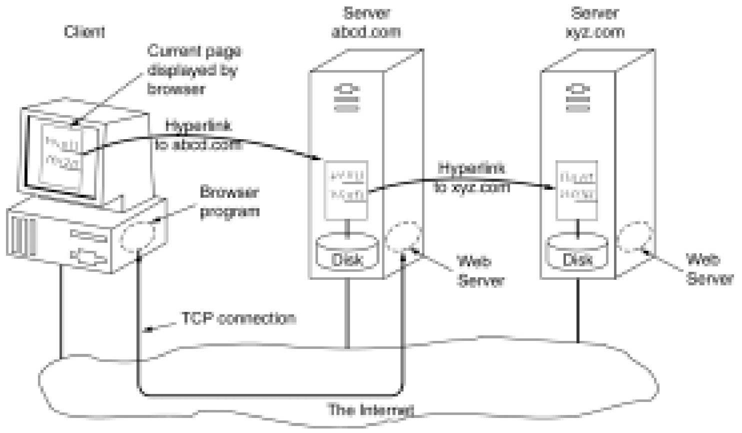
- int **bind** (int *sockfd*,
  struct sockaddr *my_addr*,
  socklen_t *addrlen*);
  - struct sockaddr_in {short int *sin_family*;
    unsigned short int *sin_port*; //16-bit port#
    struct in_addr *sin_addr*; // 32-bit IP address
    unsigned char *sin_zero*[8];};
  - struct in_addr {unsigned long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

5/18/06 CSc 450/550 9

Web overview



5/18/06 CSc 450/550 10

Web pages

- Universal resource locator (URL)
  - – e.g., http://www.cs.uvic.ca/~pan/csc450
- Hyper text markup language (HTML)

```
<html><title>UVic CSc 450/550: Computer Communications and Networks
    (Summer 2006)</title>
<center><h1>CSc 450/550: Computer Communications and
    Networks</h1></center>
<h3>Course <a
    href="http://www.csc.uvic.ca/courses/summer%202006/csc/450-
    550.html">Outline</a></h3>
```

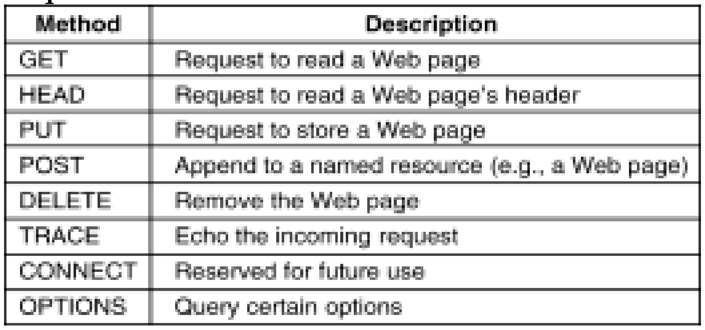
- Anchors and objects

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

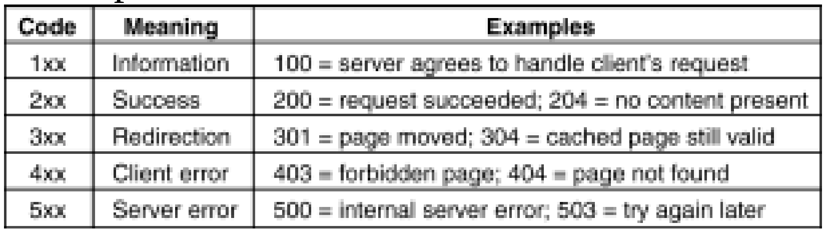


Method	Description
GET	Request to read a Web page
HEAD	Request to read a Web page's header
PUT	Request to store a Web page
POST	Append to a named resource (e.g., a Web page)
DELETE	Remove the Web page
TRACE	Echo the incoming request
CONNECT	Reserved for future use
OPTIONS	Query certain options

- 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

  – 505: not implemented
- 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
```

HTTP connections

- Non-persistent connections
  - one object per TCP connection
    - parallel connections
    - default in HTTP/1.0
- Persistent connections
  - multiple objects through one TCP connection
    - if all from the same server
    - default in HTTP/1.1
  - non-pipeline vs pipeline

Client-server states

- HTTP itself is stateless
  - server: serve individual requests from any client
  - client: make a series of requests to any server
- Many applications require states
  - client can choose to keep state (cookie)
  - cookie issued by server’s *backend* servers
  - client represents cookie in subsequent requests
    - let server remember you!

Web caching

- Aggregate user requests
  - by caching responses to previous requests
  - explore locality: same requests may occur soon!
  - reduce response time and traffic load
- Consistency control
  - strong consistency: if-modified-since
    - response: “HTTP/1.0 304 Not Modified”
  - weak consistency: time-to-live (TTL)

Content delivery

- Move content closer to end users
  - content distribution
- Redirect users to closer servers
  - information retrieval
  - how do they do that?
    - e.g., Akamai
    - DNS-based server selection (stay tuned!)

This lecture

- HTTP
  - client-server model
    - other alternatives
  - Web and hypertext: HTML
  - HTTP request and response
- Explore further
  - how efficient HTTP protocol is?
  - especially on bandwidth-constrained networks

Next lecture

- Domain name system (DNS)
  - Read CN 7.1