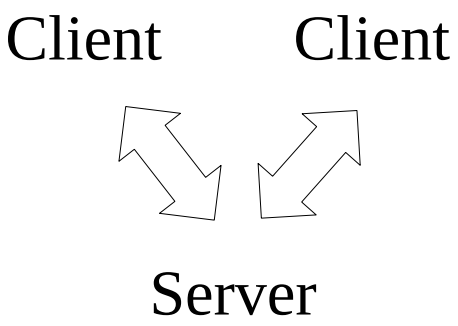
ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
UNIVERSITY OF CRETE

HY590.45
Modern Topics in Scalable Storage Systems

Kostas Magoutis
magoutis@csd.uoc.gr
http://www.csd.uoc.gr/~hy590-45

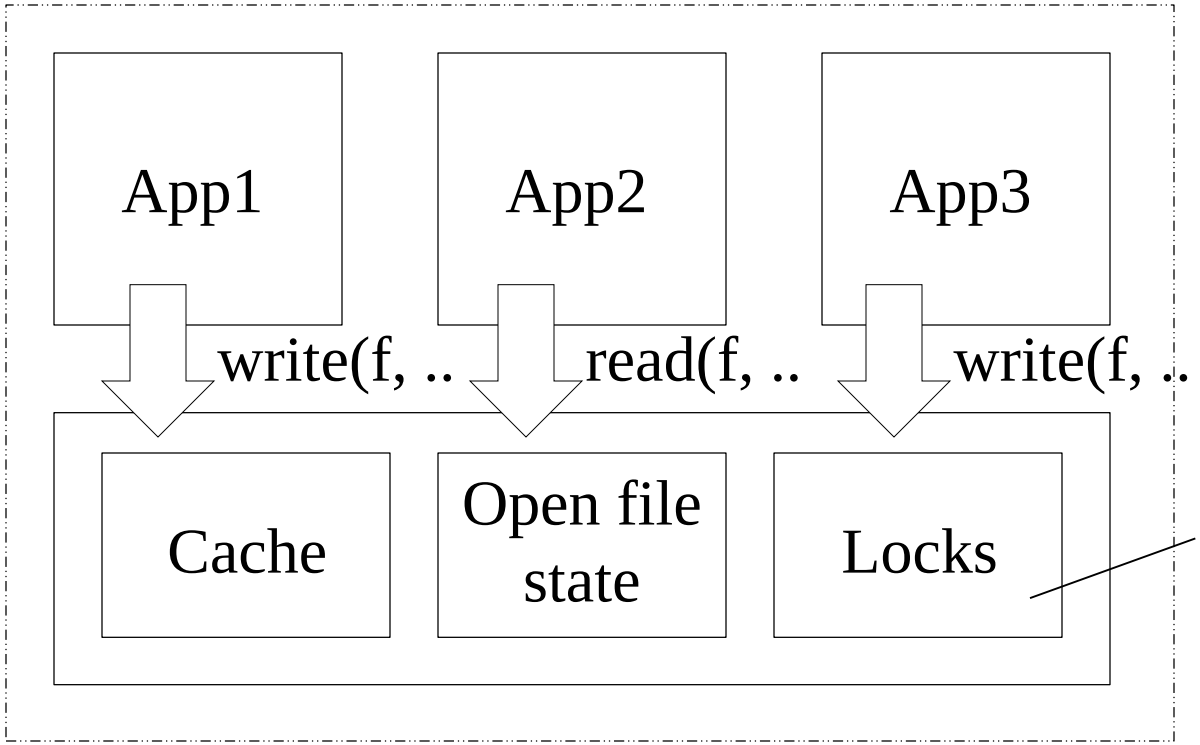
Distributed file sharing

- Benefits
  - Ability to access files from many locations
    - E.g., home directories
  - Consolidate storage management



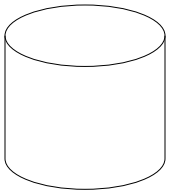
- Makes it possible to share files
  - Often concurrent readers or single writer
  - Less often, concurrent writers
    - Exclusive access to non-overlapping parts of file
    - Several data producers, concurrent append to shared file
    - Infrequent in engineering/office type workloads

File sharing in a single system



- File lock
- Range lock
  - Byte range
- Type of lock (op)
  - R/W

Single system



Data + metadata

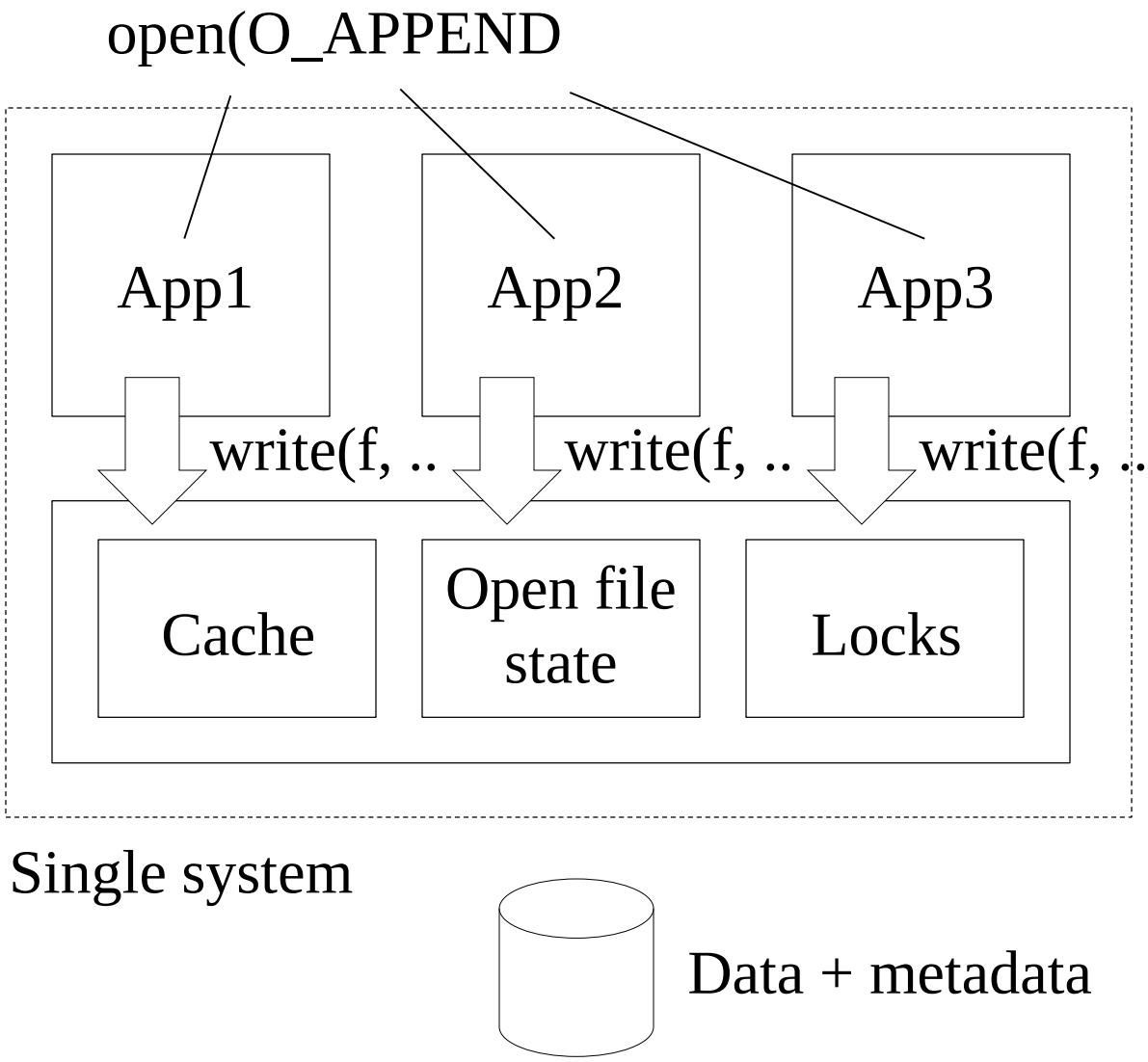
Crash recovery?

File-access APIs and semantics

Concurrent append, implicit serialization

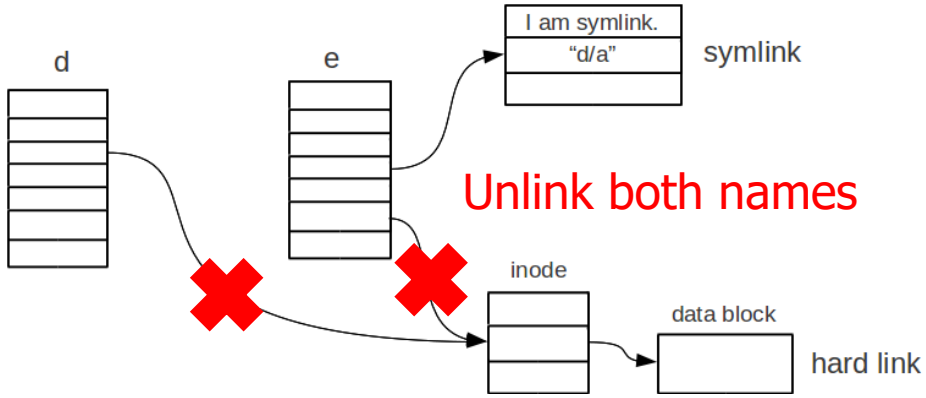
P1	P2
fd=open(f, O_APPEND	fd=open(f, O_APPEND
write(fd, …	write(fd, …
write(fd, …	write(fd, …
write(fd, …	write(fd, …
write(fd, …	write(fd, …
…	…
close(fd, …	close(fd, …

Concurrent append in a single system



File-access APIs and semantics

- Hard links: multiple names can be linked to an inode

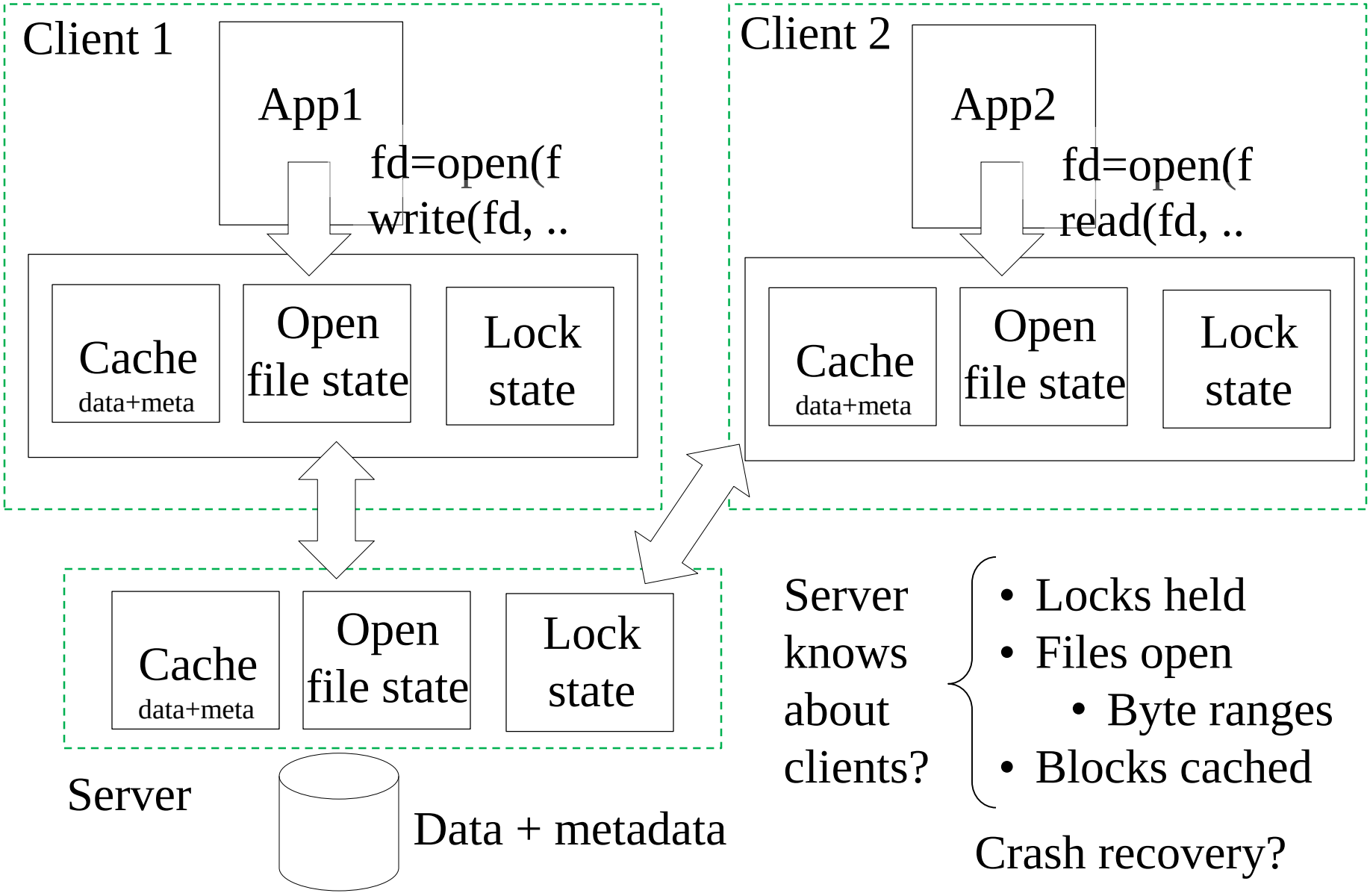


- In Unix, when files are unlinked they are not removed unless all open references to them are closed

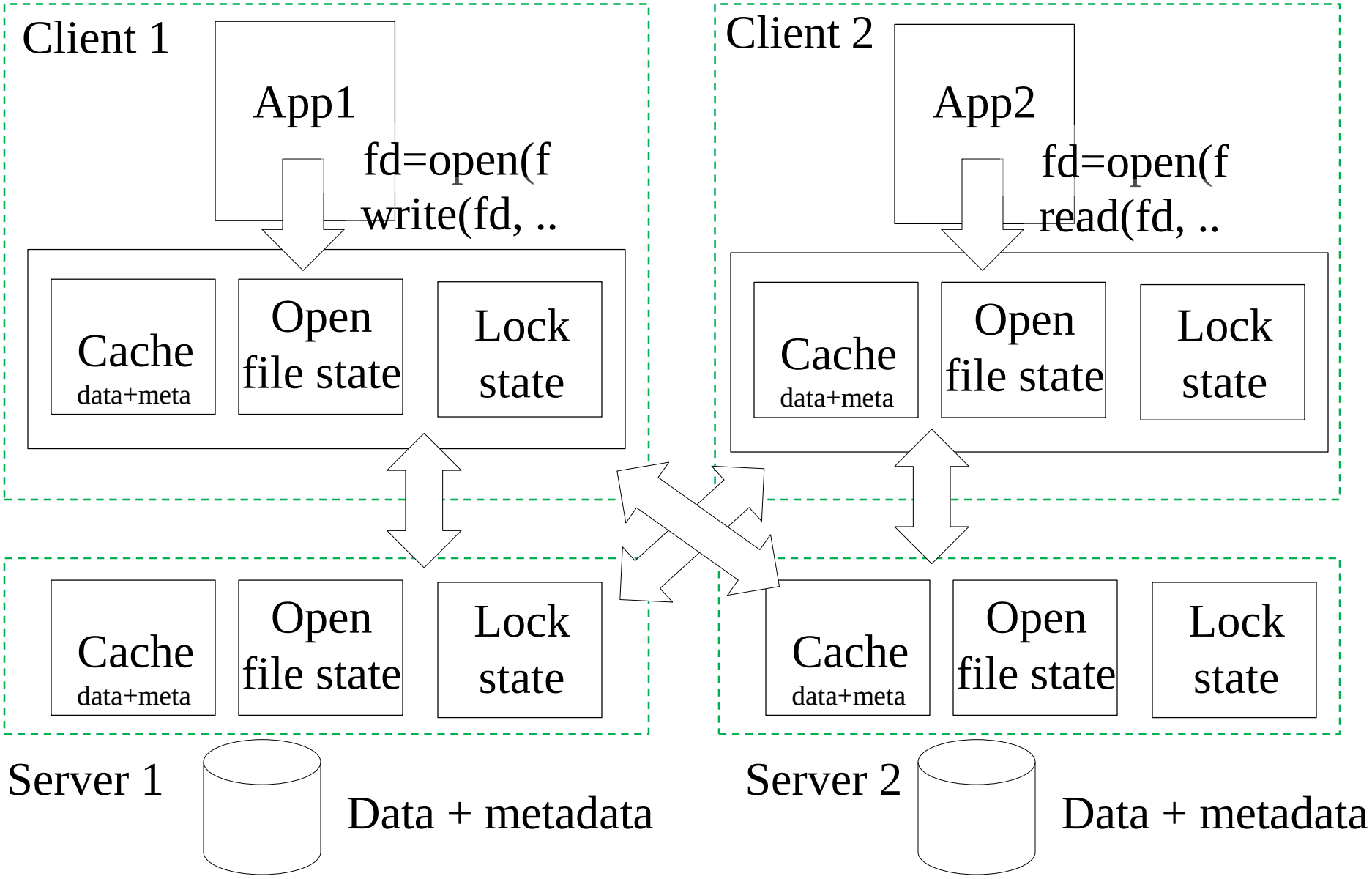
- File system semantics imply state

Image © https://medium.com/@meghamohan/hard-link-and-symbolic-link-3cad74e5b5dc

Extend to a distributed setting



Extend to a distributed setting

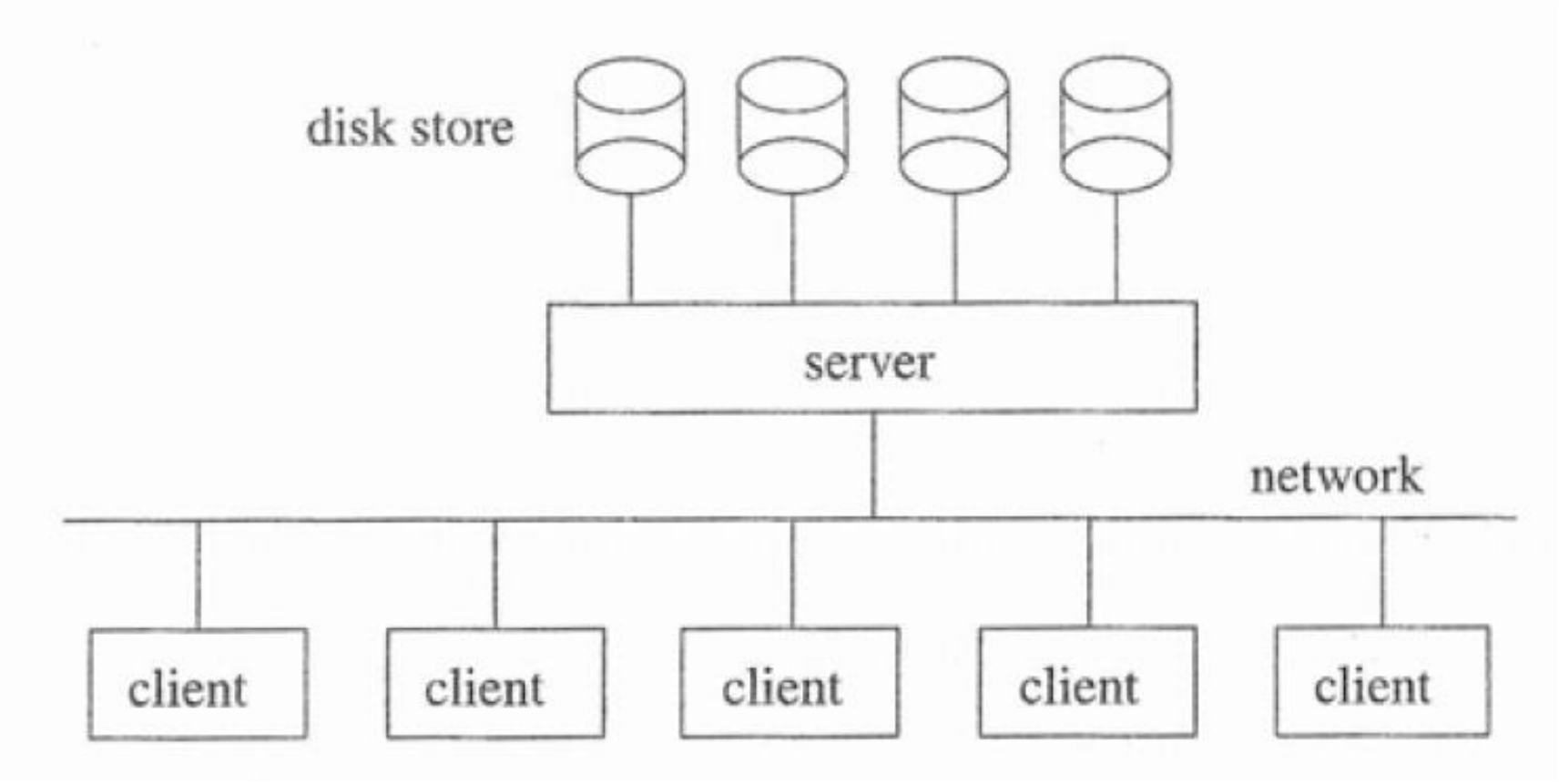


Network File System (NFS)

- History
  - UNIX United
  - SUN Network Disk
  - RFS
  - Andrew File System (AFS)

- Overview of NFS
  - Stateless
  - Aims to offer UNIX semantics
  - Transport independent
  - UNIX security and access control
  - Client caching and consistency

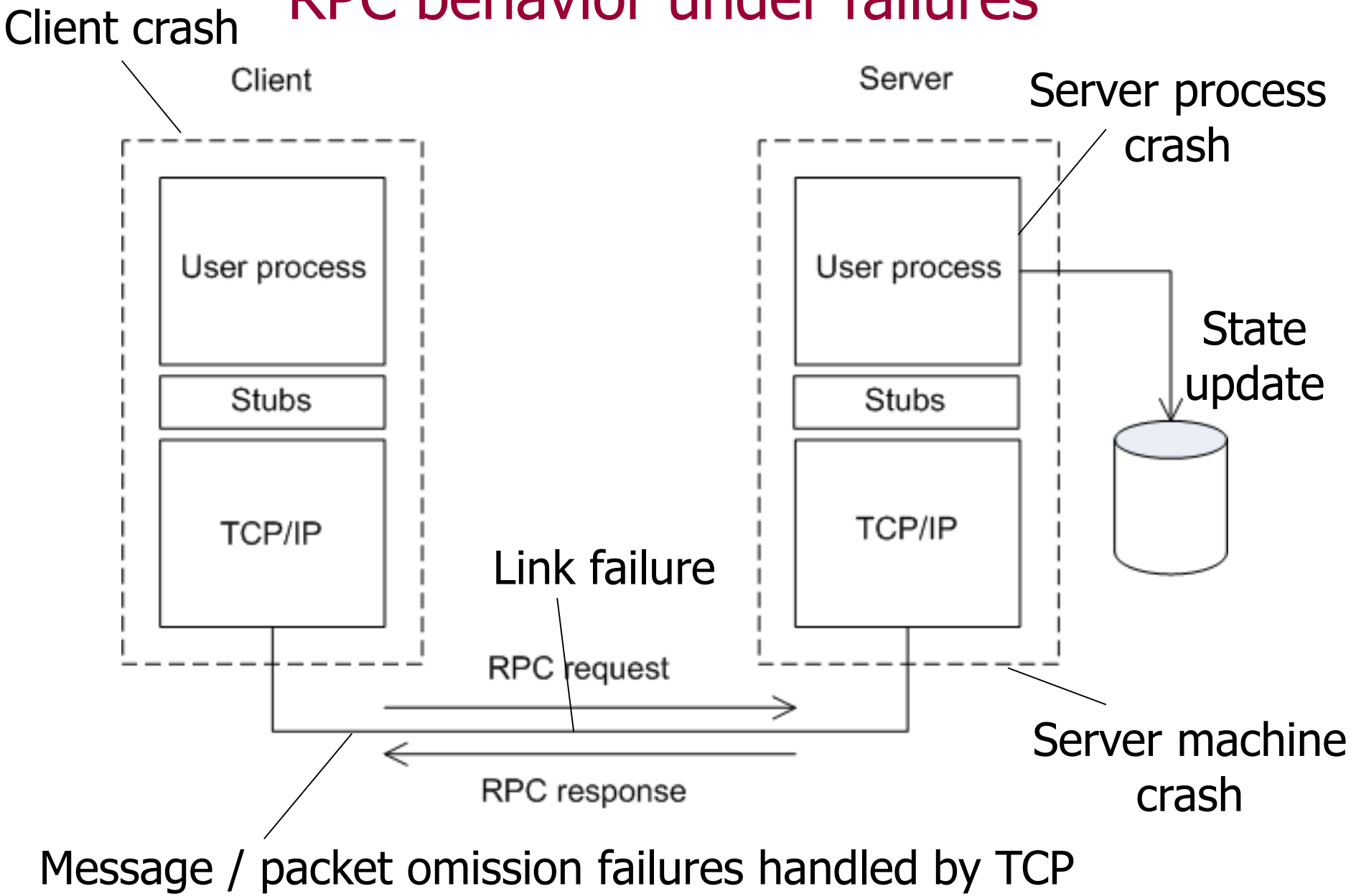
NFS division between clients and server



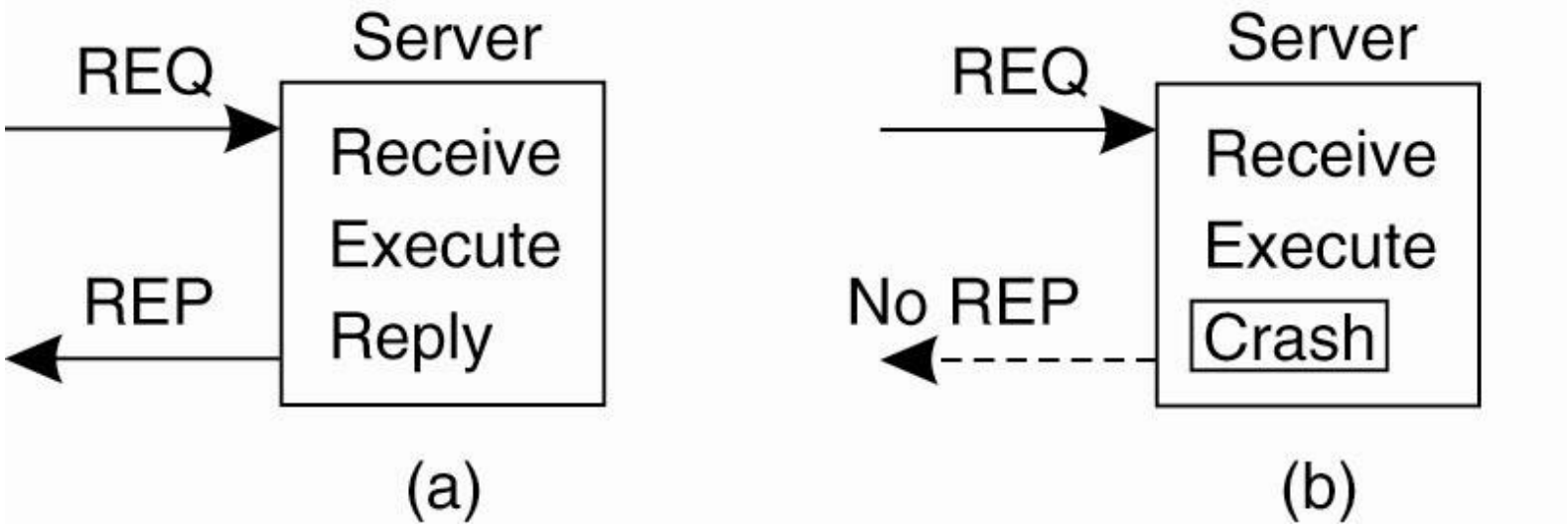
NFS structure and operation

- Based on Remote Procedure Calls (RPCs)
  - Handle problems that may occur due to crashes
- Files identified by NFS handle
  - Comprises inode id, file system id, generation number
- VFS/Vnode layer

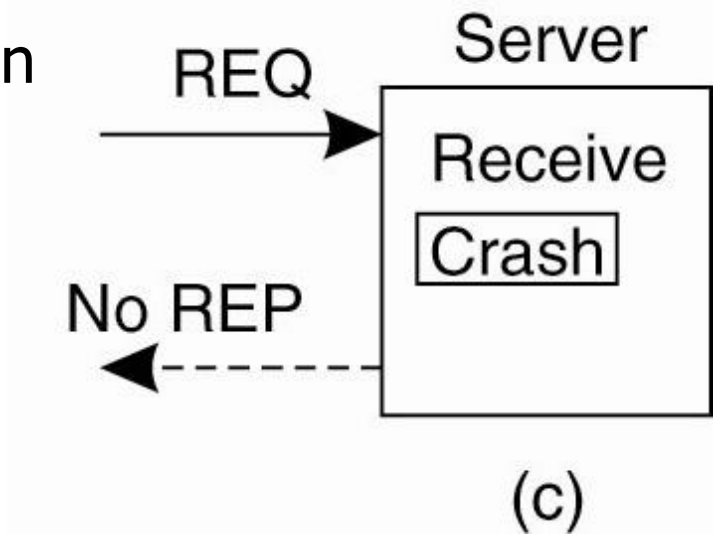
RPC behavior under failures



Server crashes



A server in client-server communication
(a) The normal case
(b) Crash after execution
(c) Crash before execution



RPC semantics

- At-least-once
  - Retry after an exception/timeout until successful
  - Good choice with idempotent operations (e.g., reads)
  - How about non-idempotent operations (e.g., writes)?

- At-most-once
  - Do not retry an operation or try to avoid duplicates