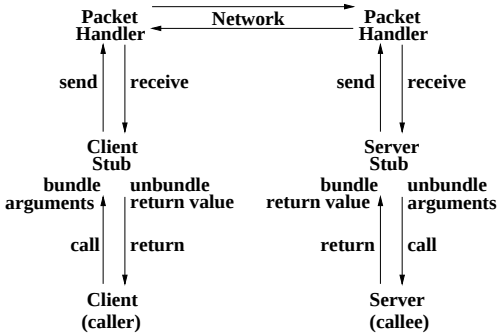
Programming Model — Last Time

Remote Procedure Call

- Servers export procedures for some set of clients to call.
- To use the server, the client does a procedure call.
- OS manages the communication.



For each procedure on which we want to support RPC:

- The RPC mechanism uses the procedure *signature* (number and type of arguments and return value)
  1. to generate a client stub that bundles up the RPC arguments and sends it off to the server, and
  2. to generate the server stub that unpacks the message, and makes the procedure call.

Distributed File Systems — Today

One of the most common uses of distributed systems.

Basic idea: we have a bunch of disks attached to different nodes and we want to be able to share disks between nodes as if all the disks were attached to every node.

Examples:

CS LAN - One server node with all the disks, and a bunch of diskless workstations on a LAN.

AppleShare - Every node is both a server with a disk and a client.

Issues:

- Naming and Transparency
- Remote file access
- Caching
- Server with state or without
- Replication

Naming and Transparency

Issues

- How are files named?
- Do file names reveal their location?
- Do file names change if the file moves?
- Do file names change if the *user* moves?

Location transparency: the name of the file does not reveal the physical storage location.

Location independence: The name of the file need not change if the file's storage location changes.

Most naming schemes used in practice do not have location independence, but many have location transparency.

Example naming strategies:

1. Absolute names: <machine name: path name>
   for example, AppleShare

   Advantages:

   Disadvantages:

Naming and Transparency

2. Mount Points (NFS - Sun's Network File System)
   - Each host has a set of local names for remote locations.
   - Each host has a mount table (/etc/fstab) or executable (/etc/fs/nsf/mount) that specifies <remote path name @ machine name> and a
     <local path name>.
   - At boot time, the local name is bound to the remote name.
   - Users then refer to the <local path name> as if it were local, and the NFS takes care of the mapping.

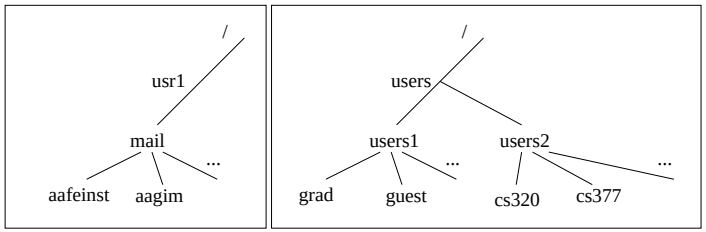
   Advantages:

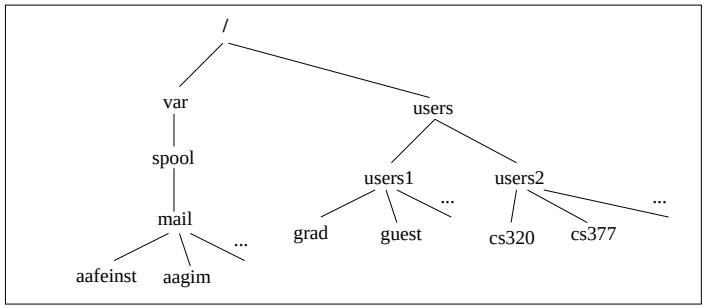
   Disadvantages:

NFS File System

Partial contents of /etc/fstab for UMass Edlab machines:

```
/usr1/mail@elux3.cs.umass.edu:/var/spool/mail
/users/users1@elsrv1:/users/users1
/users/users2@elsrv1:/users/users2
/users/users3@elsrv2:/users/users3
/users/users4@elsrv2:/users/users4
/courses/cs300@elsrv3:/courses/cs300
/rcf/mipsel/4.2/share@elsrv1:/exp/rcf/share
/rcf/common@elsrv1:/exp/rcf/common
```





Naming and Transparency

3. One single name space (CMU's Andrew and Berkeley's Sprite)
   - No matter which node you are on, the file names are the same.
   - Set of workstation clients, and a set of dedicated file server machines.
   - When a client starts up, it gets its file name structure from a server.
   - As users access files, the server sends copies to the workstation and the workstation caches the files (more on how this works later).

Advantages:

Disadvantages:

Remote File Access and Caching

Once the user specifies a remote file, the OS can do the access either

1. remotely, on the server machine and then return the results using RPC (called *remote service*), or
2. can transfer the file (or part of the file) to the requesting host, and perform local accesses (called *caching*)

Caching Issues:

- Where and when are file blocks cached?
- When are modifications propagated back to the remote file?
- What happens if multiple clients cache the same file?

Remote File Access and Caching

Location

1. Local disk

   Advantages:

   Disadvantages:

2. Local memory

   Advantages:

   Disadvantages:

Cache Update Policies

When to write local changes to the server has a central role in determining distributed file system performance.

Write through: yields the most reliable results since every write hits the remote disk before the process continues, but it has the poorest performance.

- Caching with write through is equivalent to using remote service for all writes, and exploits caching only for reads.

Write back: yields the quickest response time since the write need only hit cache before the process continues.

- It reduces network traffic and the number of writes to the disk when a process repeatedly writes to the same disk block, since only one of the writes will go across the network.
- If a user machine crashes, the unwritten data is lost.

Cache Update Policies

Write back variations: differ on *when* to flush the modified data back to server. From the least to the most frequent writes:

1. Write when a modified file is closed (Andrew).
   - Useful for files that are open and used for a long time.
   - Makes process wait on file close.
   - If a file is open and closed for only a short period of time, this won't help.
   - Problem if multiple clients for the same file since each might have a different local copy.
2. Write when a modified block is evicted from memory.
   - With large memories this might not happen frequently enough to achieve reasonable reliability.
3. Write modified files every 30 seconds (NFS & Sprite).
   - Failures can only lead to files that are 30 seconds out of date.

Summary

- Naming
  - Desire name independence, but it is difficult to attain
  - Location dependent names are most prevalent
- Speed up remote file access with caching
- Need to write changes back to disk
- Consistancy (more next time)