Resource Managers become Servers

- Story so far:

- Resource Mother is
  - just a process which mothers a resource
    - (hardware e.g. printer, data eg a flag or semaphore)
  - you access the resource by communicating with its mothering process

A server

- Is a resource mother, PLUS:
- a server is able to accept requests from many clients concurrently.
- It avoids becoming *blocked* for "long" periods of time

A server process : mutex

- It often serializes requests so their effects
  - are atomic
- (effectively implements mutual exclusion
  - on the resource)

Server process: state considered harmful

- the server tries to be *stateless*
  - so that transactions do not affect one another
  - so that crash recovery is facilitated

Statelessness

- More precisely,
  - it has a fixed initial state to which it returns after each access request is served
  - access requests should have no effect on this initial state

But Statelessness

- Cannot be forsaken!
  - Many *transactions* are stateful:
    - e.g., contents of my online shopping basket
  - All *sessions* are stateful
    - response to your command depends on
      - it and on
      - previous commands

Names for servers

- the server has a name ("print service") which is
  - well-known and can be *translated or resolved*
  - into its process id (Pid), object id (oid) or address.

- a server can be someone else's client too.

Server structures:

- Monolithic server or
- Manager plus workers
  - manager avoids becoming *blocked* by passing service requests to *workers*
  - flavours:
    - administrator
    - manager, etc
    - see WM Gentleman, in *Software Practise & Experience*

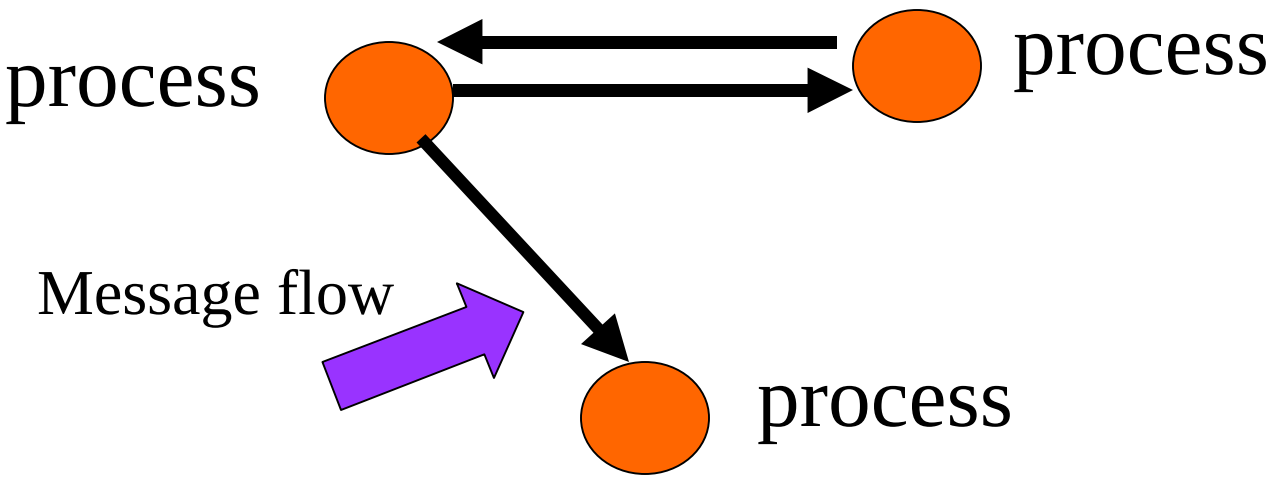
Server structures

- God did not ordain the client-server architecture:

- Gnutella is NOT client-server
  - (it's peer to peer), as in . . .

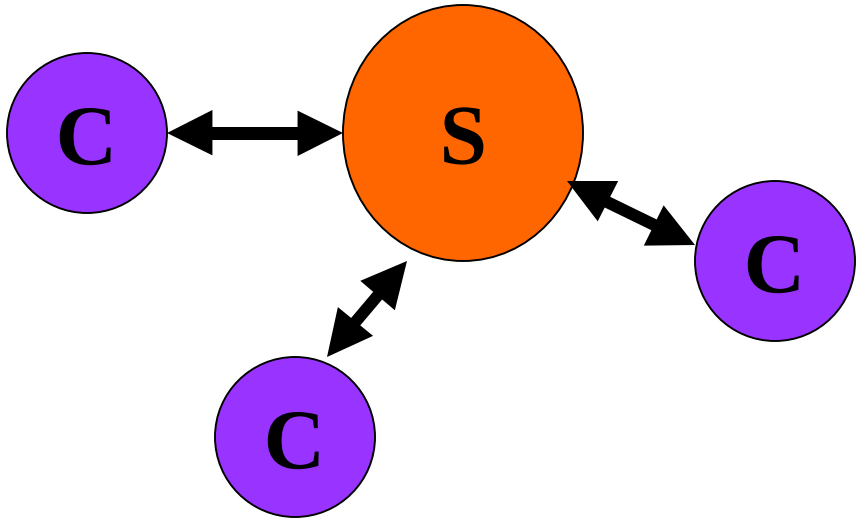
Computational models again

- 1] peer to peer: set of equal (peer) processes communicating by messages



Computational models

- 2] Set of *clients* getting *services* from *servers*
  - (or set of workers getting work from a manager)



Computational models

- 3] object-oriented

Computational models

- They aren’t all mutually exclusive:
  - clients & servers can be processes communicating via messages (and often are)
  - any of the processes can be an Object

Openness

- ill-defined buzzword
- suggests that interfaces of interest are defined, documented, stable and accessible.
- in particular
  - a defined, documented, stable, (single) ipc mechanism (SUN's SPRING OS)

Parallelism (speed!, NOT Concurrency) promoted by

- multiple cpus
- multiple users interacting simultaneously
- multiple processes
- (independent threads of control)

  per program

Ability to Scale Up

- no explicit or implicit size limits in the design
  - number of cpus
  - number of computers
  - number of subnet nodes . . .
- so that performance can be increased without limit
- Easier said than done!

Common performance bottlenecks

- Physical:
  - bandwidth of shared memory
  - bandwidth of busses

Performance bottlenecks

- speed of
  - disc arms (10-20 *msec*)
  - subnet ports
  - subnet links (but fiber is here! 32 * 10^{12} bps)
  - subnet logical channels
    - (after protocol overheads, bad news)

scaling-up:

- sizes of name spaces
  - host address space (IP addresses)
  - switch address space (Level 2 addresses)
  - process space per host
  - primary memory address space (killed the VAX)
- see: Apertos variable-length addresses

Fault Tolerance

- requires multiple hardware units (cpu, disc, . . .) &

  multiple software units (class copies)

  as provided by distributed systems.

- necesssary but far from sufficient

Transparency (t) flavours

- Access operation t:
  - local & remote accesses by same operations
    - (e.g. read, write, or send, receive)

Transparency (t) flavours

- Location t:
  - need not know object location
- concurrency t.
  - concurrent processes sharing an object with no interference (mutex, atomicity)

Transparency (t) flavours

- replication t.
  - can replicate e.g. a print server for fault tolerance or performance

  WITHOUT changing application code

  - (How?? hint: name-address translation)
  - Recent Work: SUN's Project Orion

Transparency (t) flavours

- migration t:
  - can migrate objects on the fly without changing code

Transparency (t) flavours

- performance t:
  - system can be reconfigured on the fly

Transparency (t) flavours

- scaling t.
  - system can be scaled up and down on the fly

Web - ology

(new to 3rd edition)

History

- First there was *hypertext*
  - a word can also be a *pointer*
- Then there was *gopher*
  - a hypertext pointer could point to a (text) file on another computer (filesystem hacks)
- finally there was The Web
  - Uniform Resource Locators (URL)
  - client-server architecture & protocol (http)

Details: html

< IMG SRC = “http://www.cdk3.net/WebExample/earth/.jpg”>
1

webpage URL

image

<P> *paragraph delimiter* *2*

welcome to earth; also see the 3

<A HREF =
“http://www.cdk3.net/webexmpl/moon.html”>Moon</A> 4

a webpage link with a URL as pointer value pointer

<P> *5*

html:

- Which all displays as

welcome to earth; also see the <u>moon</u>

Uniform Resource Locator Needs to specify

- resource type (protocol)
  - http: hypertext, (hypertext transfer protocol)
  - ftp: file (file transfer protocol ftp)
  - mailto: email (email protocol)

Uniform Resource Locator Needs to specify

- Location
  - hostname cdk3.net
  - filename within host webexample/moon
  - typing info .html

cdk3.net/webexample/moon/ .html

ftp.download.com/prog.exe

joe@myISP.net

Altogether

http:// cdk3.net/webexample/moon/ .html

ftp:// ftp.download.com/prog.exe

mailto:joe@myISP.net

More about http URLs

Server DNS name	path on server	args
www.cdk3.net	default	none
www.w3.org	protocol/act.html	none
www.google.com	search	q = kindberg

–

http://www.cdk3.net /default

http:// www.w3.org/protocol/act.html

http:// www.google.com/search/q = kindberg

Server pathname conventions

~ emanning is in emanning/public_html

home subdirectory

/dir1/subdir11/subdir 112 resolves to the file

/dir1/subdir11/subdir 112/ index.html

(root web page of a tree)

http

- Request-reply protocol
- advises browser of *type* of message using MIME encoding (jpg, gif, etc)
- 1 resource per reply per request
- only access control is *server challenges* (e.g. for a password)