DISTRIBUTED SYSTEMS
Principles and Paradigms
Second Edition
ANDREW S. TANENBAUM
MAARTEN VAN STEEN

Chapter 11
DISTRIBUTED FILE SYSTEMS

Client-Server Architectures (1)

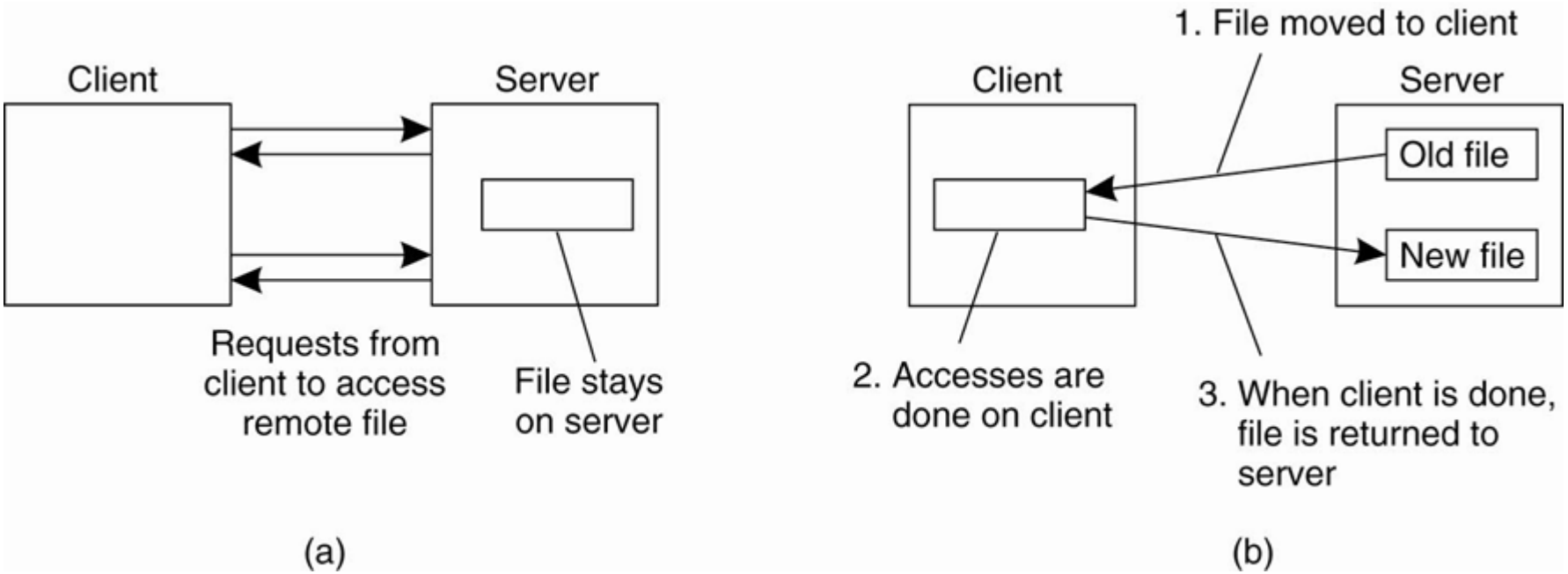


Figure 11-1. (a) The remote access model. (b) The upload/download model.

Client-Server Architectures (2)

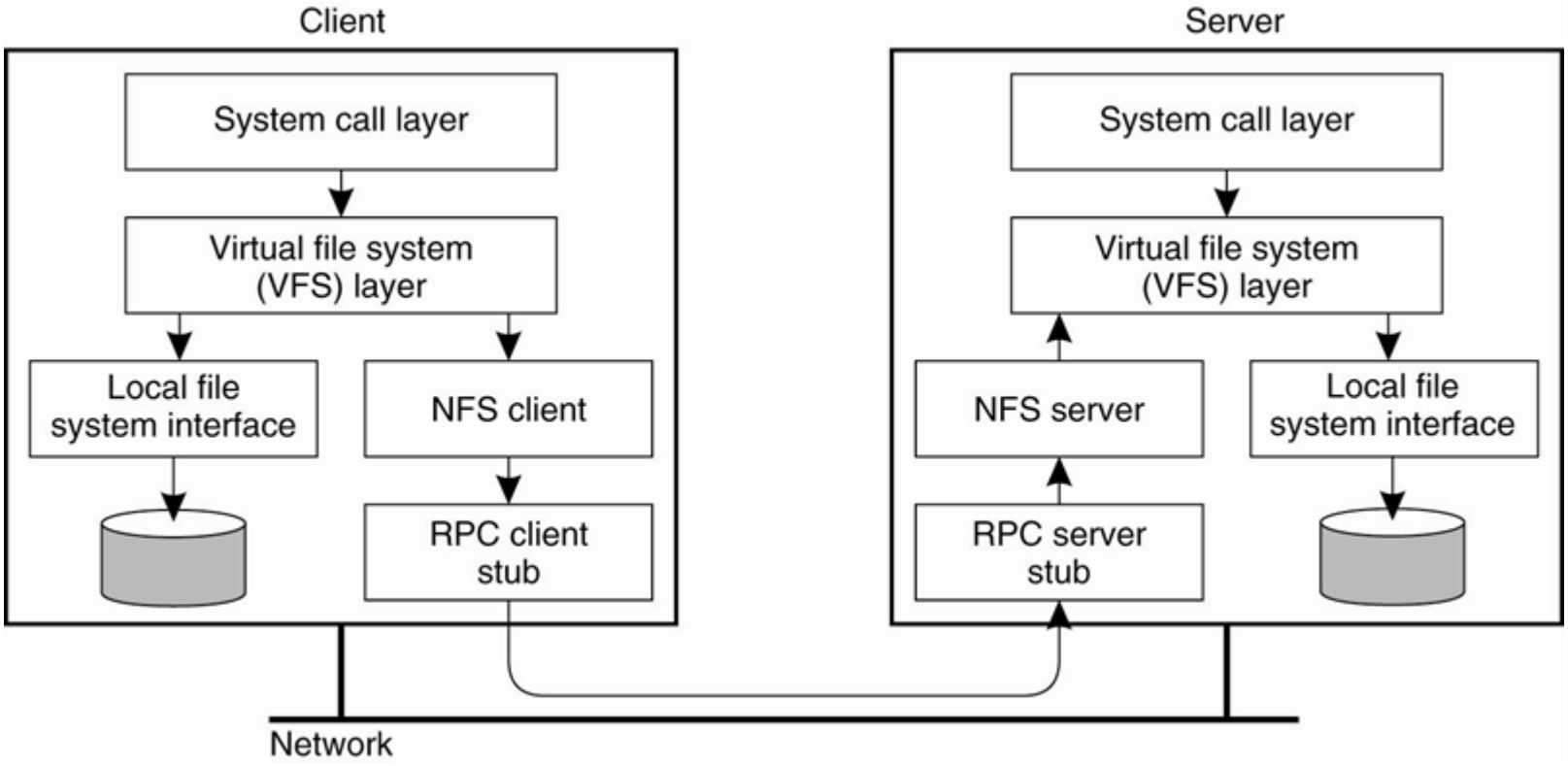


Figure 11-2. The basic NFS architecture for UNIX systems.

File System Model (1)

Operation	v3	v4	Description
Create	Yes	No	Create a regular file
Create	No	Yes	Create a nonregular file
Link	Yes	Yes	Create a hard link to a file
Symlink	Yes	No	Create a symbolic link to a file
Mkdir	Yes	No	Create a subdirectory in a given directory
Mknod	Yes	No	Create a special file
Rename	Yes	Yes	Change the name of a file
Remove	Yes	Yes	Remove a file from a file system
Rmdir	Yes	No	Remove an empty subdirectory from a directory

Figure 11-3. An incomplete list of file system operations supported by NFS.

File System Model (2)

Operation	v3	v4	Description
Open	No	Yes	Open a file
Close	No	Yes	Close a file
Lookup	Yes	Yes	Look up a file by means of a file name
Readdir	Yes	Yes	Read the entries in a directory
Readlink	Yes	Yes	Read the path name stored in a symbolic link
Getattr	Yes	Yes	Get the attribute values for a file
Setattr	Yes	Yes	Set one or more attribute values for a file
Read	Yes	Yes	Read the data contained in a file
Write	Yes	Yes	Write data to a file

Figure 11-3. An incomplete list of file system operations supported by NFS.

Cluster-Based Distributed File Systems (1)

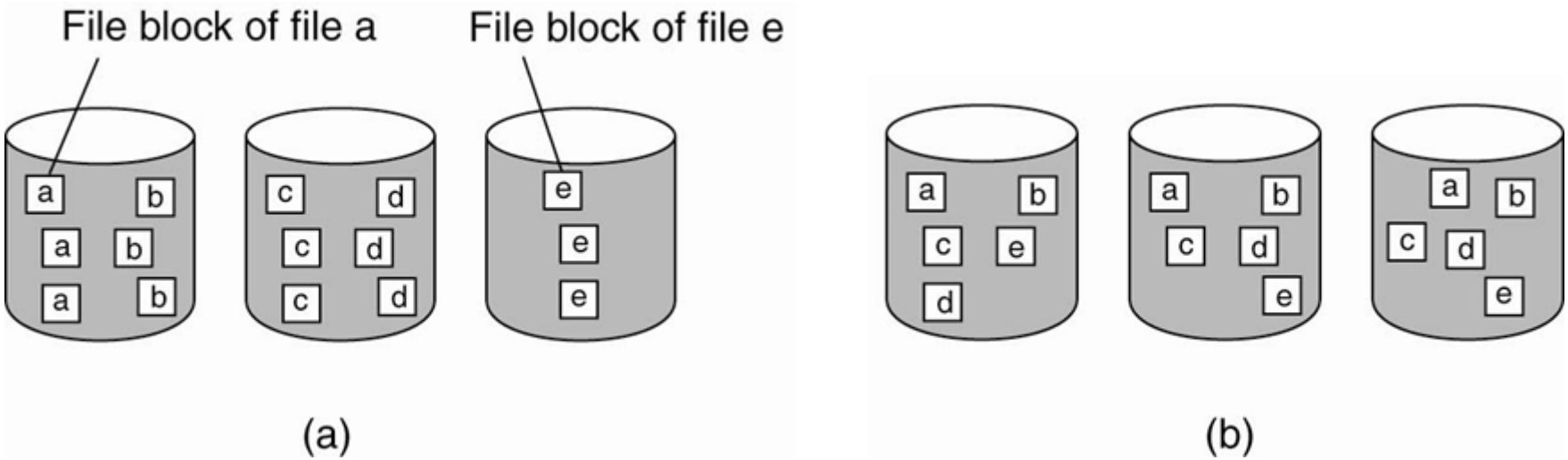


Figure 11-4. The difference between (a) distributing whole files across several servers and (b) striping files for parallel access.

Cluster-Based Distributed File Systems (2)

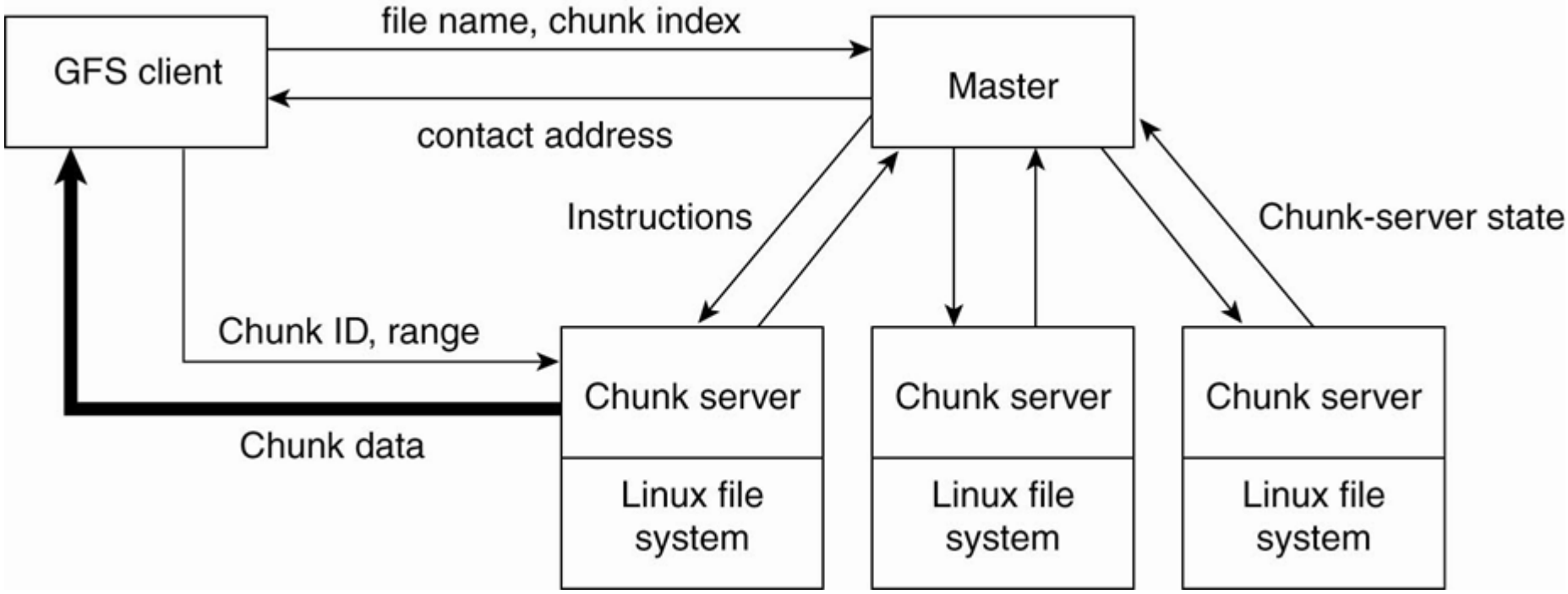


Figure 11-5. The organization of a Google cluster of servers.

Symmetric Architectures

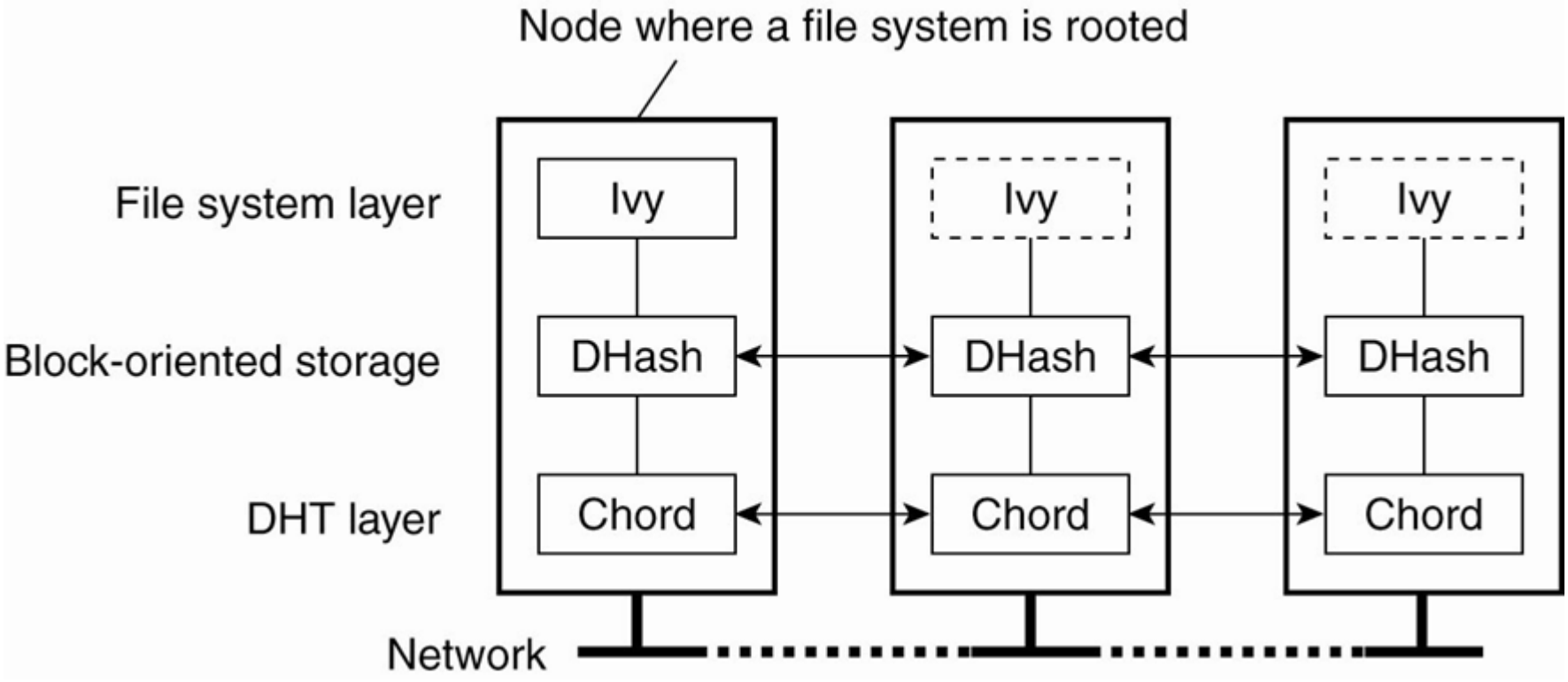


Figure 11-6. The organization of the Ivy distributed file system.

Remote Procedure Calls in NFS

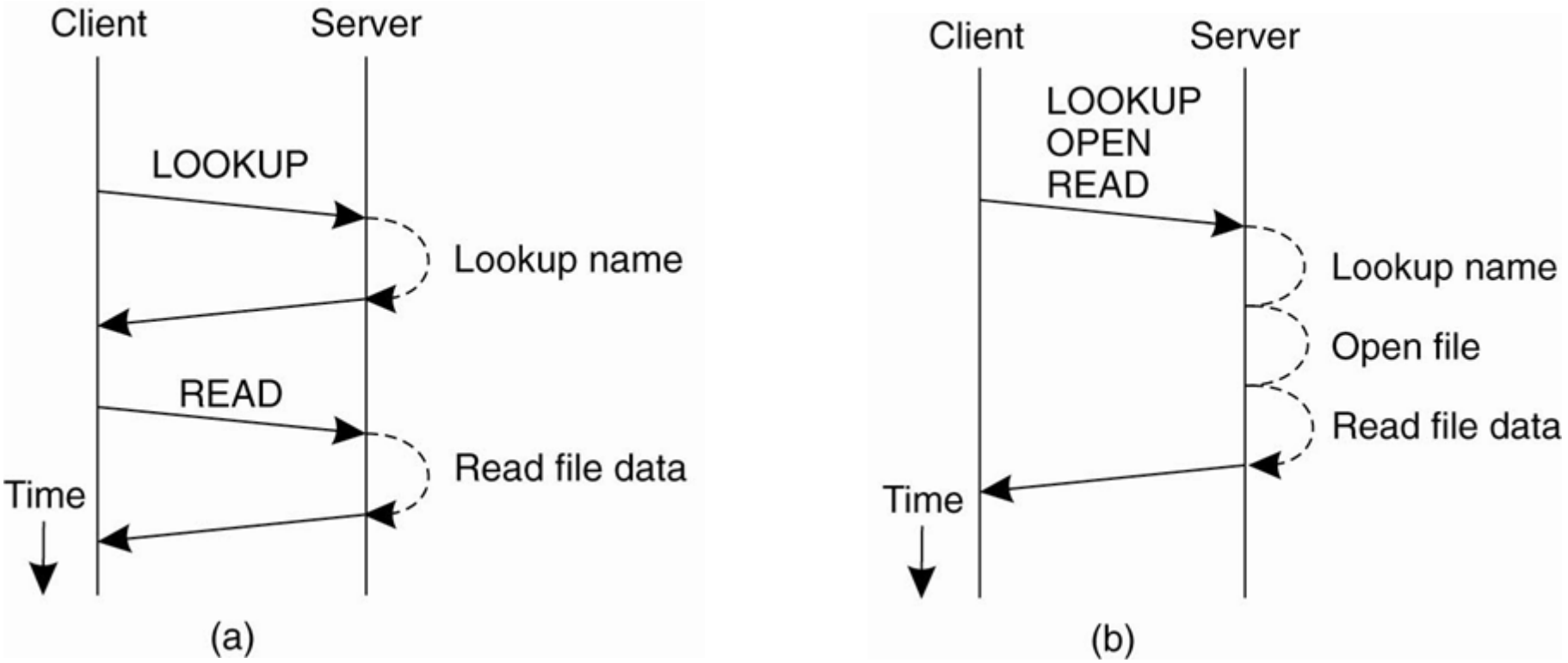


Figure 11-7. (a) Reading data from a file in NFS version 3. (b) Reading data using a compound procedure in version 4.

The RPC2 Subsystem (1)

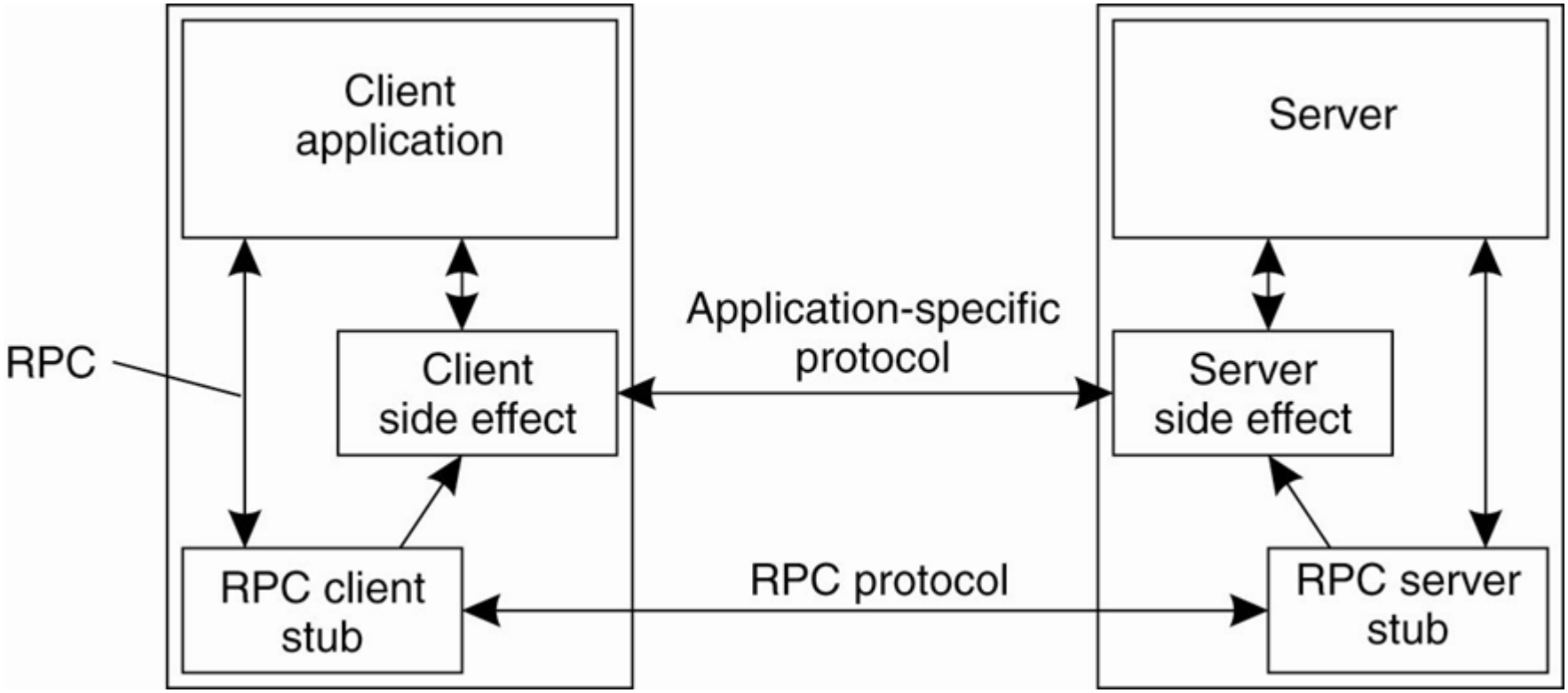


Figure 11-8. Side effects in Coda's RPC2 system.

The RPC2 Subsystem (2)

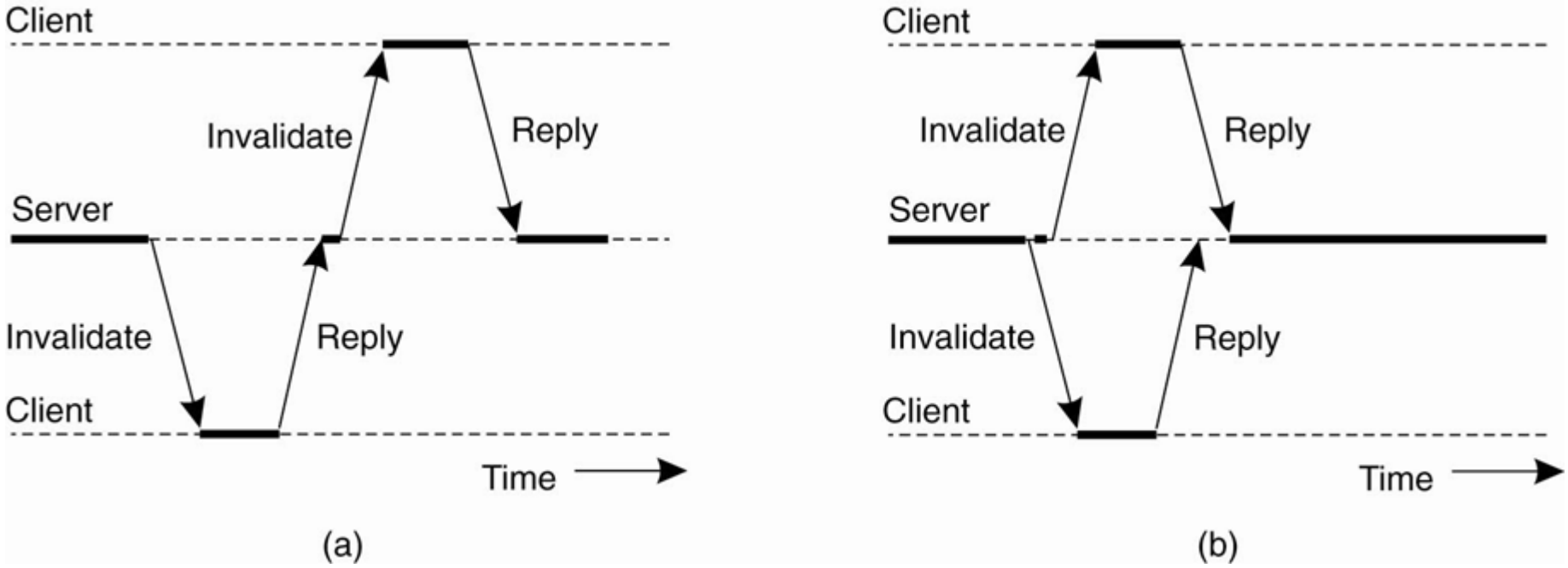


Figure 11-9. (a) Sending an invalidation message one at a time. (b) Sending invalidation messages in parallel.

File-Oriented Communication in Plan 9

File	Description
ctl	Used to write protocol-specific control commands
data	Used to read and write data
listen	Used to accept incoming connection setup requests
local	Provides information on the caller's side of the connection
remote	Provides information on the other side of the connection
status	Provides diagnostic information on the current status of the connection

Figure 11-10. Files associated with a single TCP connection in Plan 9.

Naming in NFS (1)

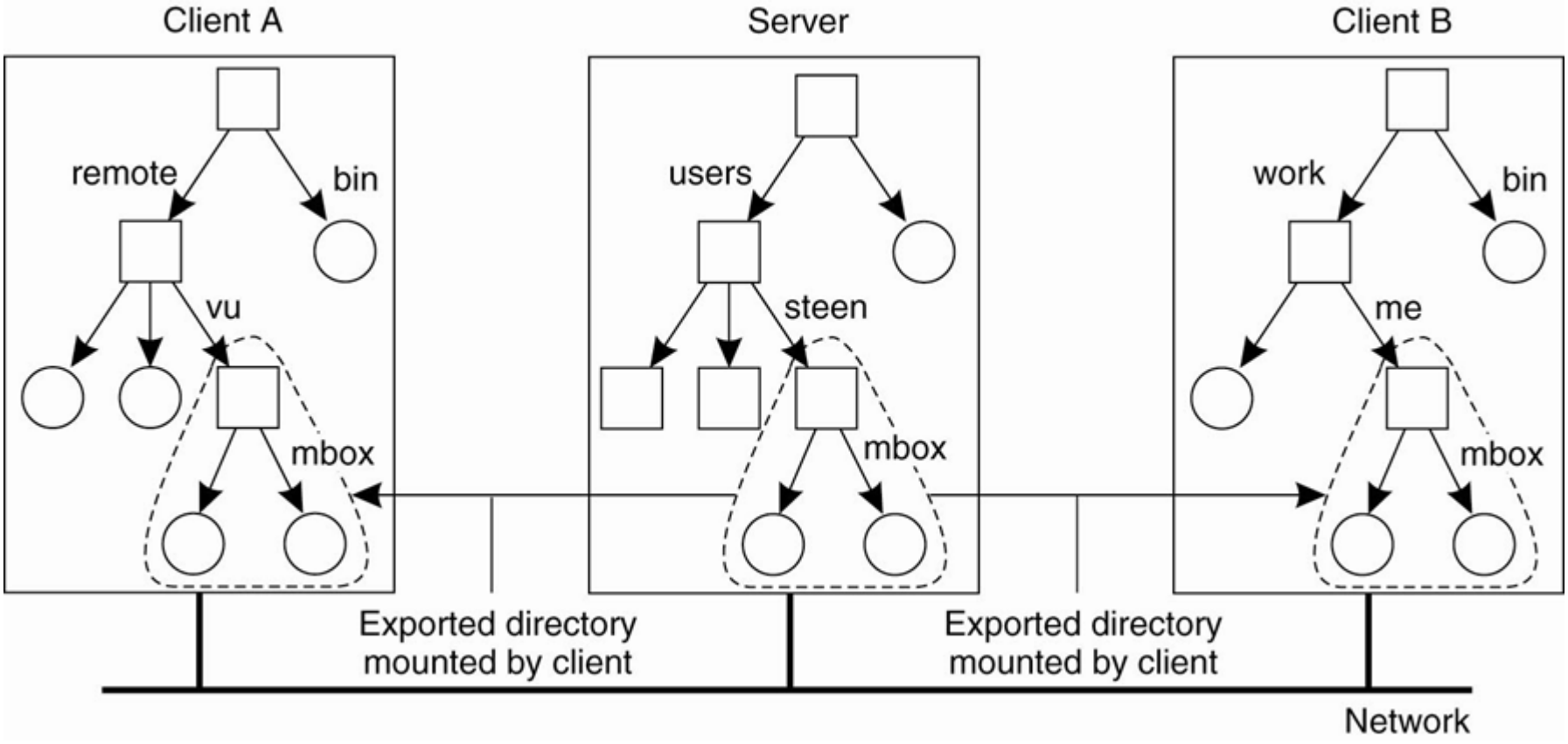


Figure 11-11. Mounting (part of) a remote file system in NFS.

Naming in NFS (2)

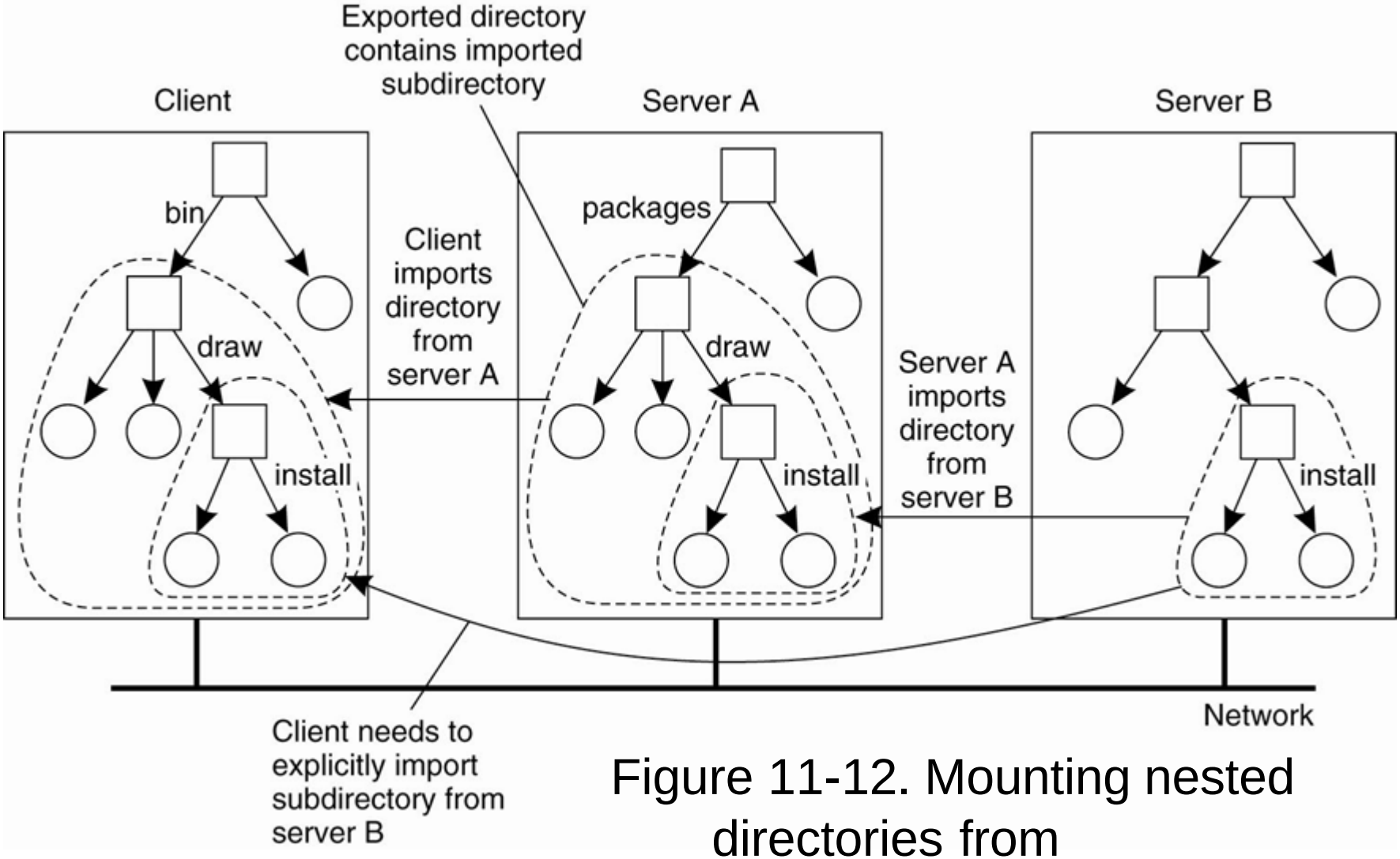


Figure 11-12. Mounting nested directories from multiple servers in NFS.

Automounting (1)

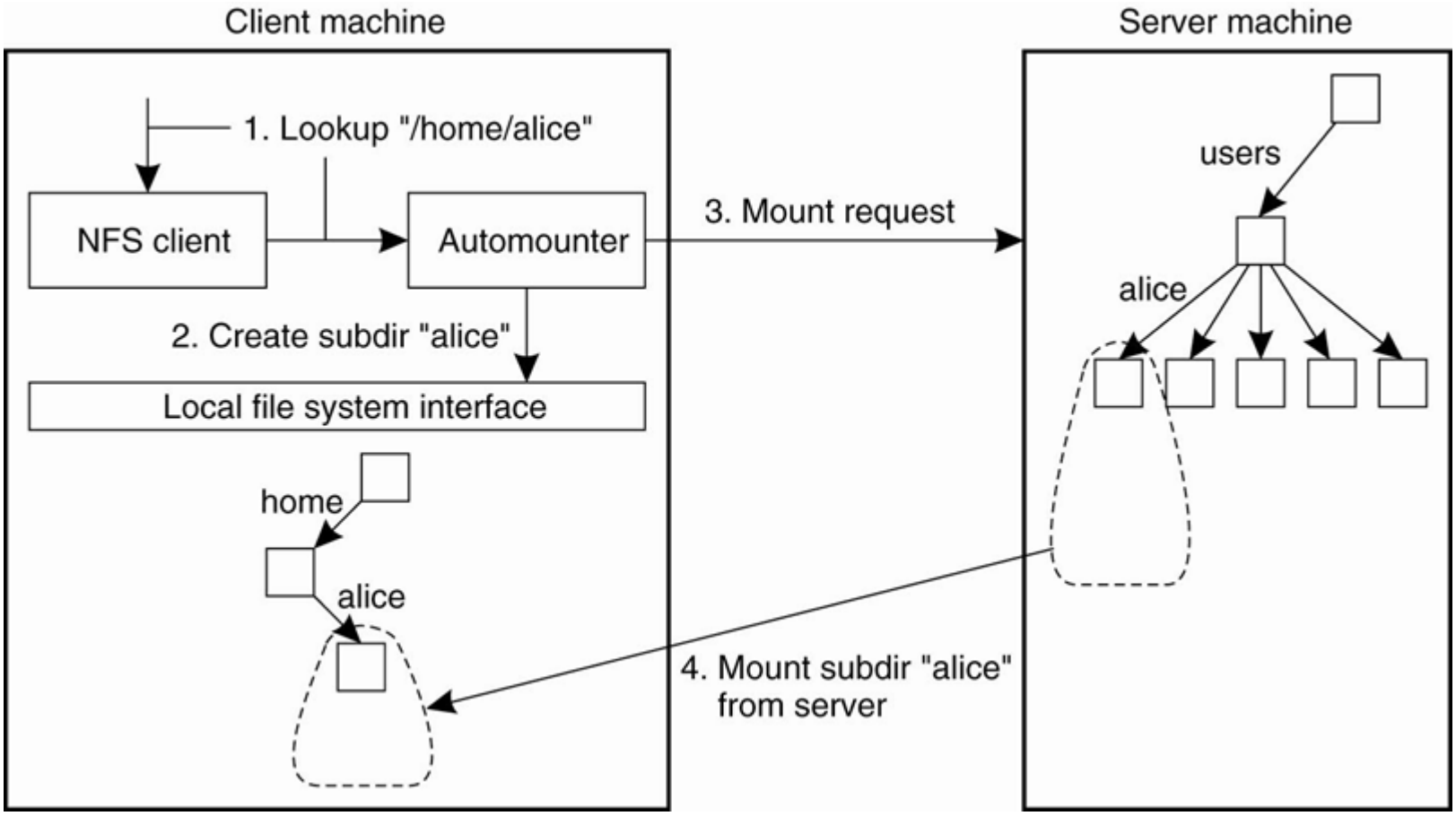


Figure 11-13. A simple automounter for NFS.

Automounting (2)

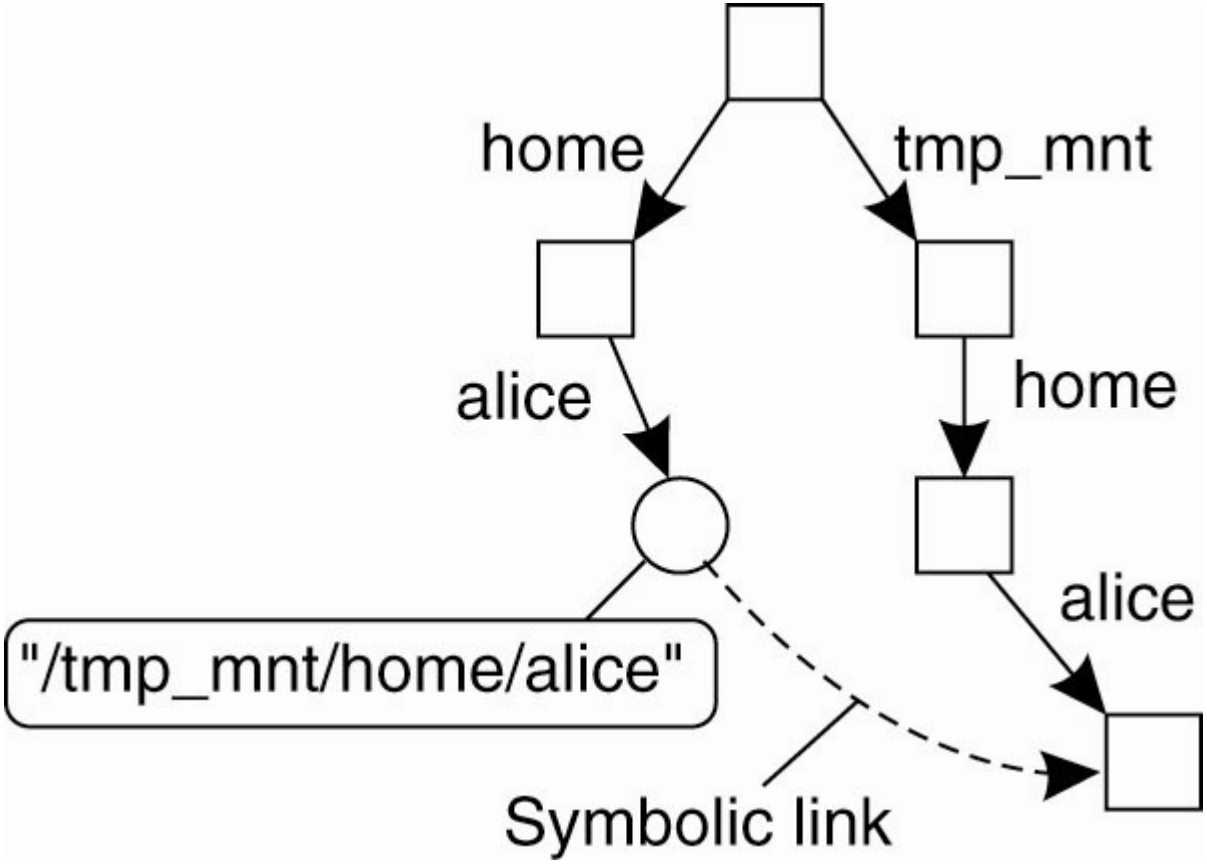


Figure 11-14. Using symbolic links with automounting.

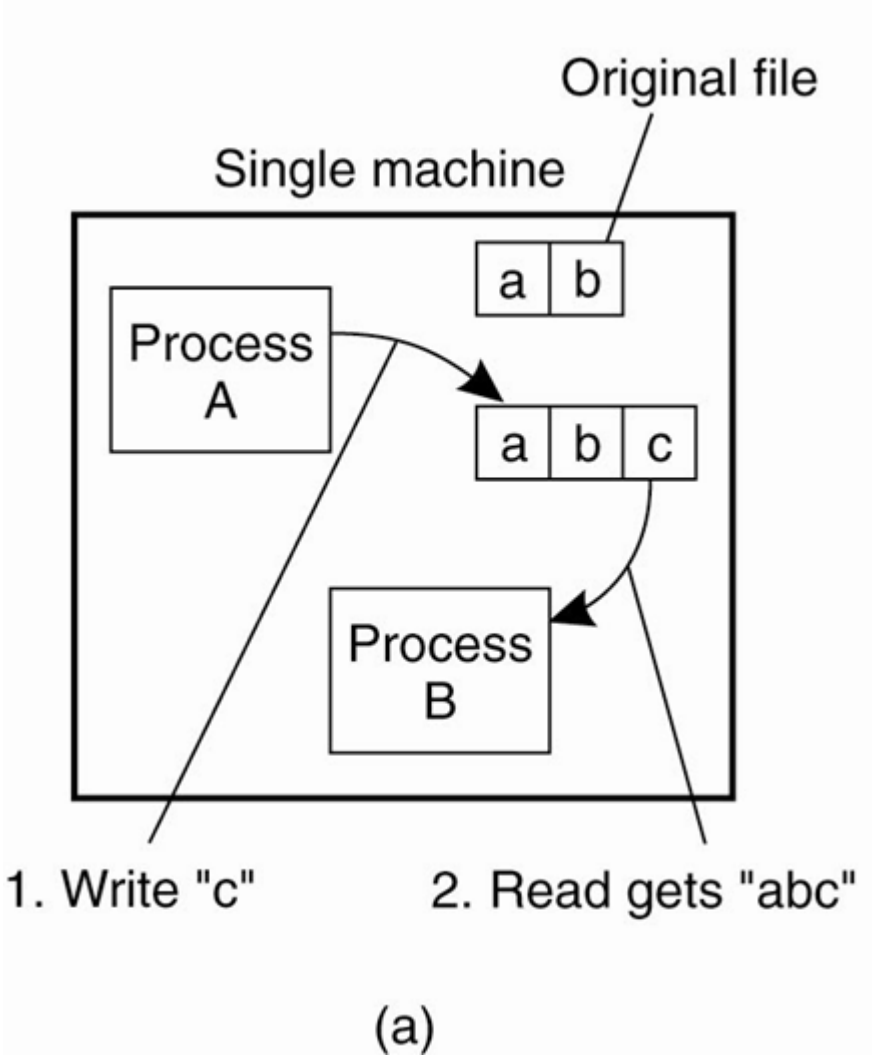
Constructing a Global Name Space

Junction	Description
GNS junction	Refers to another GNS instance
Logical file-system name	Reference to subtree to be looked up in a location service
Logical file name	Reference to a file to be looked up in a location service
Physical file-system name	Reference to directly remote-accessible subtree
Physical file name	Reference to directly remote-accessible file

Figure 11-15. Junctions in GNS.

Semantics of File Sharing (1)

Figure 11-16. (a) On a single processor, when a read follows a write, the value returned by the read is the value just written.



Semantics of File Sharing (2)

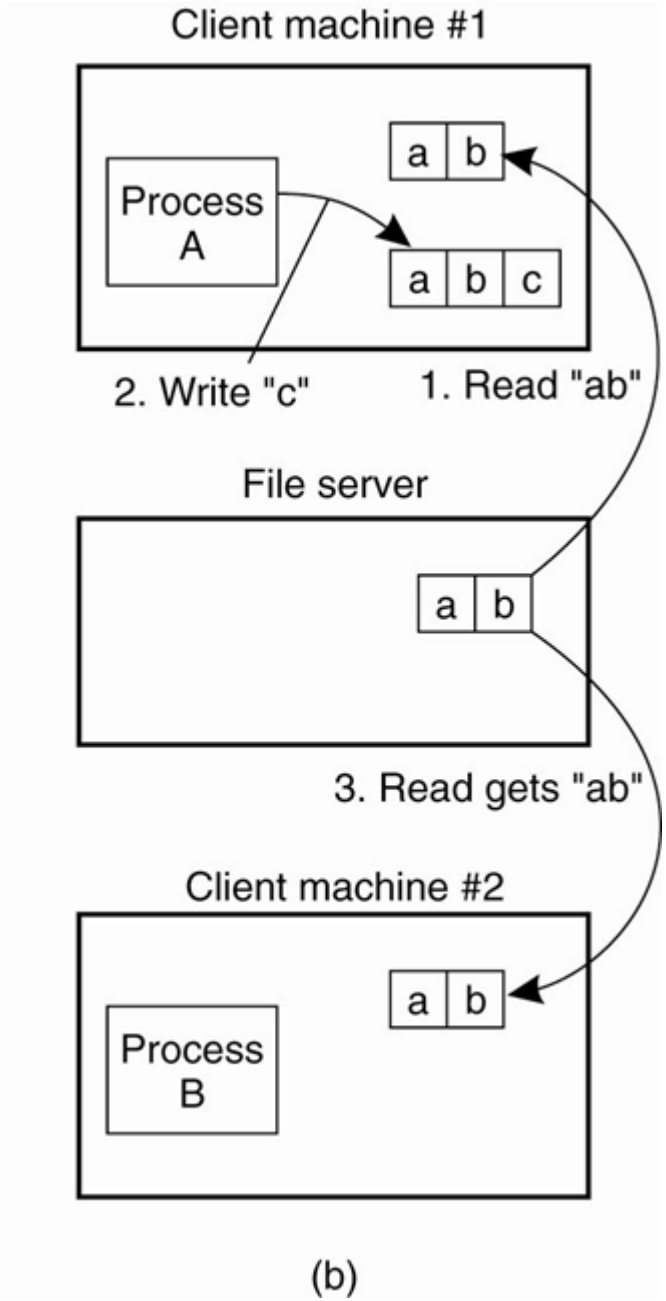


Figure 11-16. (b) In a distributed system with caching, obsolete values may be returned.

Semantics of File Sharing (3)

Method	Comment
UNIX semantics	Every operation on a file is instantly visible to all processes
Session semantics	No changes are visible to other processes until the file is closed
Immutable files	No updates are possible; simplifies sharing and replication
Transactions	All changes occur atomically

Figure 11-17. Four ways of dealing with the shared files in a distributed system.

File Locking (1)

Operation	Description
Lock	Create a lock for a range of bytes
Lockt	Test whether a conflicting lock has been granted
Locku	Remove a lock from a range of bytes
Renew	Renew the lease on a specified lock

Figure 11-18. NFSv4 operations related to file locking.

File Locking (2)

Request access	Current file denial state: NONE	READ	WRITE	BOTH
READ	Succeed	Fail	Succeed	Fail
WRITE	Succeed	Succeed	Fail	Fail
BOTH	Succeed	Fail	Fail	Fail

(a)

Figure 11-19. The result of an open operation with share reservations in NFS. (a) When the client requests shared access given the current denial state.

File Locking (3)

Current access state	Requested file denial state: NONE	READ	WRITE	BOTH
READ	Succeed	Fail	Succeed	Fail
WRITE	Succeed	Succeed	Fail	Fail
BOTH	Succeed	Fail	Fail	Fail

(b)

Figure 11-19. The result of an open operation with share reservations in NFS. (b) When the client requests a denial state given the current file access state.

Sharing Files in Coda

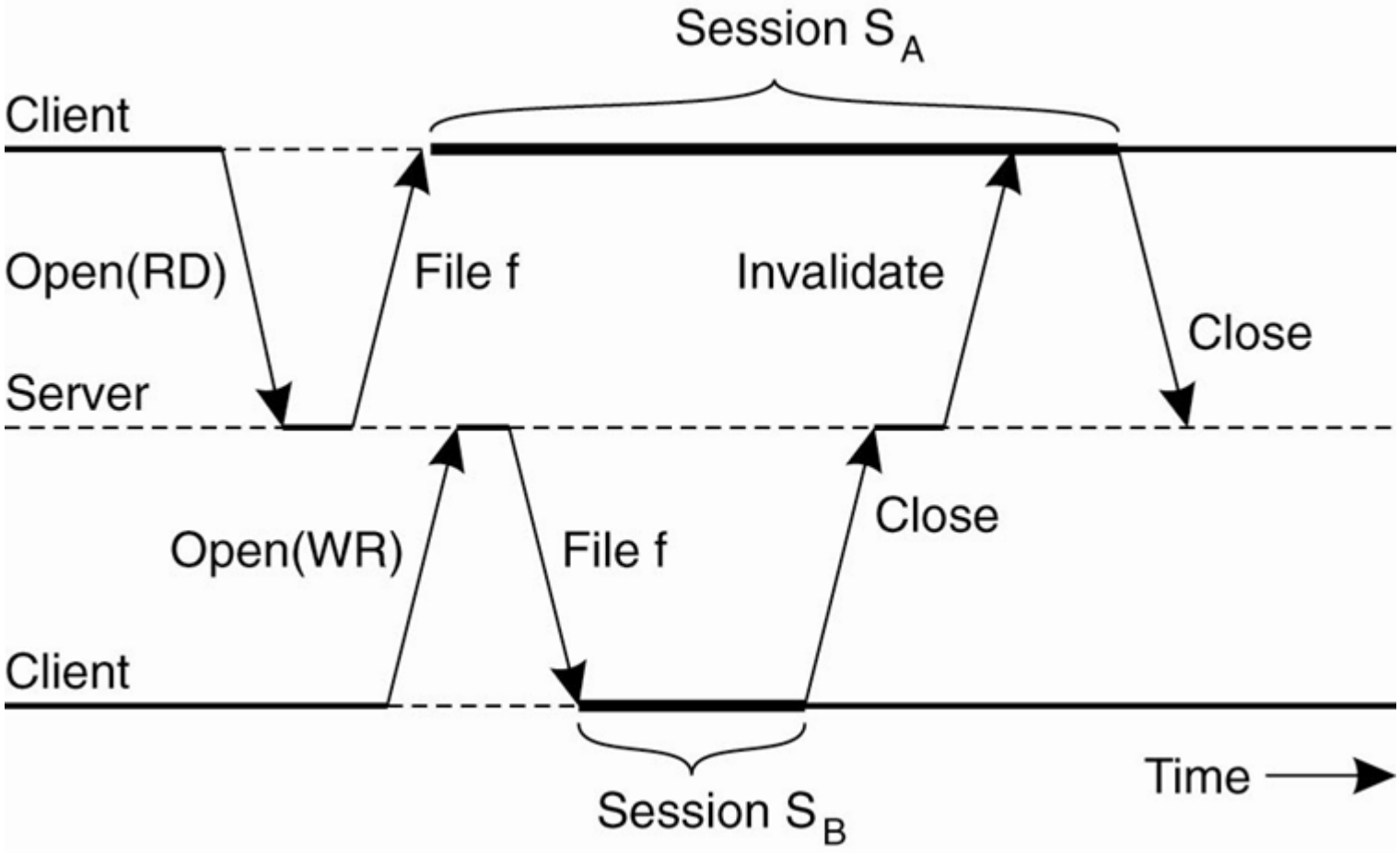


Figure 11-20. The transactional behavior in sharing files in Coda.

Client-Side Caching (1)

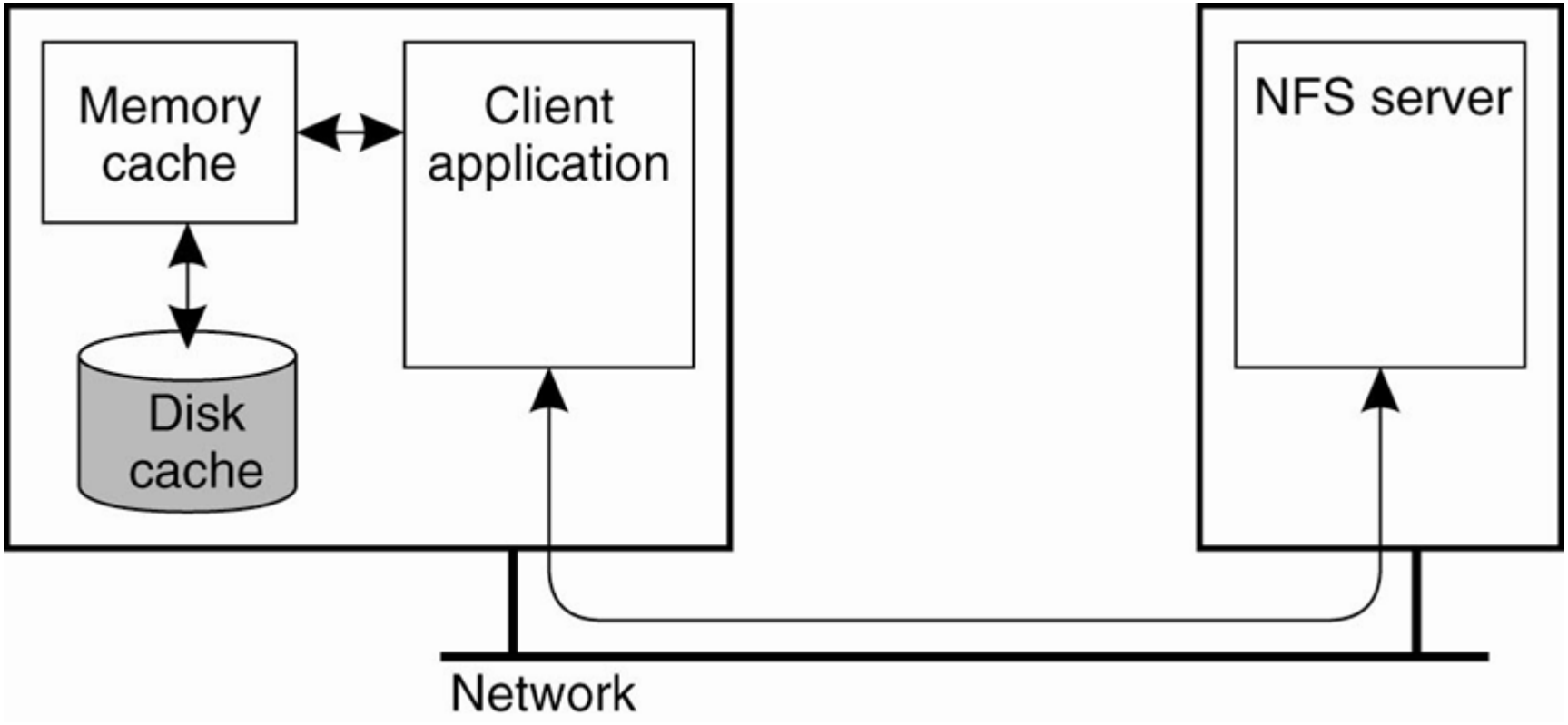


Figure 11-21. Client-side caching in NFS.

Client-Side Caching (2)

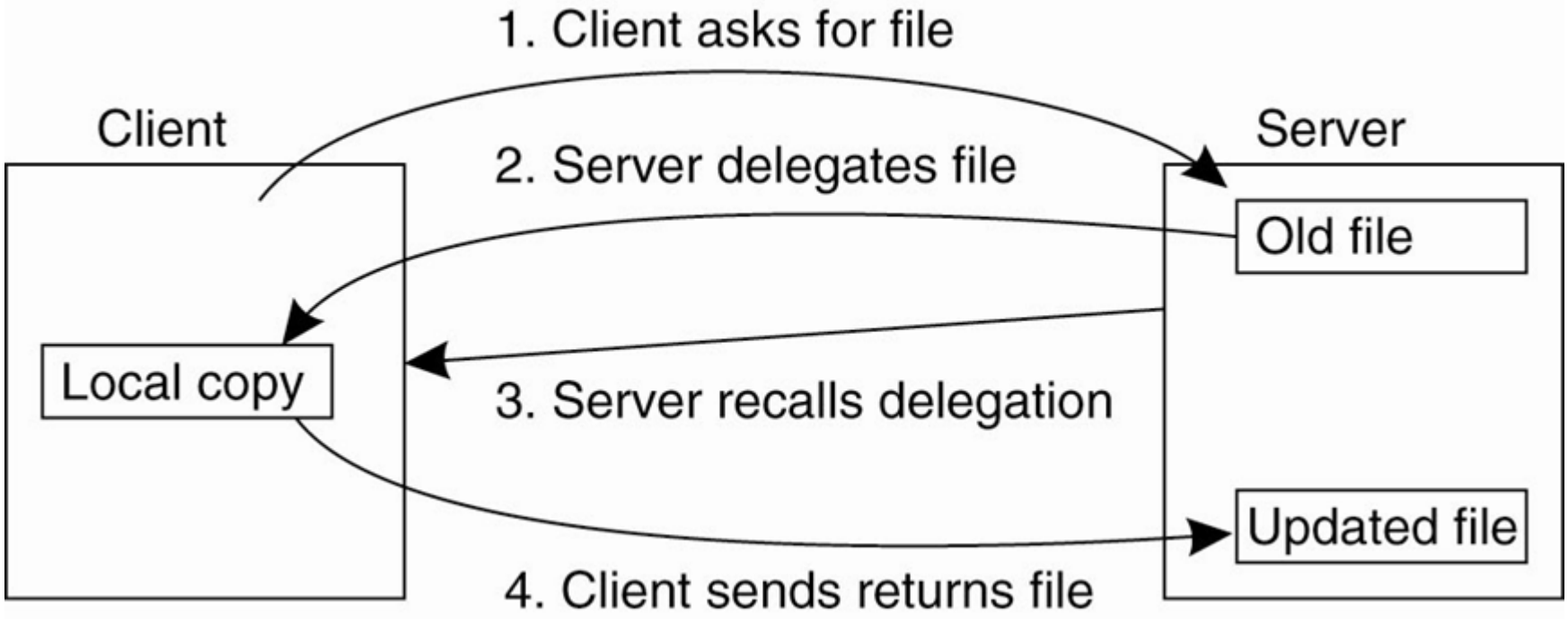


Figure 11-22. Using the NFSv4 callback mechanism to recall file delegation.

Client-Side Caching in Coda

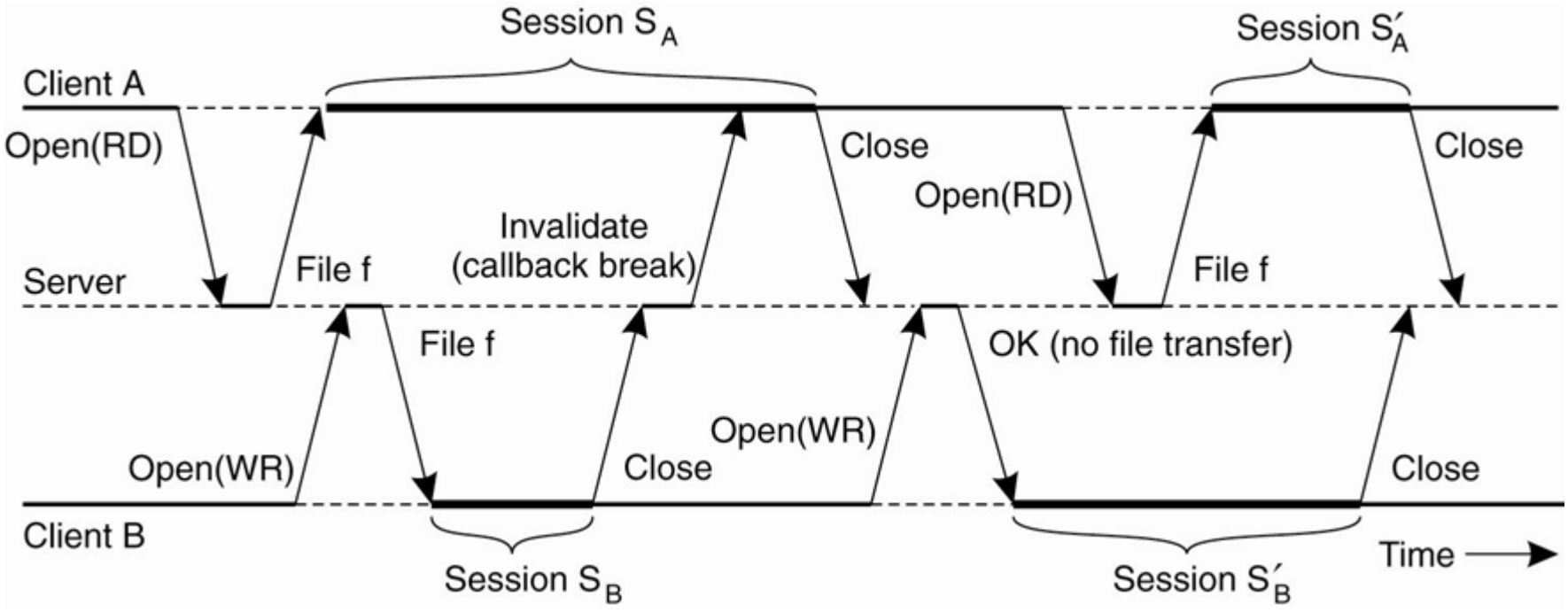


Figure 11-23. The use of local copies when opening a session in Coda.

Server Replication in Coda

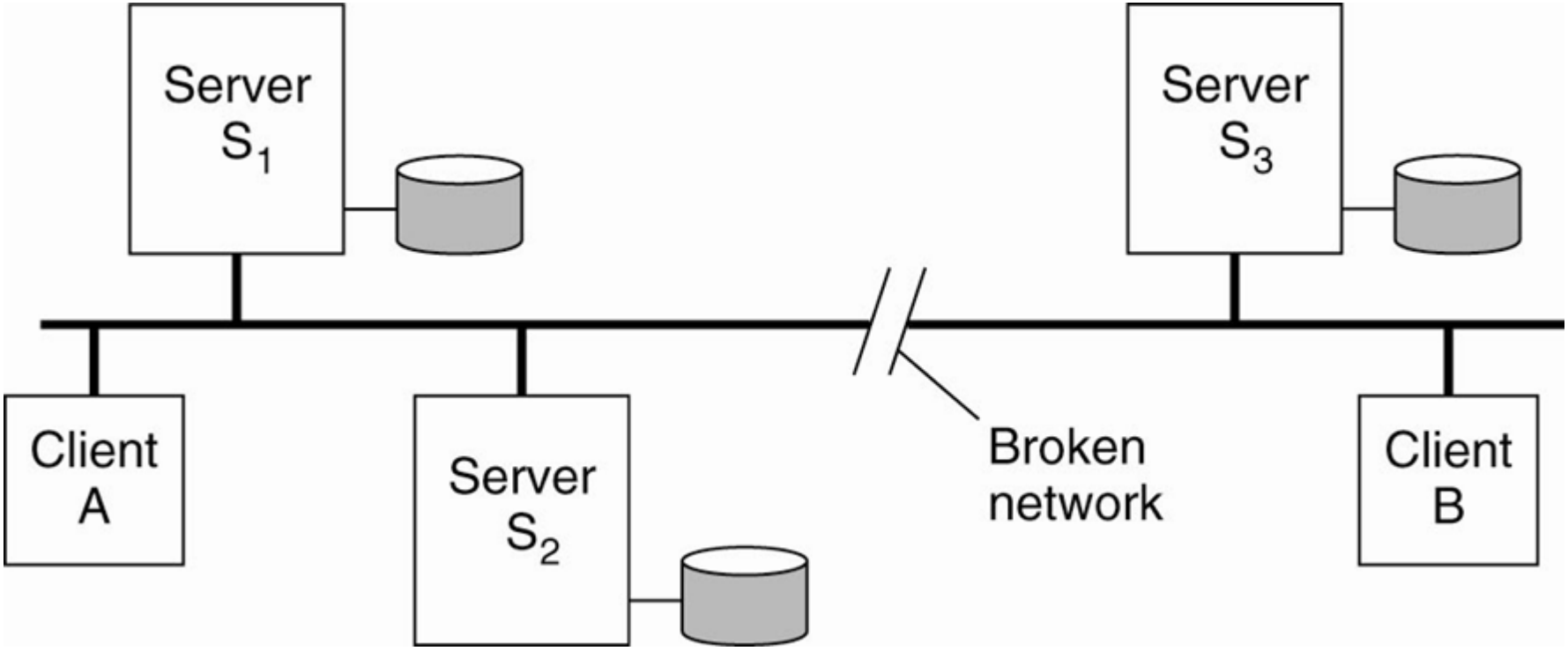


Figure 11-24. Two clients with a different AVSG for the same replicated file.

Structured Peer-to-Peer Systems

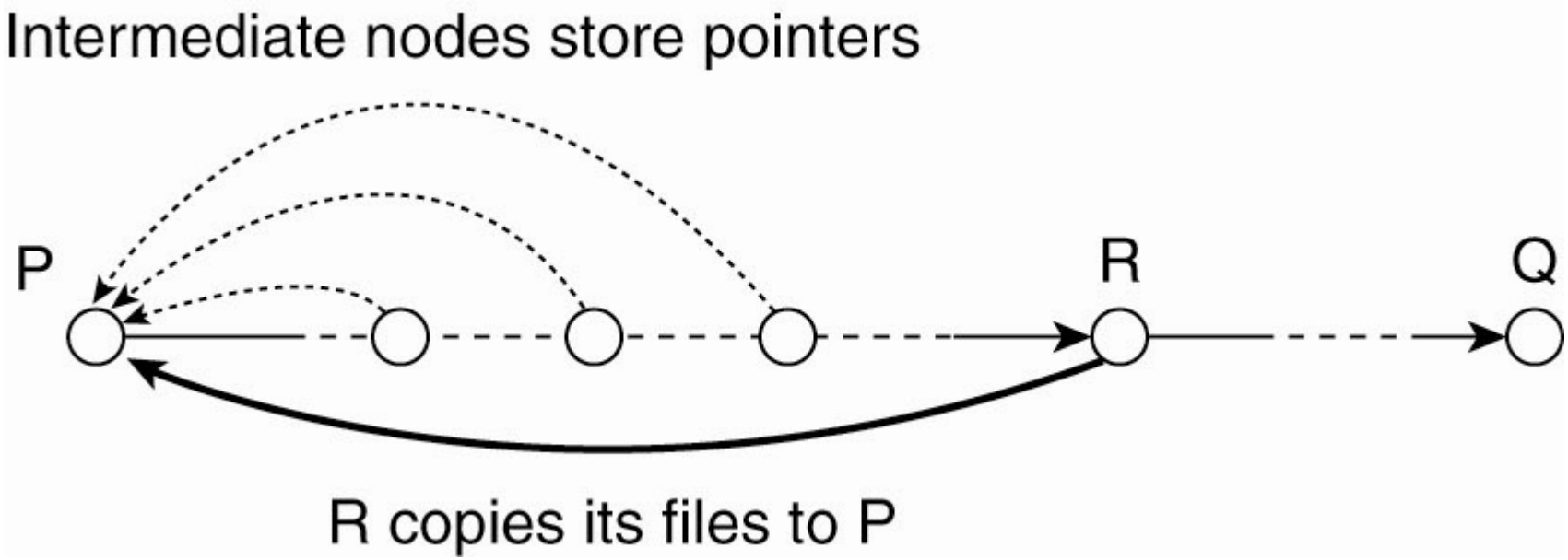


Figure 11-25. Balancing load in a peer-to-peer system by replication.

Handling Byzantine Failures

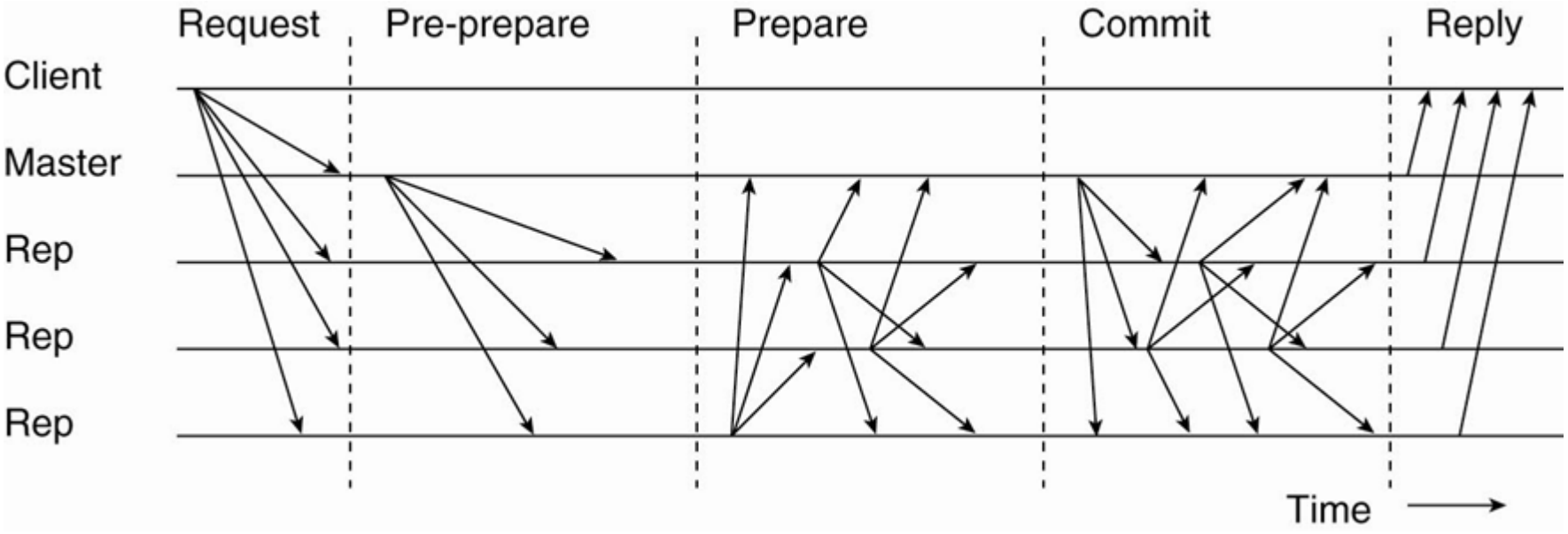


Figure 11-26. The different phases in Byzantine fault tolerance.

High Availability in Peer-to-Peer Systems

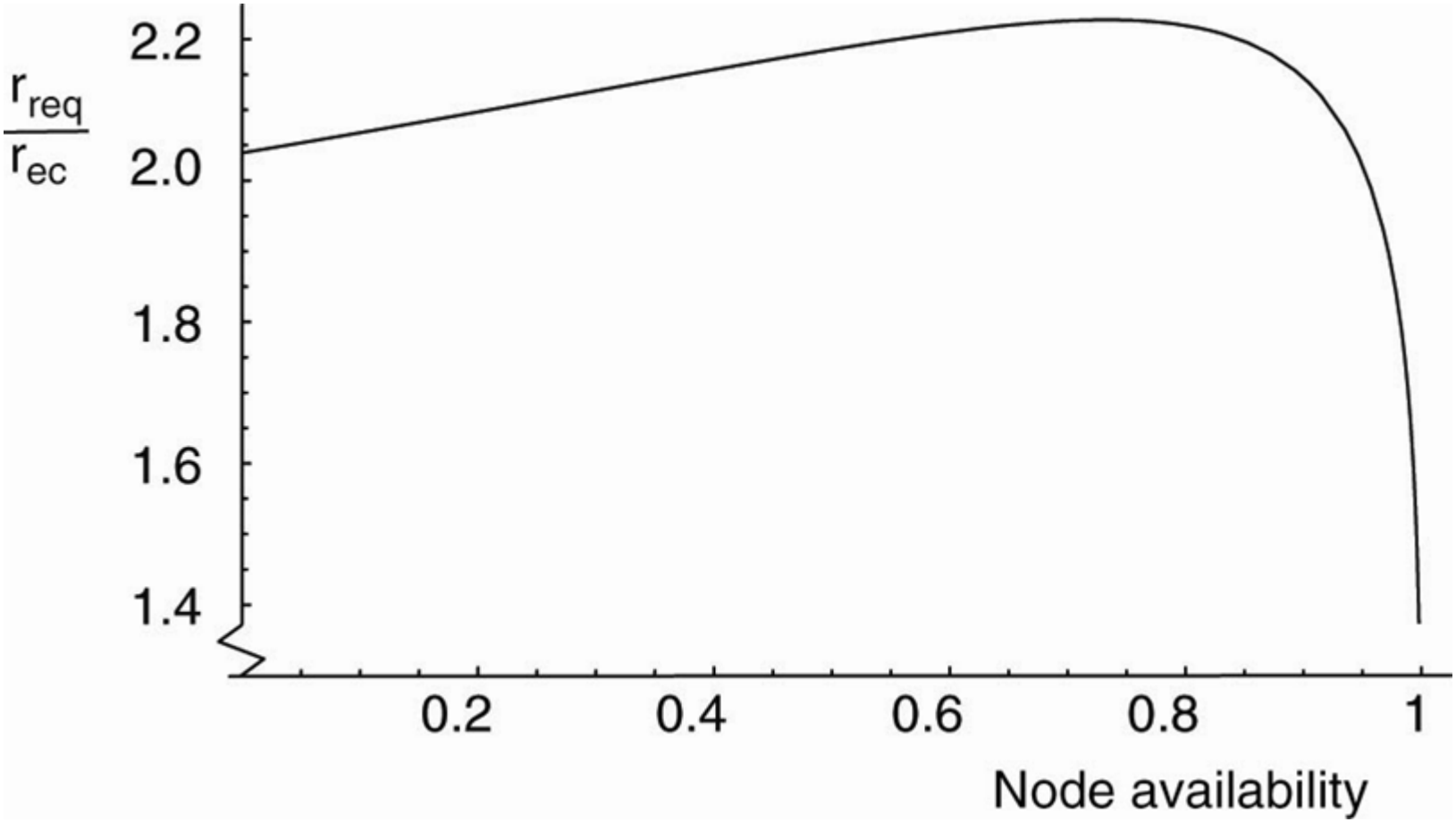


Figure 11-27. The ratio r_{rep}/r_{ec} as a function of node availability a.

Security in NFS

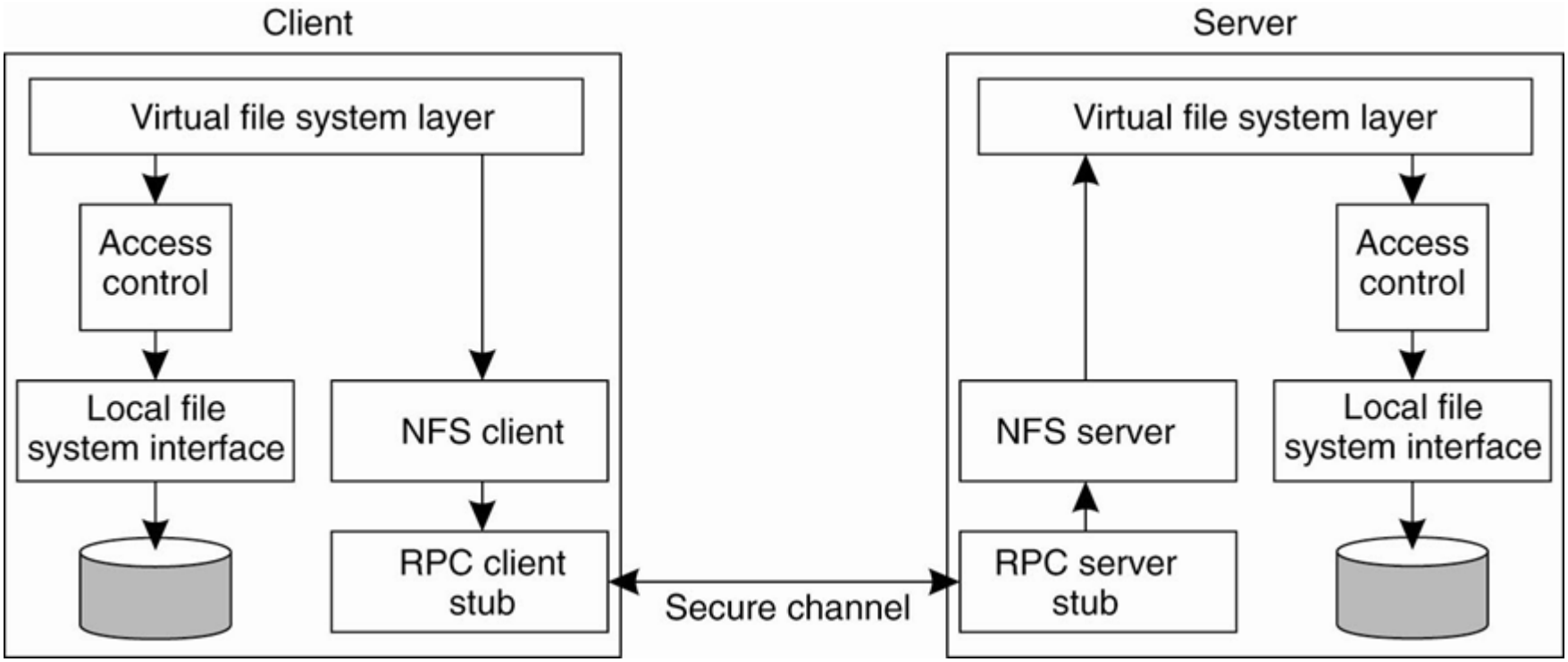


Figure 11-28. The NFS security architecture.

Secure RPCs

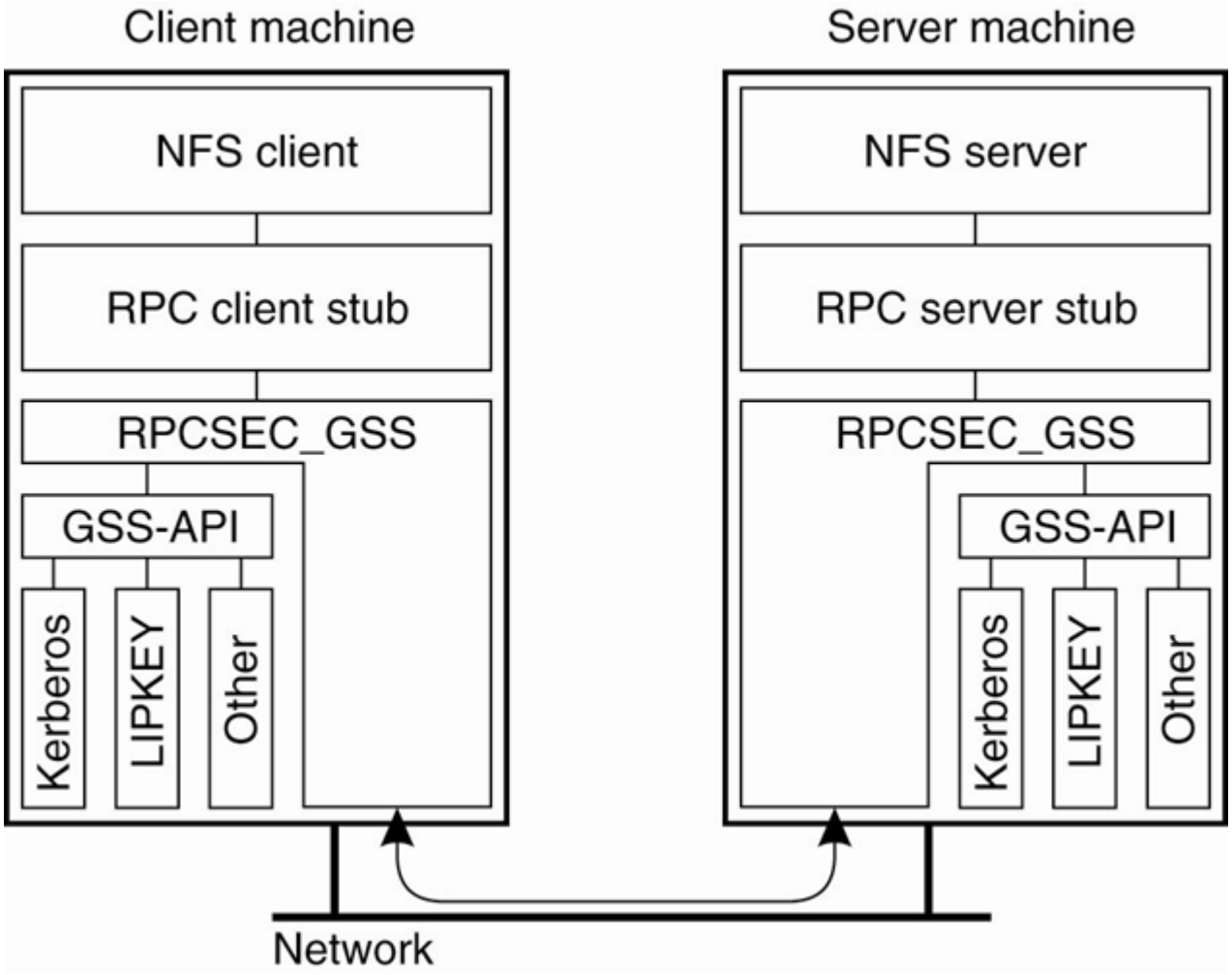


Figure 11-29. Secure RPC in NFSv4.

Access Control

Type of user	Description
Owner	The owner of a file
Group	The group of users associated with a file
Everyone	Any user or process
Interactive	Any process accessing the file from an interactive terminal
Network	Any process accessing the file via the network
Dialup	Any process accessing the file through a dialup connection to the server
Batch	Any process accessing the file as part of batch job
Anonymous	Anyone accessing the file without authentication
Authenticated	Any authenticated user or process
Service	Any system-defined service process

Figure 11-30. The various kinds of users and processes distinguished by NFS with respect to access control.

Decentralized Authentication (1)

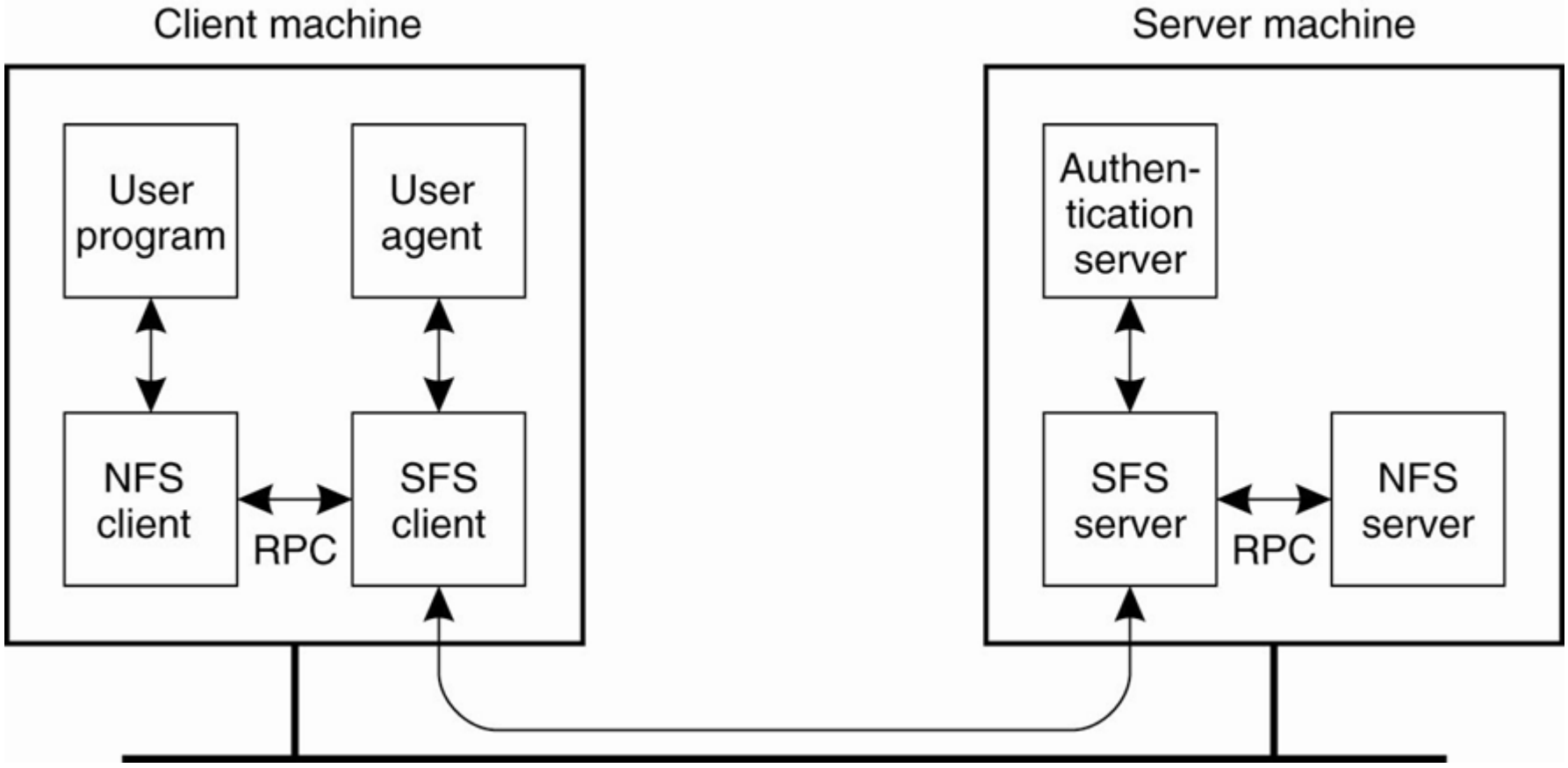


Figure 11-31. The organization of SFS.

Decentralized Authentication (2)

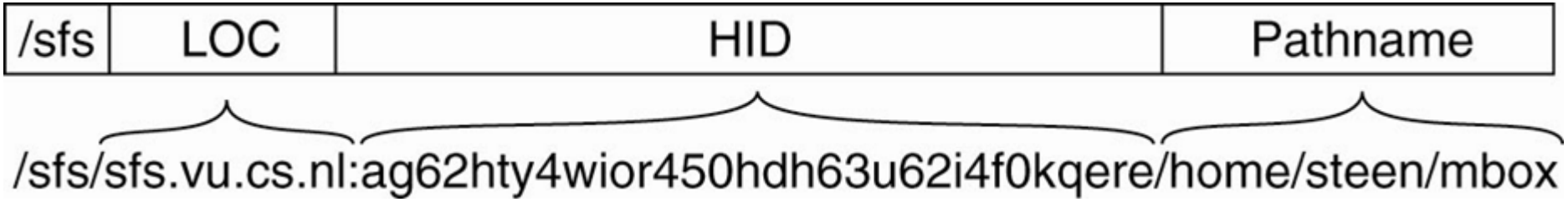


Figure 11-32. A self-certifying pathname in SFS.

Secure Lookups in DHT-Based Systems

Secure routing requires that three issues are dealt with:

1. Nodes are assigned identifiers in a secure way.
2. Routing tables are securely maintained.
3. Lookup requests are securely forwarded between nodes.

Secure Collaborative Storage

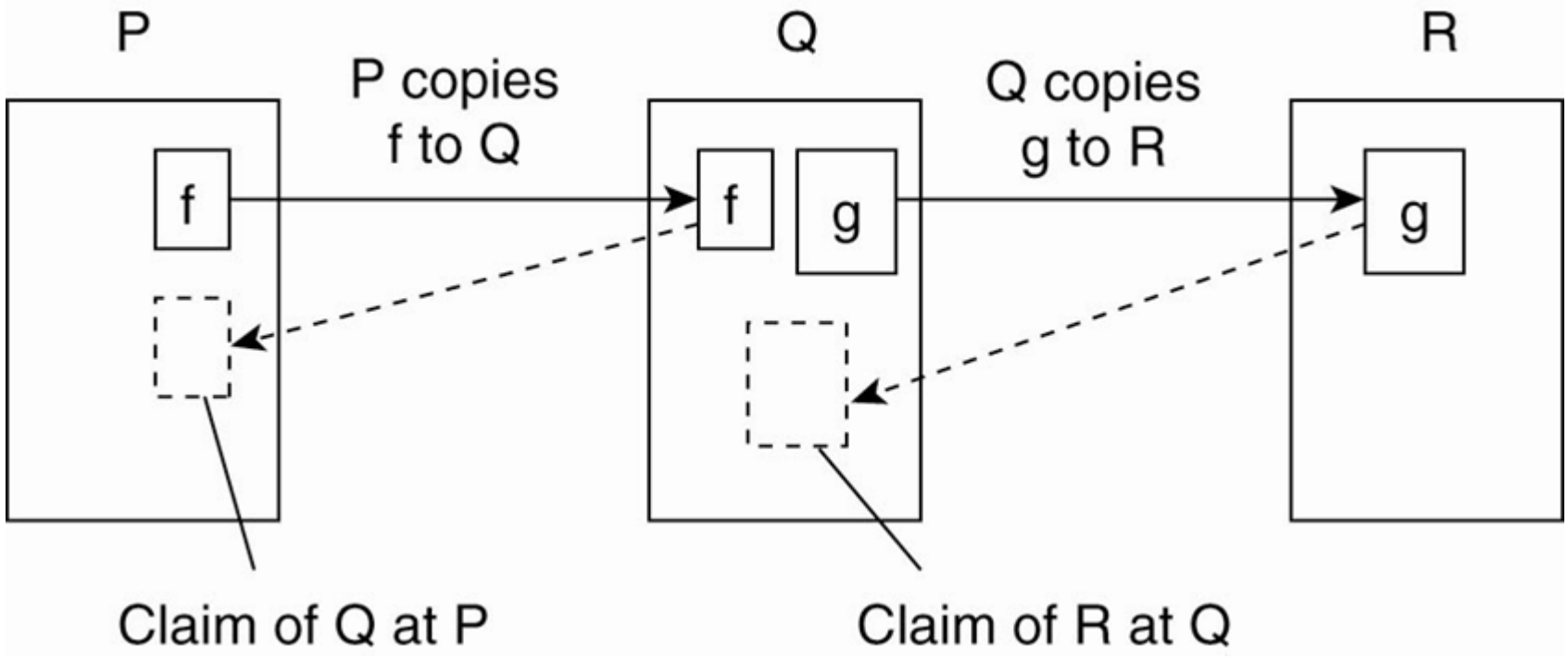


Figure 11-33. The principle of storage claims in the Samsara peer-to-peer system.