Distributed File Systems

Special Problems:

- *Naming:* identify files systemwide
- how are concurrent reads and writes executed?

Naming
Aims:

- *Location Transparency:*
  name does not give any hint on location
- *Location independence:*
  files can be moved without name being changed

standard approach: *Remote mounting*
Make remote file system available under local name
Achieves only location transparency
Latter difficult to achieve (requires name server)

Concurrent Read and Writes
no problem if one central server: can enforce order of operations
With cache, get consistency problems
One possible solution: adopt *session semantics*
Changes visible only after `close`-operation
Problem: File pointers not usable

Implementation Issues

Main issue: stateless vs. stateful servers
(Should server keep information about requests?)
Properties of stateless servers

- Fault tolerance
- No `open`/`close`-requests needed
- No problems with client crashes

Advantage of stateful servers

- Read ahead possible
- Idempotency easier
- File locking possible

Main problem: Cache consistency, especially for stateless servers

NFS

Idea: make file systems available on other hosts
Works on different architectures
⇒ Need well-defined protocol
RPC's used for this purpose

Stateless system
⇒ no `open`/`close` RPC's
Each RPC contains absolute address in file
Caching employed:

- Server does normal caching (no ill-effects)
- Client caches reads and writes
  ⇒ obtain inconsistency
  Data sent to server only when
  - $>$ 8k written
  - file closed on client
  - timeout reached
  `open` on client checks server