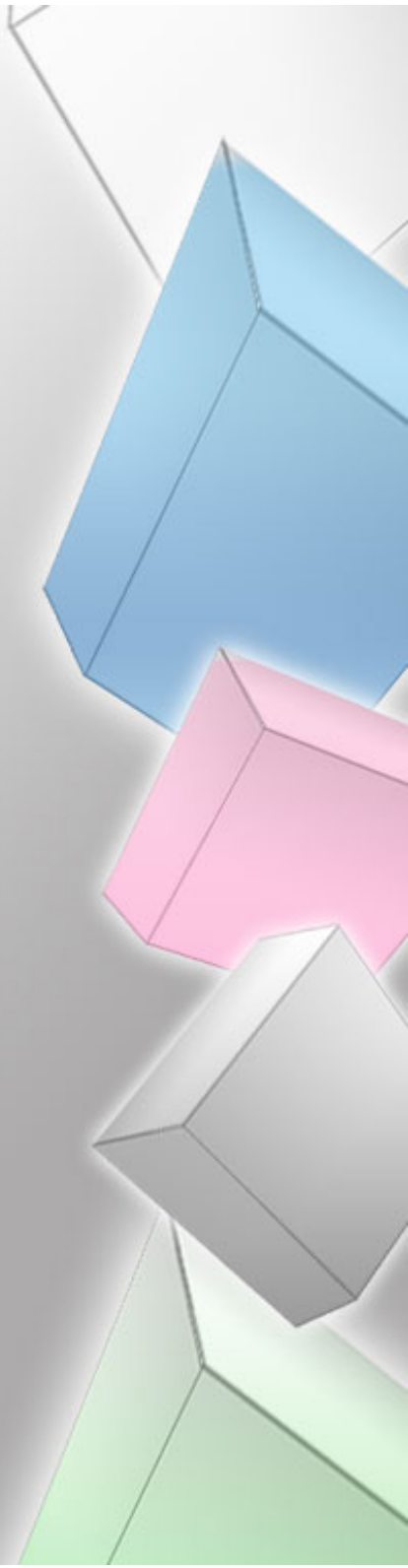
Advanced Topics in Operating Systems

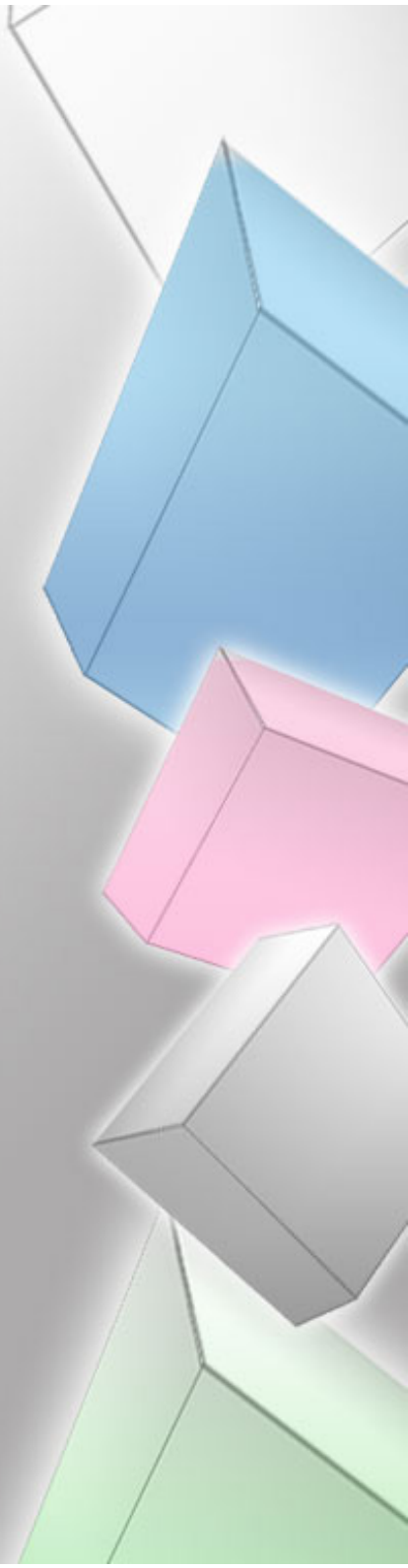
MSc in Computer Science
UNYT-UoG

Dr. Marenglen Biba
8-9-10 January 2010



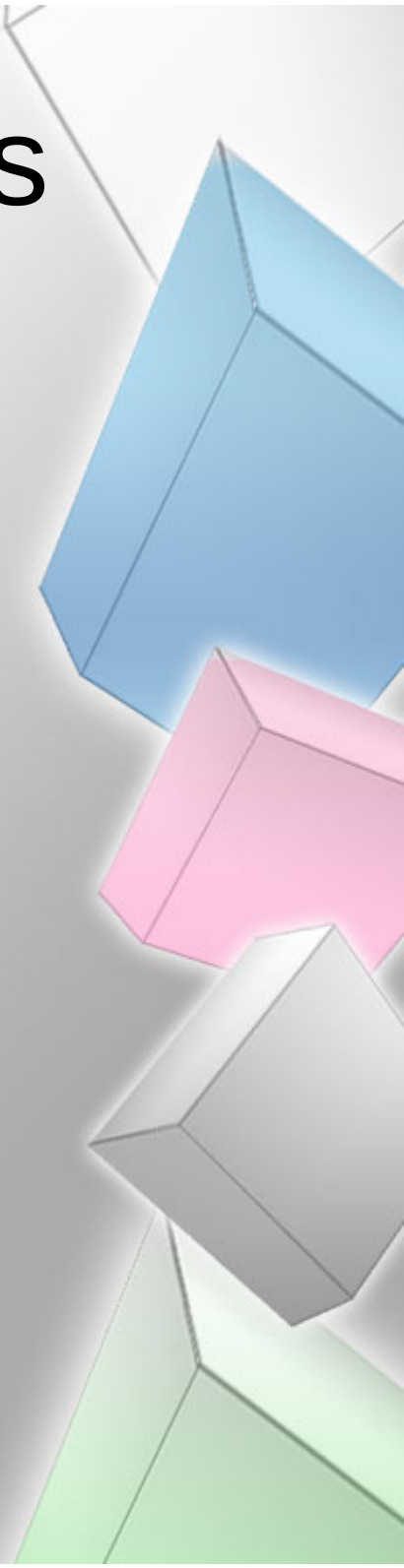
Lesson 11

01: Introduction
02: Architectures
03: Processes
04: Communication
05: Naming
06: Synchronization
07: Consistency & Replication
08: Fault Tolerance
09: Security
10: Distributed Object-Based Systems
11: Distributed File Systems
12: Distributed Web-Based Systems
13: Distributed Coordination-Based Systems



Distributed File Systems

- Architecture
- Processes
- Communication
- Naming
- Synchronization
- Consistency and Replication
- Security



Client-Server Architectures (1)

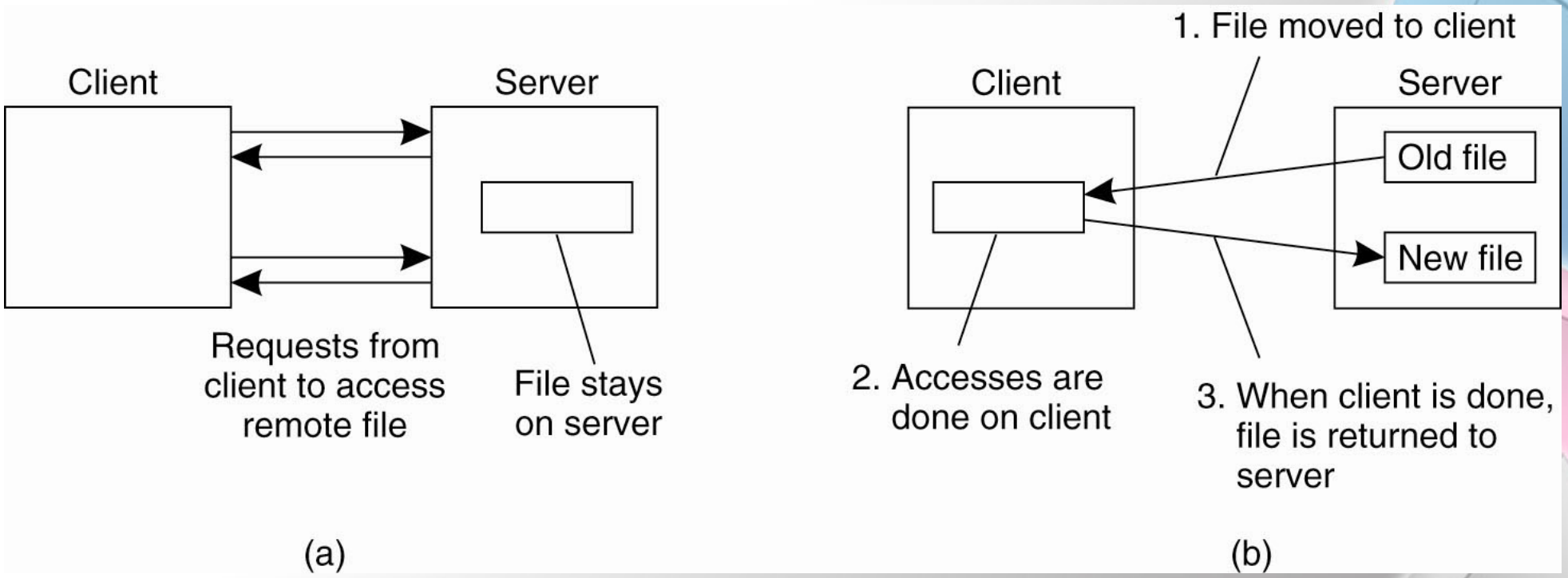


Figure 11-1. (a) The remote access model. (b) The upload/download model.

Client-Server Architectures (2)

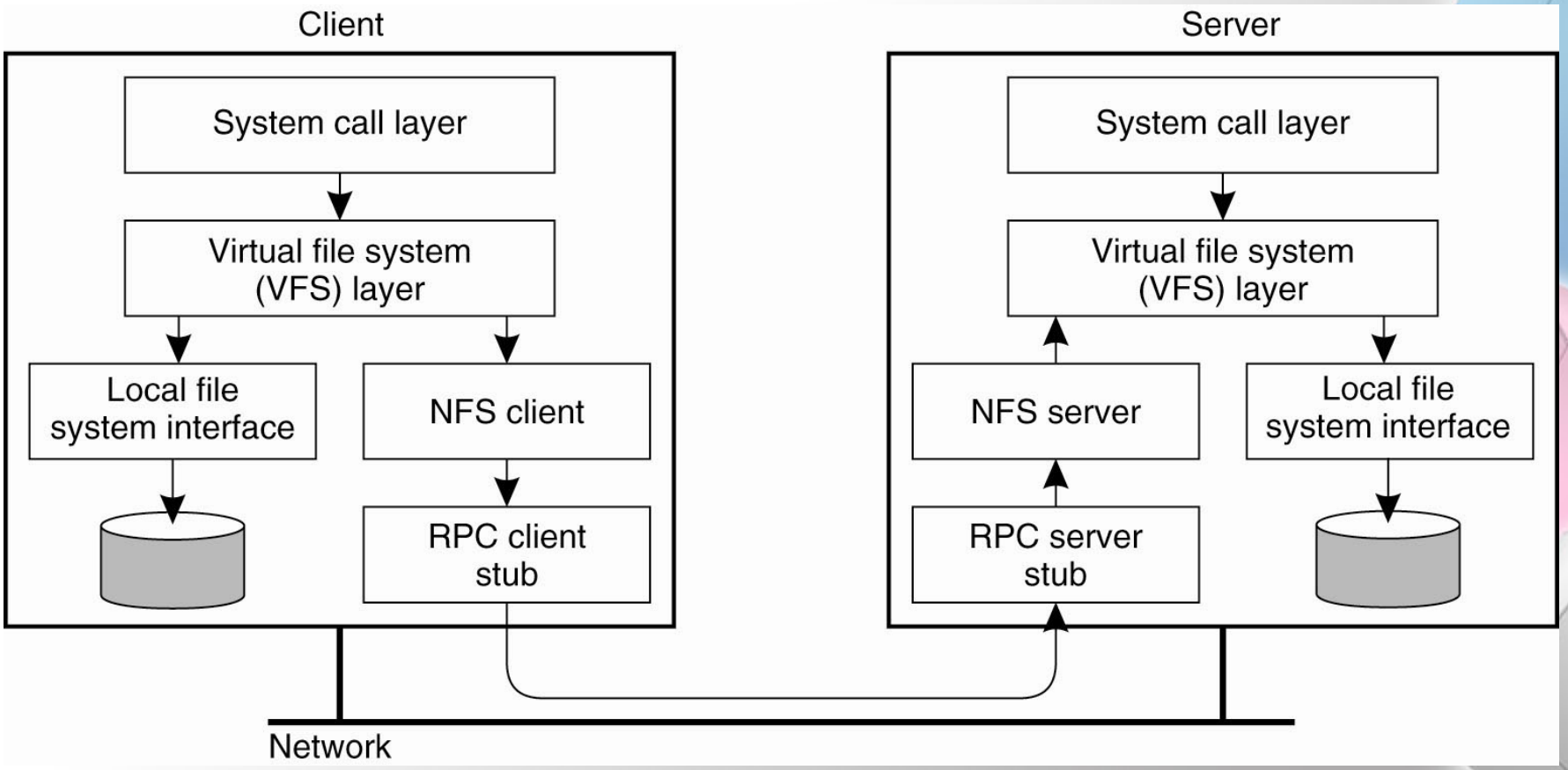


Figure 11-2. The basic NFS architecture for UNIX systems.

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)

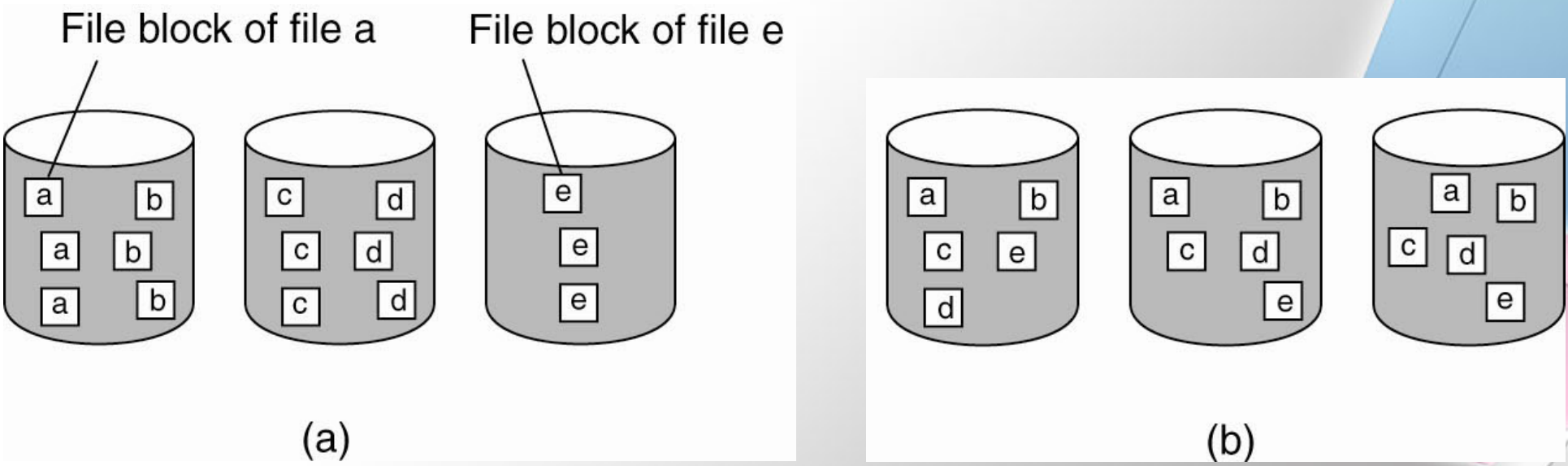


Figure 11-4. The difference between (a) distributing whole files across several servers and (b) striping files for parallel access.

Cluster-Based Distributed File Systems
Google File System

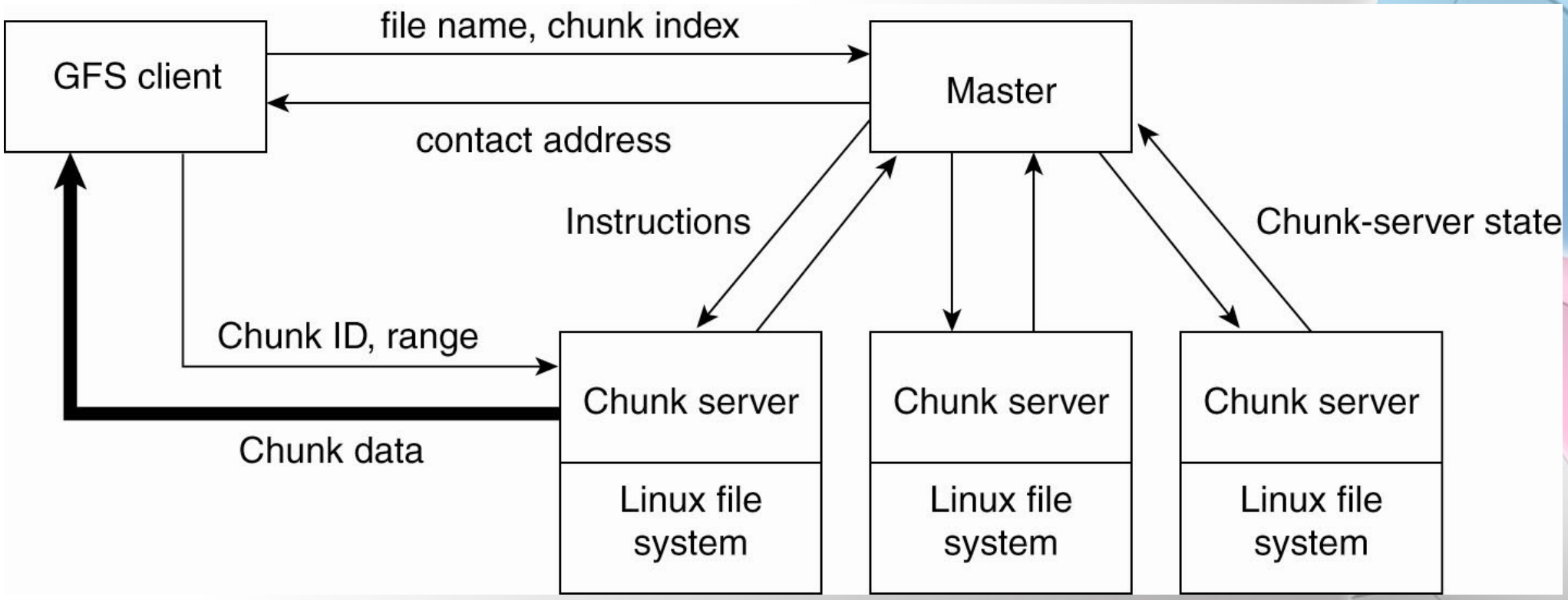


Figure 11-5. The organization of a Google cluster of servers.

Symmetric Architectures: P2P

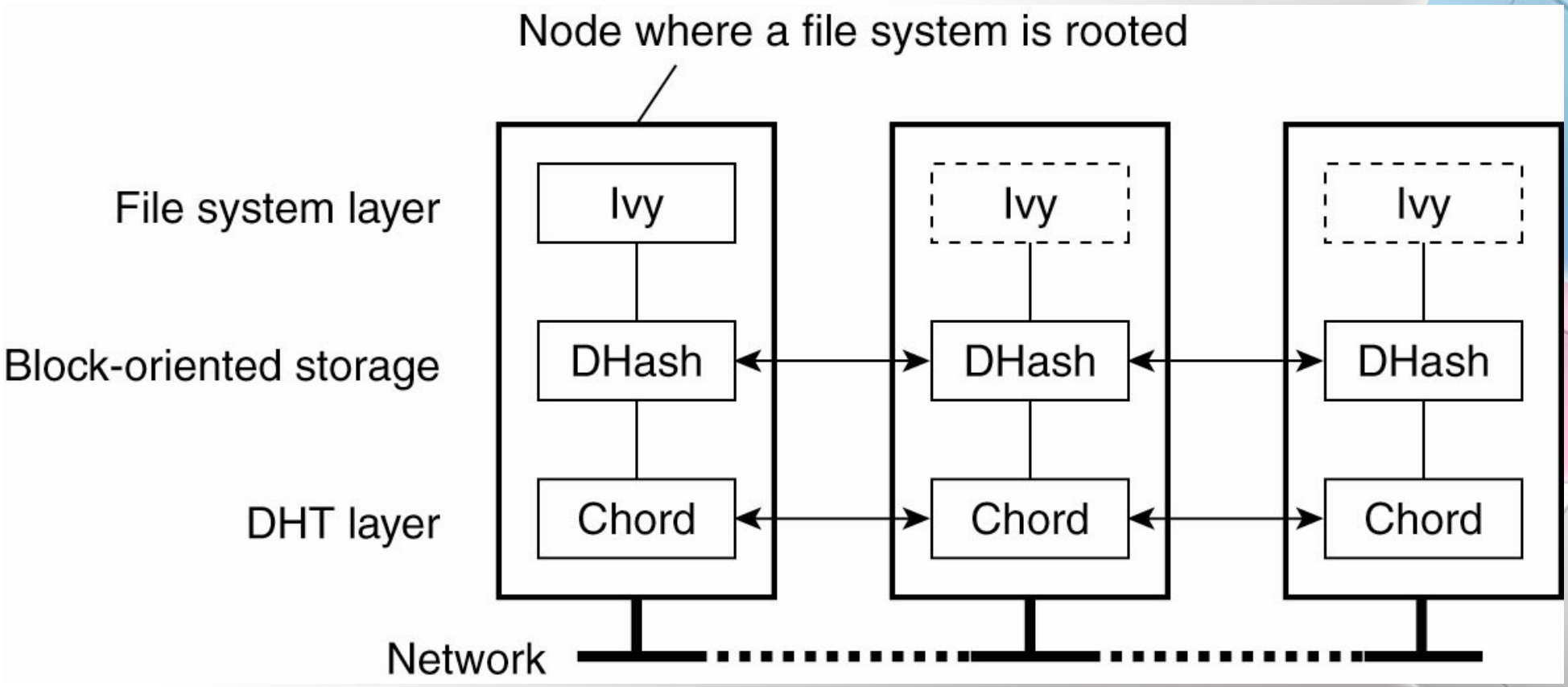
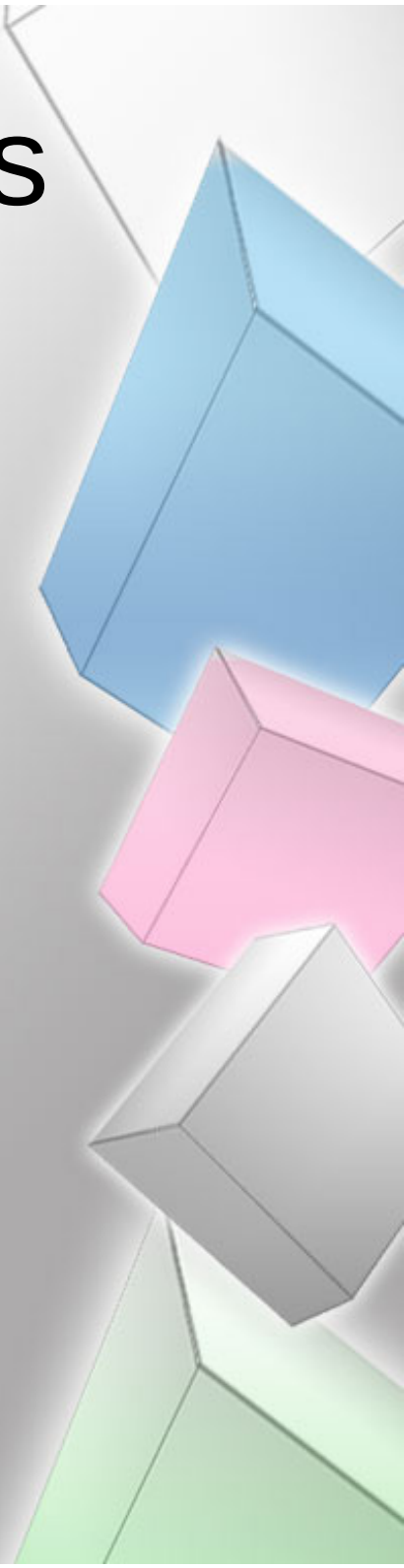


Figure 11-6. The organization of the Ivy distributed file system.

Distributed File Systems

- Architecture
- Processes
- Communication
- Naming
- Synchronization
- Consistency and Replication
- Security

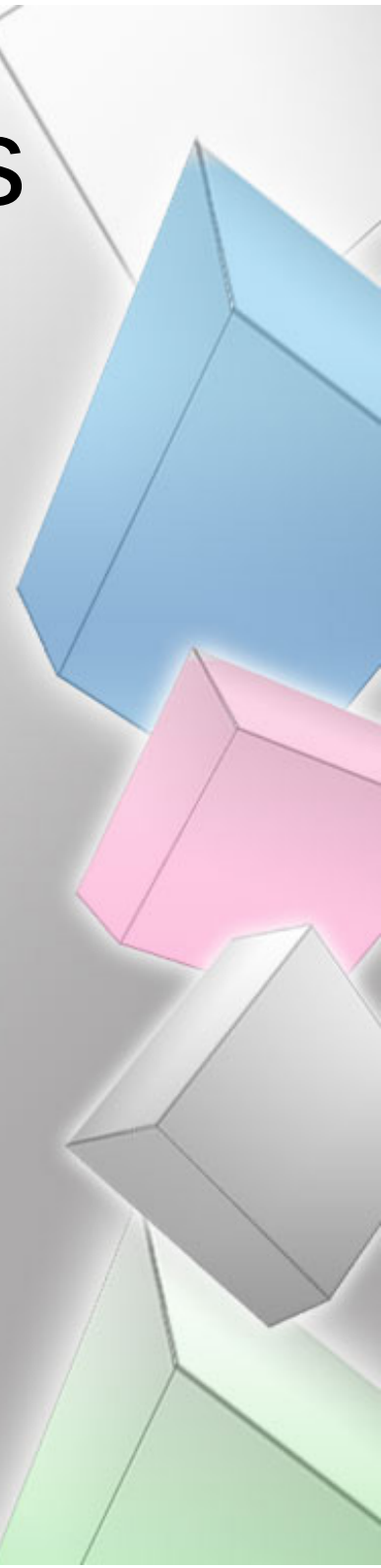


Processes

- When it comes to processes, distributed file systems have no unusual properties.
- In many cases, there will be different types of cooperating processes: storage servers and file managers, just as we described above for the various organizations.
- NFS Servers
  - Stateless: simplicity
  - Stateful : NFS version 4. Keep information about clients
    - Supposed to work also in WAN (wide-are networks)

Distributed File Systems

- Architecture
- Processes
- Communication
- Naming
- Synchronization
- Consistency and Replication
- Security



Remote Procedure Calls in NFS

Open Network Computing RPC (ONC RPC) protocol

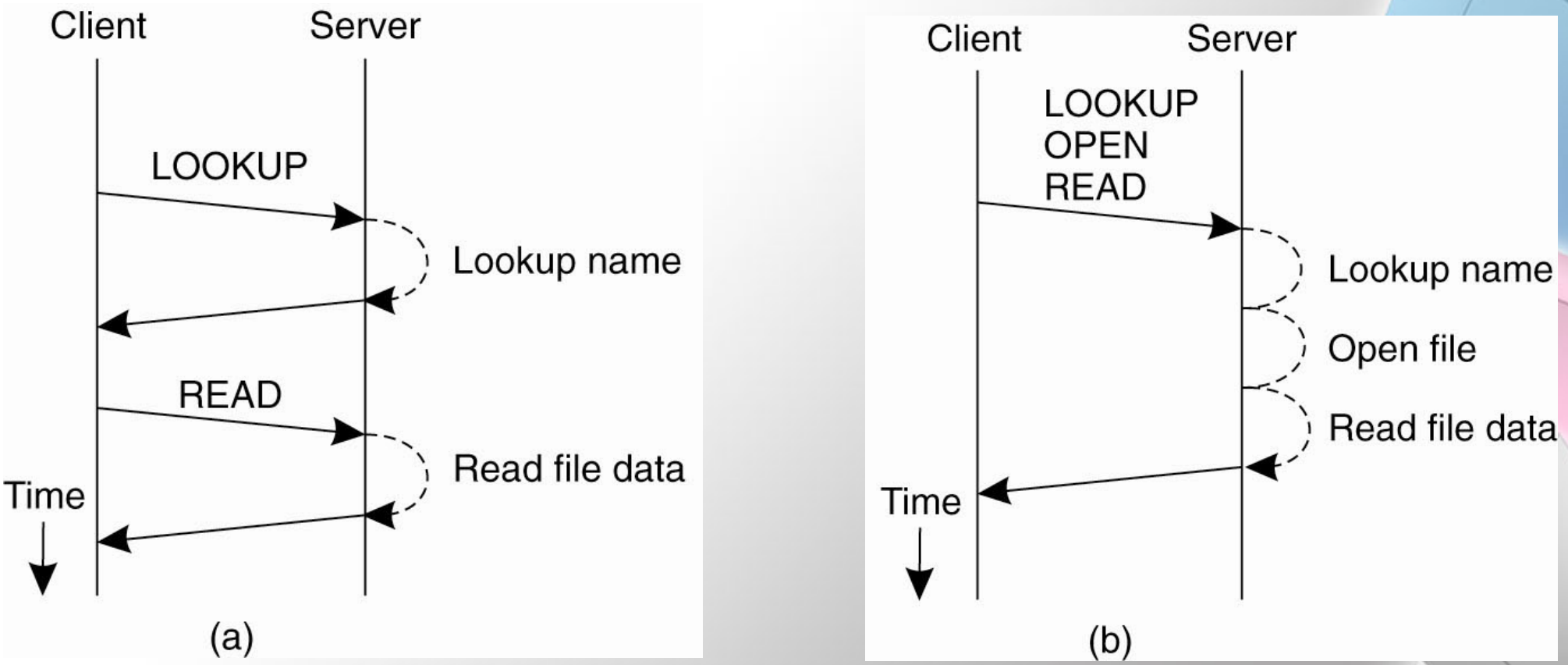
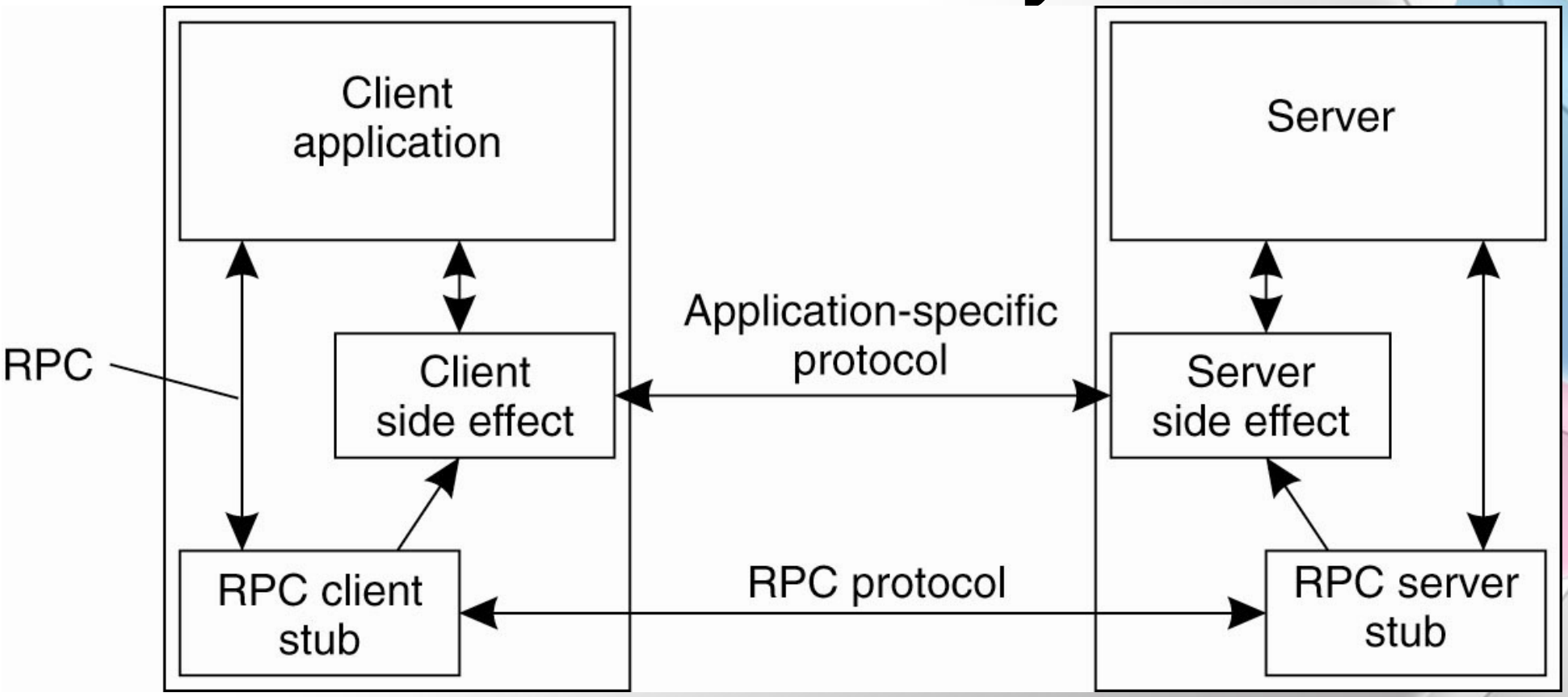


Figure 11-7. (a) Reading data from a file in NFS version 3. (b) Reading data using a compound procedure in version 4.

The RPC2 Subsystem



A side effect is a mechanism by which the client and server can communicate using an application-specific protocol. Consider, for example, a client opening a file at a video server.

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

File-Oriented Communication in Plan 9

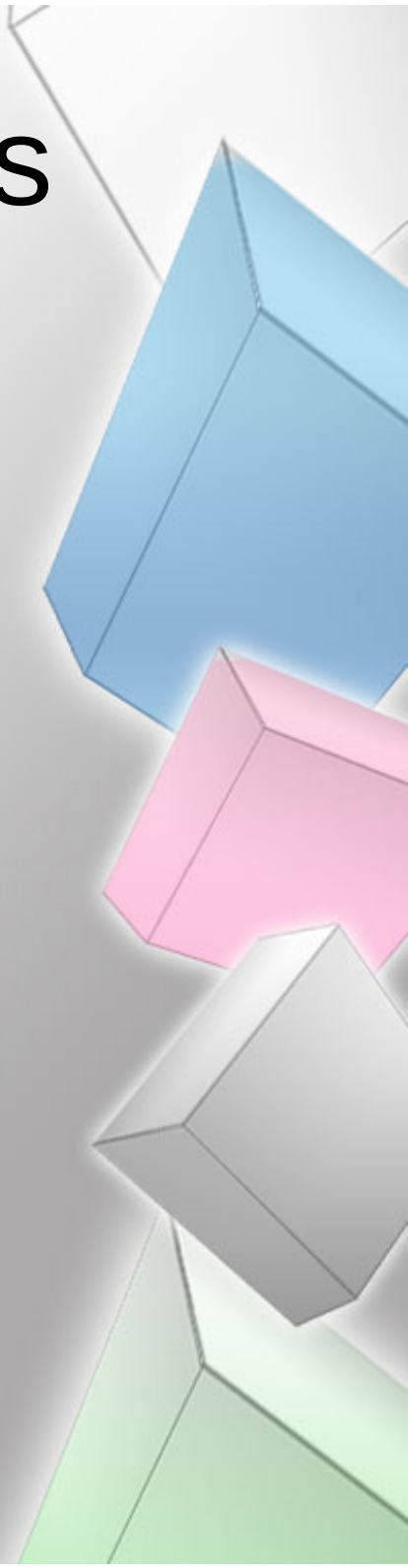
Plan 9 (Pike et al., 1995). is not so much a distributed file system, but rather a *file-based distributed system.*

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.

Distributed File Systems

- Architecture
- Processes
- Communication
- Naming
- Synchronization
- Consistency and Replication
- Security



Naming in NFS (1)

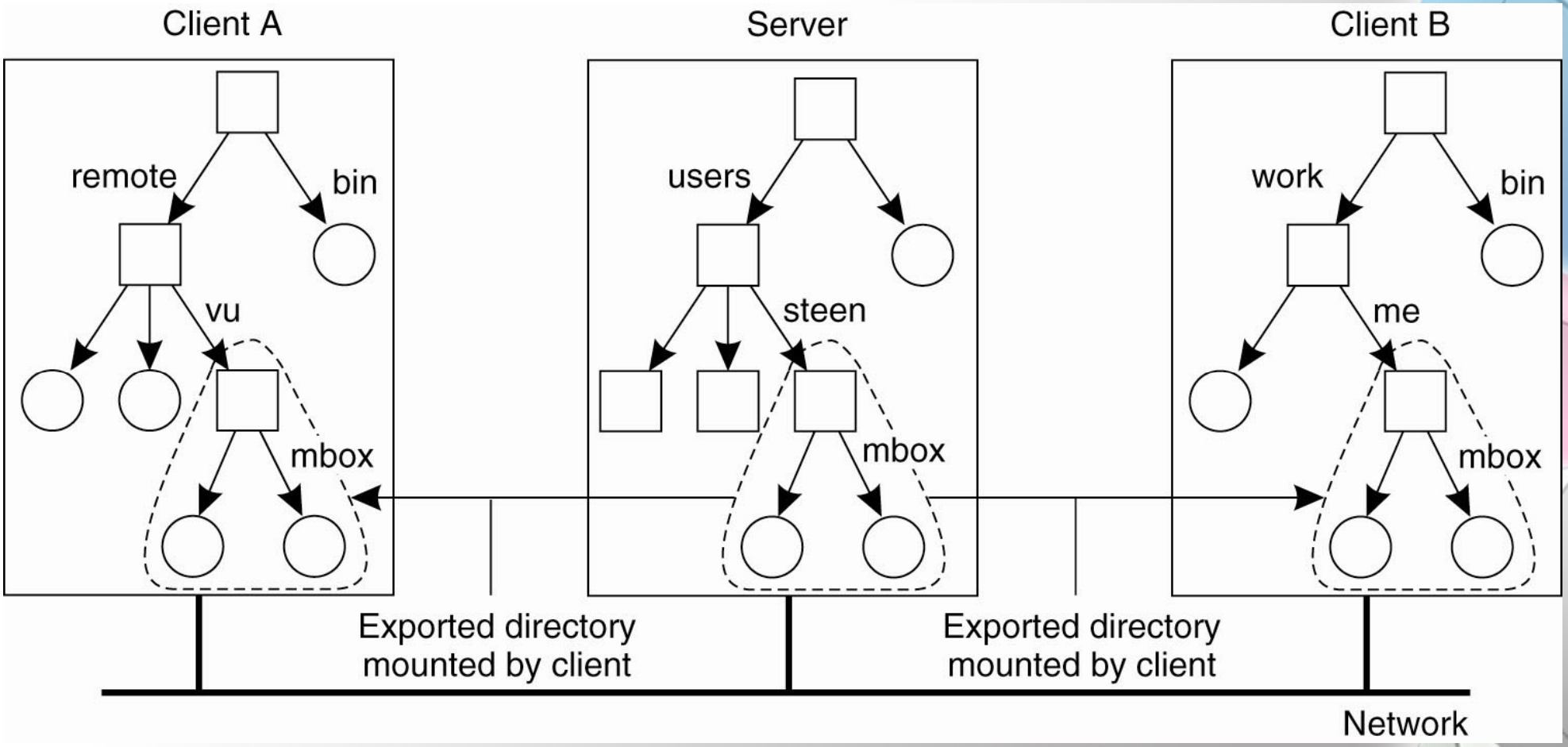


Figure 11-11. Mounting (part of) a remote file system in NFS.

Naming in NFS (2)

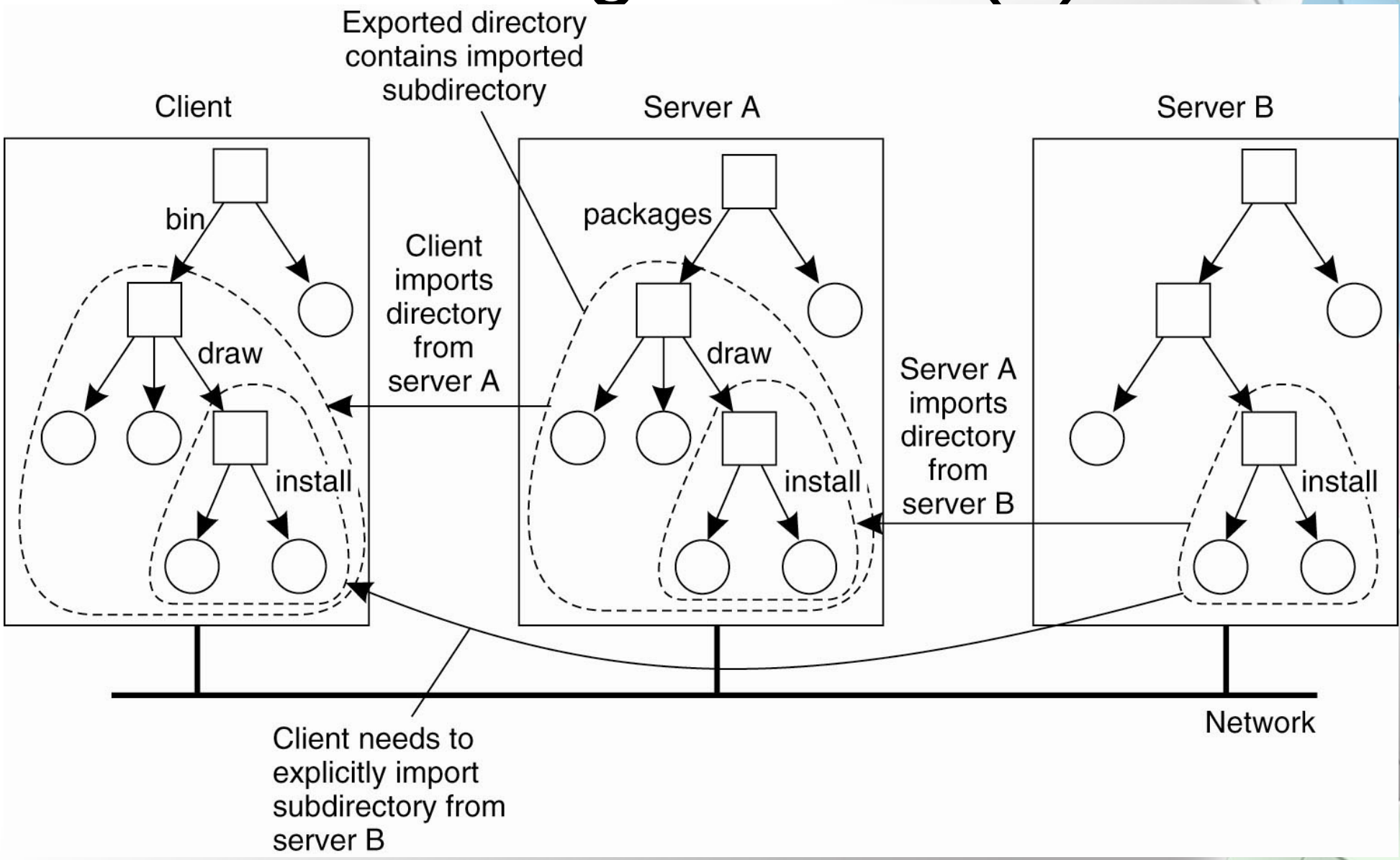


Figure 11-12. Mounting nested directories from multiple servers in NFS.

File Handles

- A file handle is a reference to a file within a file system. It is independent of the name of the file it refers to.
  - A file handle is created by the server that is hosting the file system and is unique with respect to all file systems exported by the server.
  - It is created when the file is created. The client is kept ignorant of the actual content of a file handle; it is completely opaque. File handles were 32 bytes in NFS version 2, but were variable up to 64 bytes in version 3 and 128 bytes in version 4.
- Ideally, a file handle is implemented as a **true identifier** for a file relative to a file system.
- **For one thing, this means that as long as the file exists, it should have one and the same file handle.**

Automounting (1)

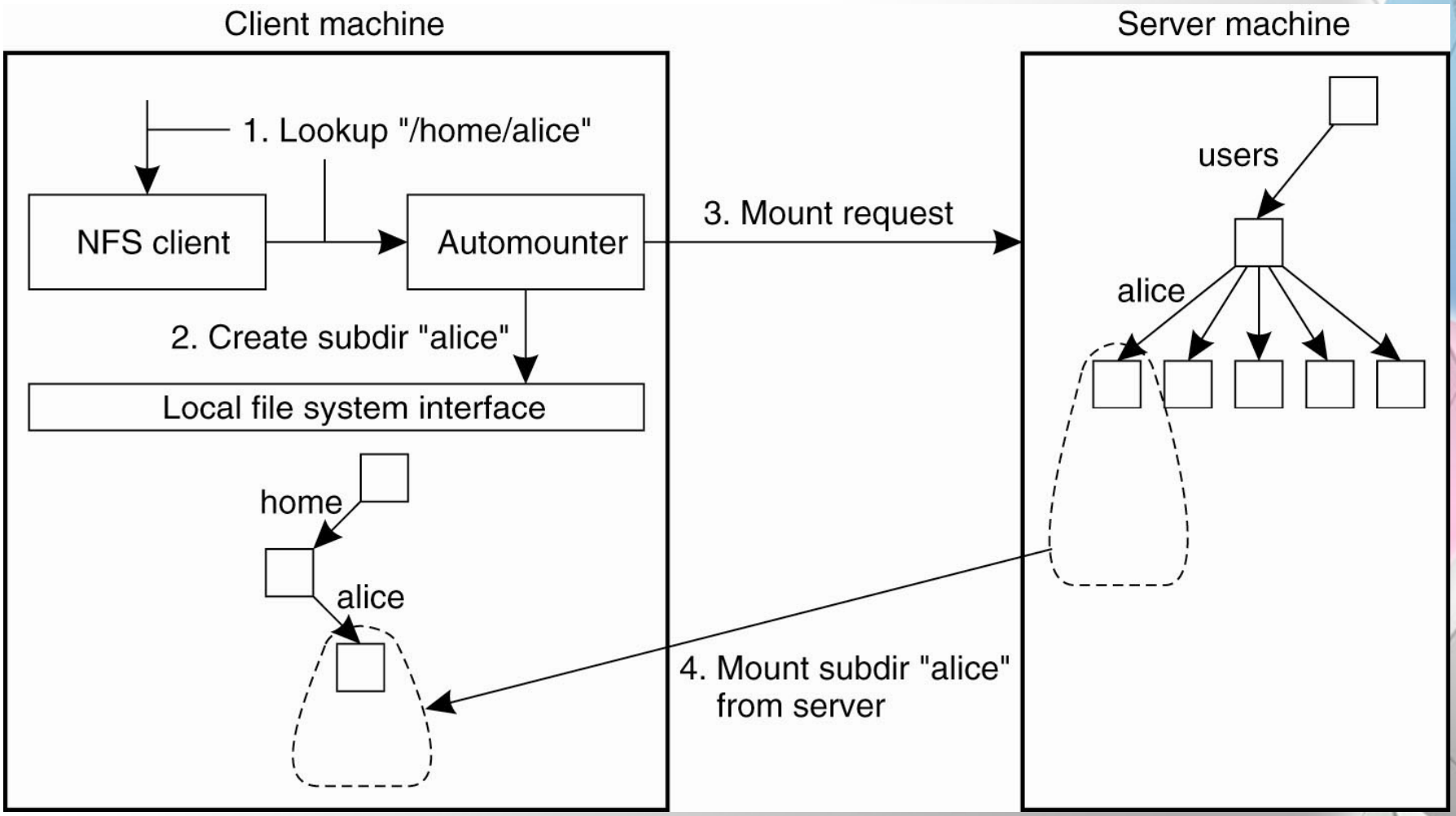
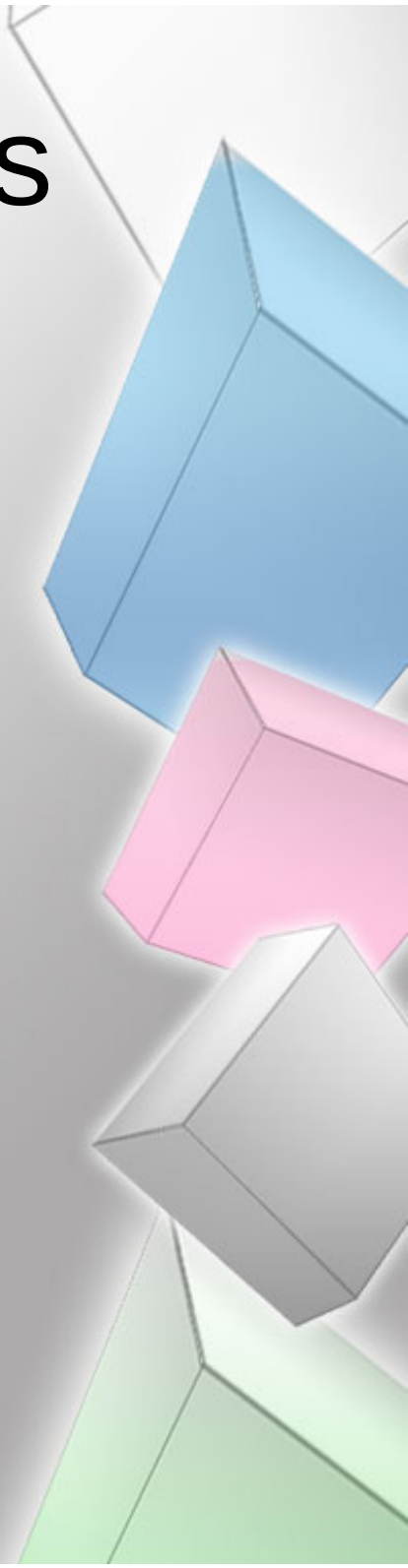


Figure 11-13. A simple automounter for NFS.

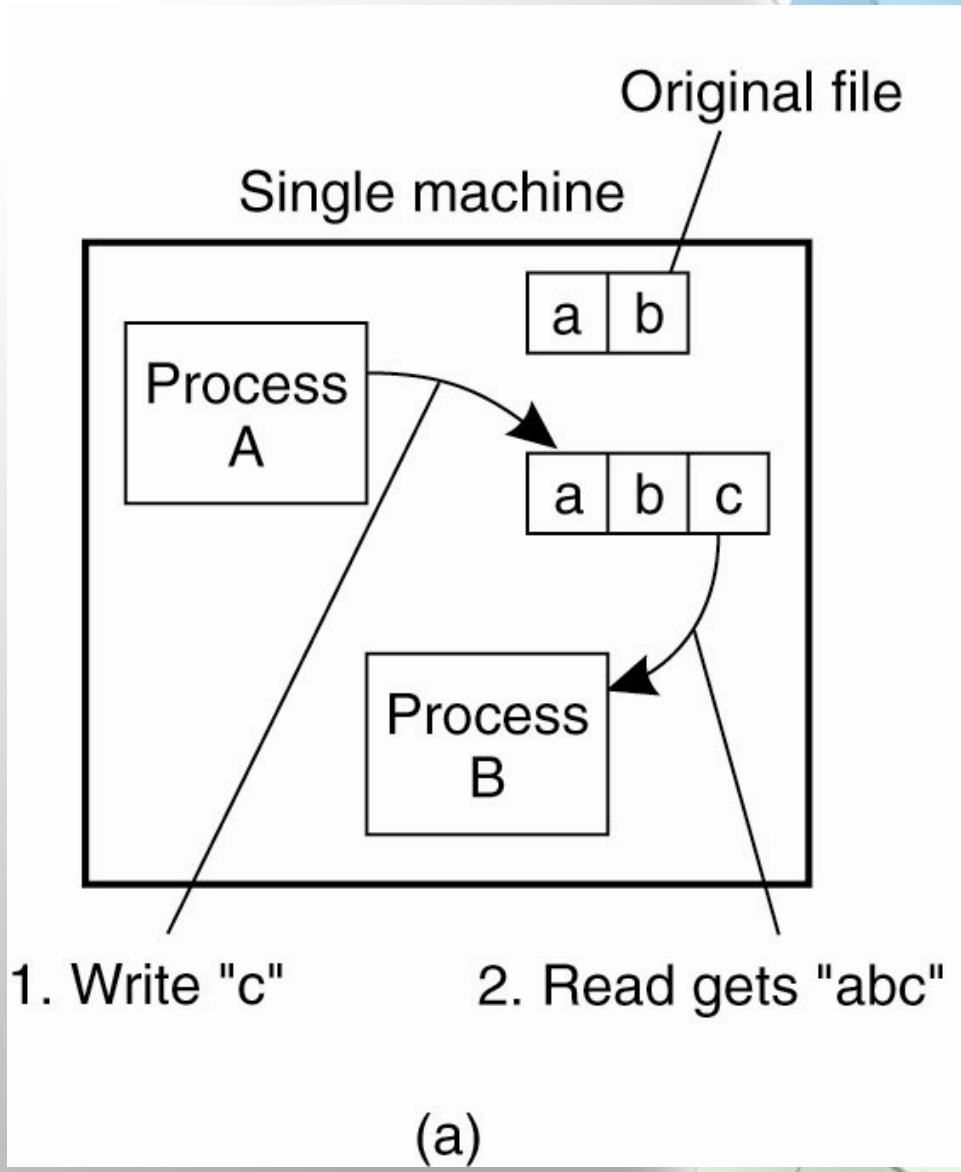
Distributed File Systems

- Architecture
- Processes
- Communication
- Naming
- Synchronization
- Consistency and Replication
- Security



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 (UNIX Semantics).



Semantics of File Sharing (2)

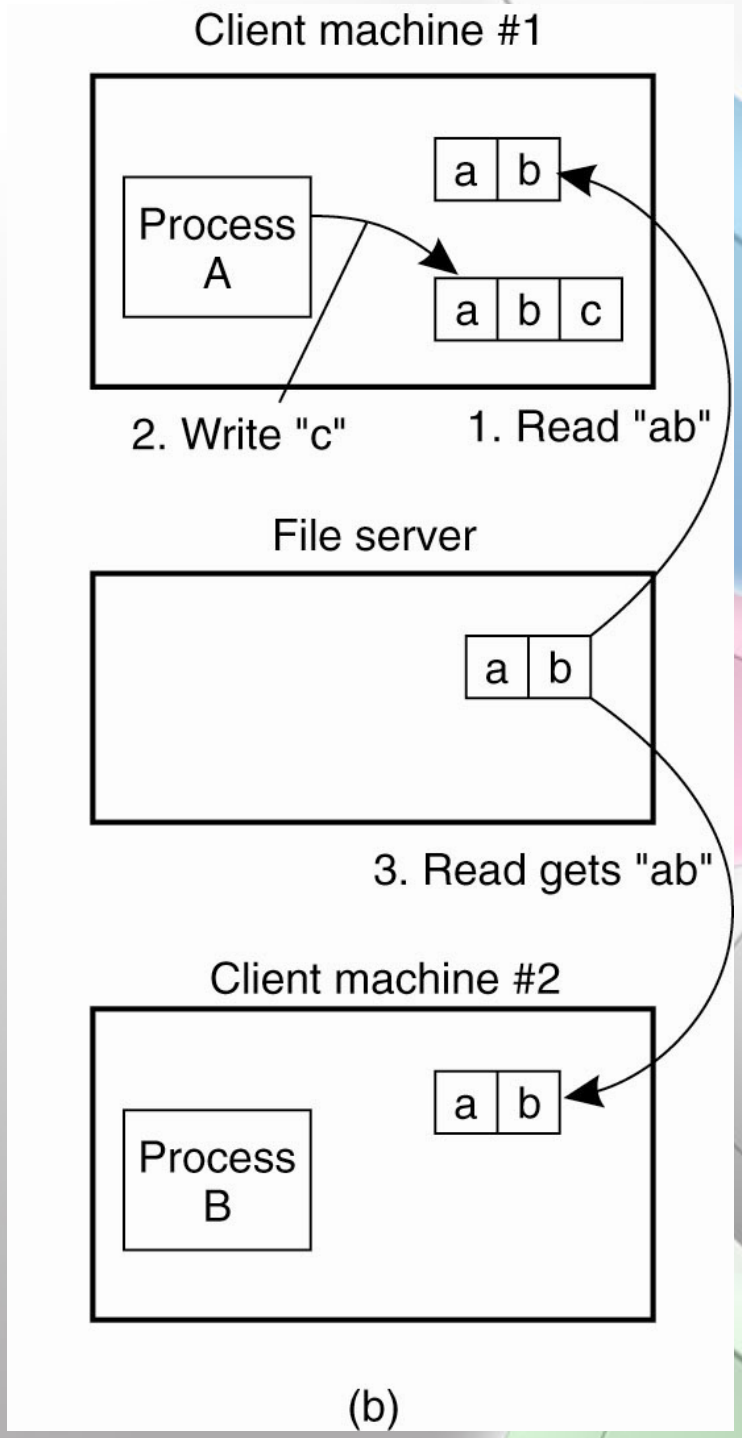


Figure 11-16. (b) In a distributed system with caching, obsolete values may be returned.

Session Semantics

- "Changes to an open file are initially visible only to the process (or possibly machine) that modified the file. Only when the file is closed are the changes made visible to other processes (or machines)".

- OK, but what happens if two or more clients are simultaneously caching and modifying the same file? Solutions:
  - Immutable files
  - Atomic transactions

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.

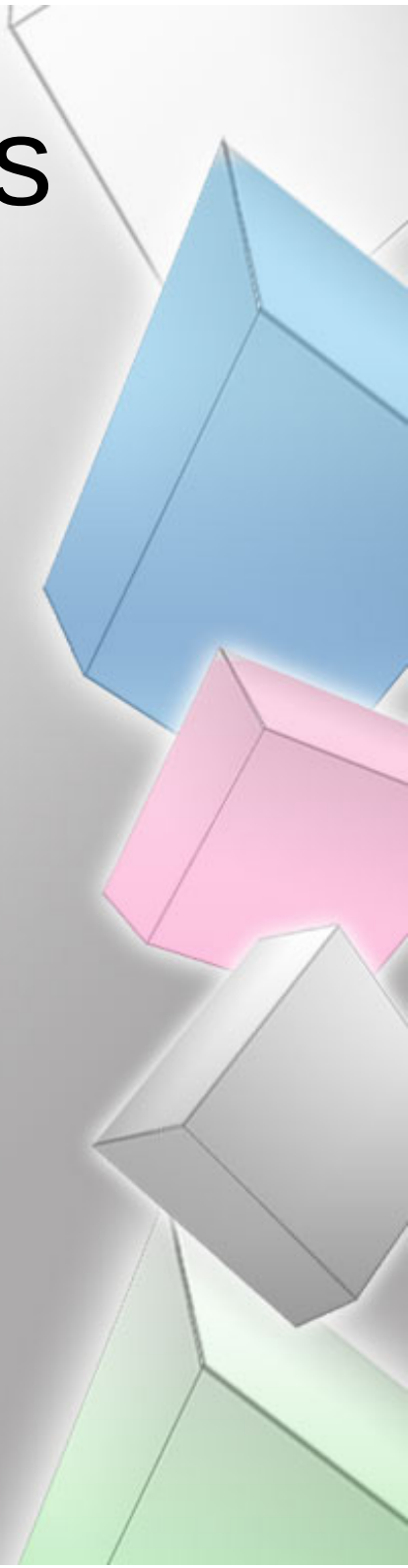
File Locking (1)

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.

Distributed File Systems

- Architecture
- Processes
- Communication
- Naming
- Synchronization
- Consistency and Replication
- Security



Client-Side Caching (1)

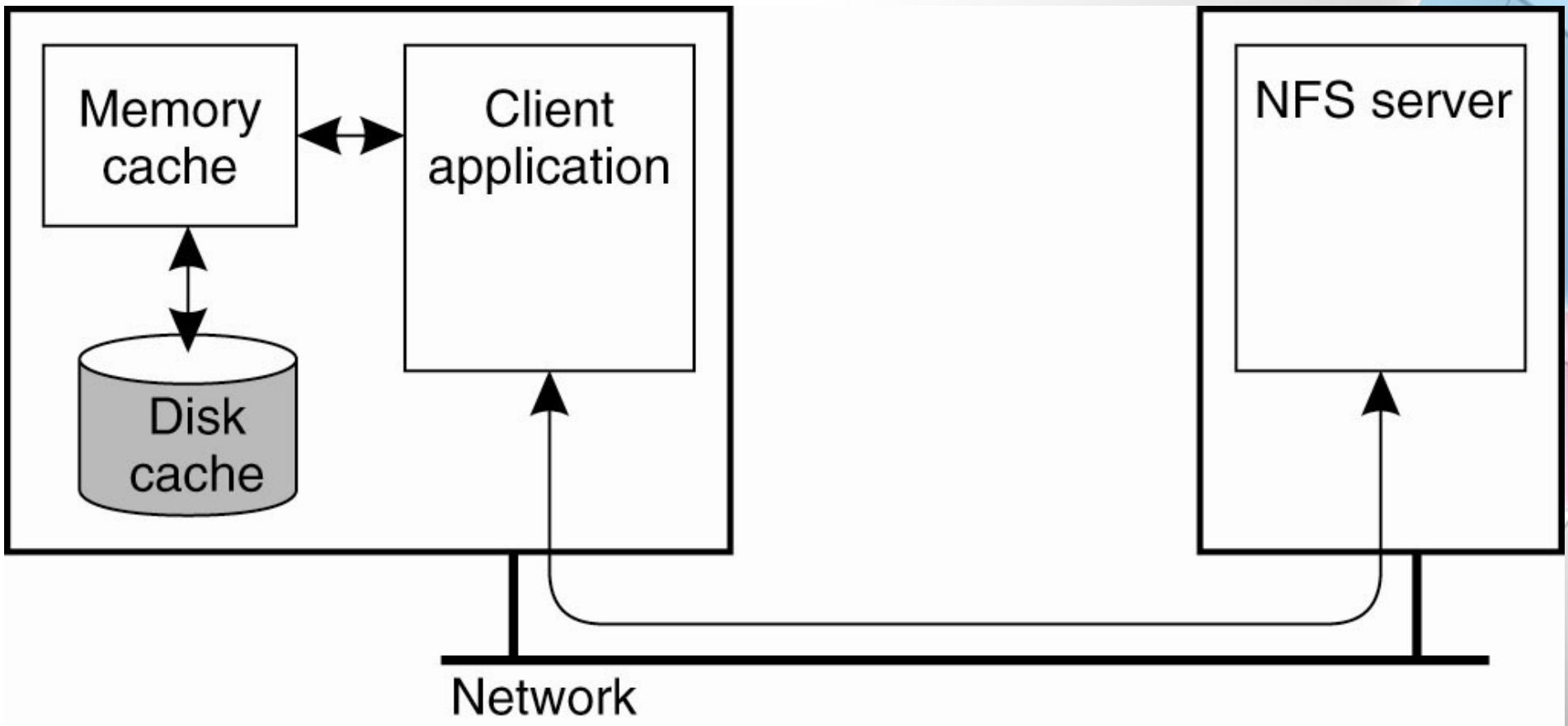


Figure 11-21. Client-side caching in NFS.

Client-Side Caching (2)

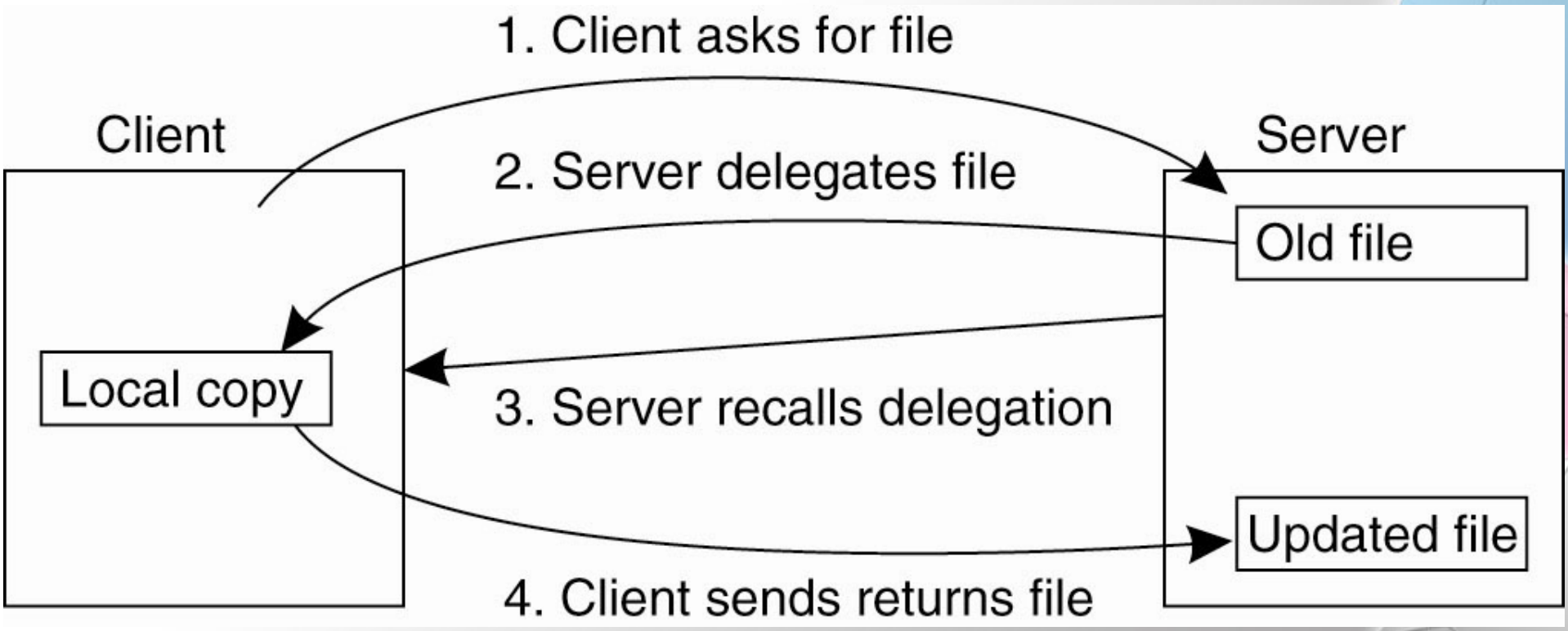


Figure 11-22. Using the NFSv4 callback mechanism to recall file delegation.

Server Replication in Coda

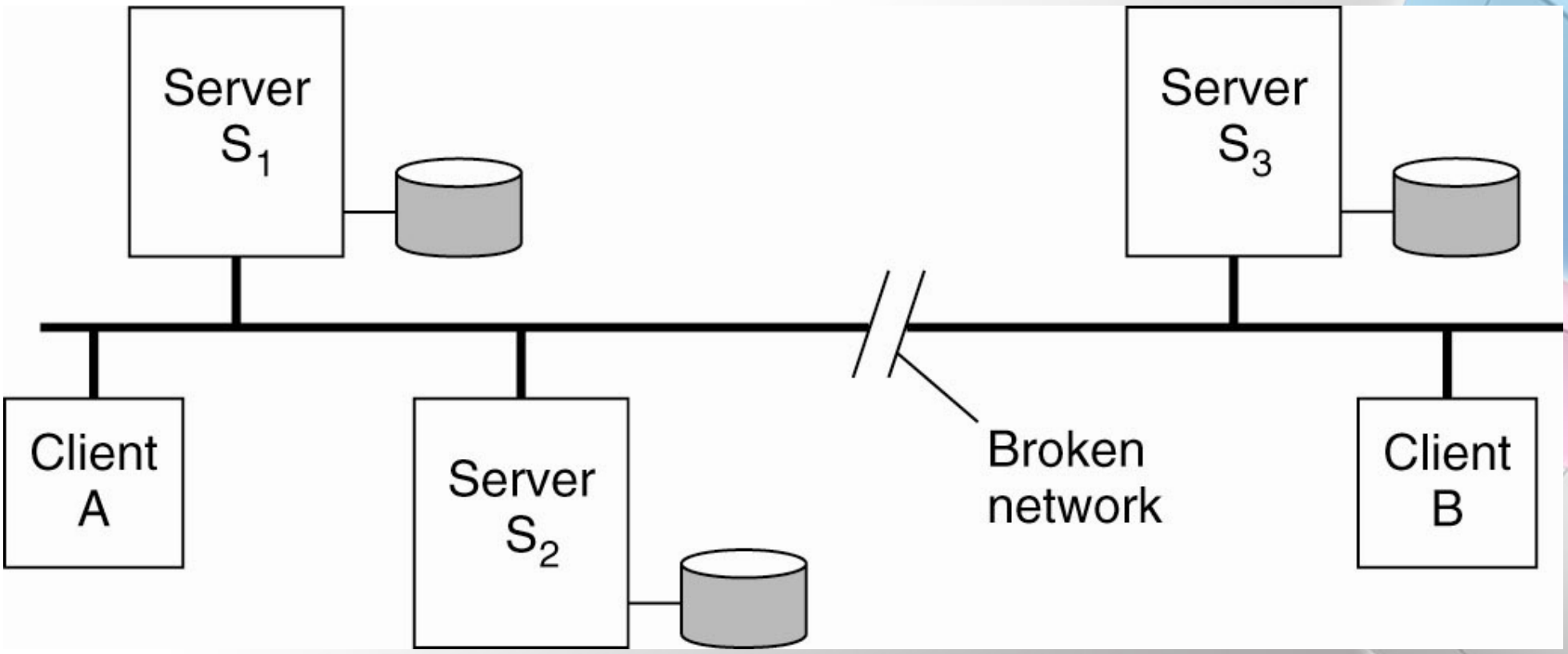


Figure 11-24. Two clients with a different Accessible Volume Storage Group (AVSG) for the same replicated file.

Unstructured P2P systems

- Assuming that nodes in an unstructured P2P system can be instructed to hold copies of files, what is then the **best allocation** of file copies to nodes?
- One policy is to uniformly distribute *n* copies of each file across the entire network.
  - This policy ignores that different files may have different request rates, that is, that some files are more popular than others.
- As an alternative, another policy is to replicate files according to how often they are searched for:
  - the more popular a file is, the more replicas we create and distribute across the overlay.

Structured Peer-to-Peer Systems

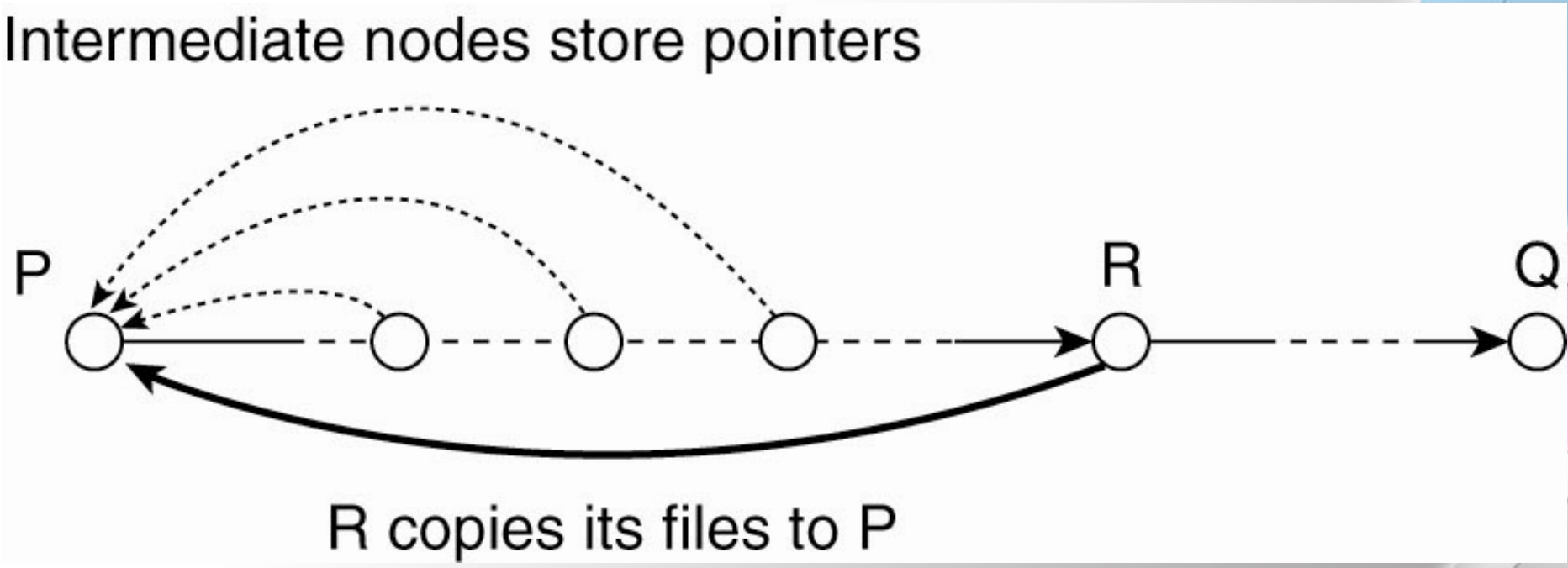
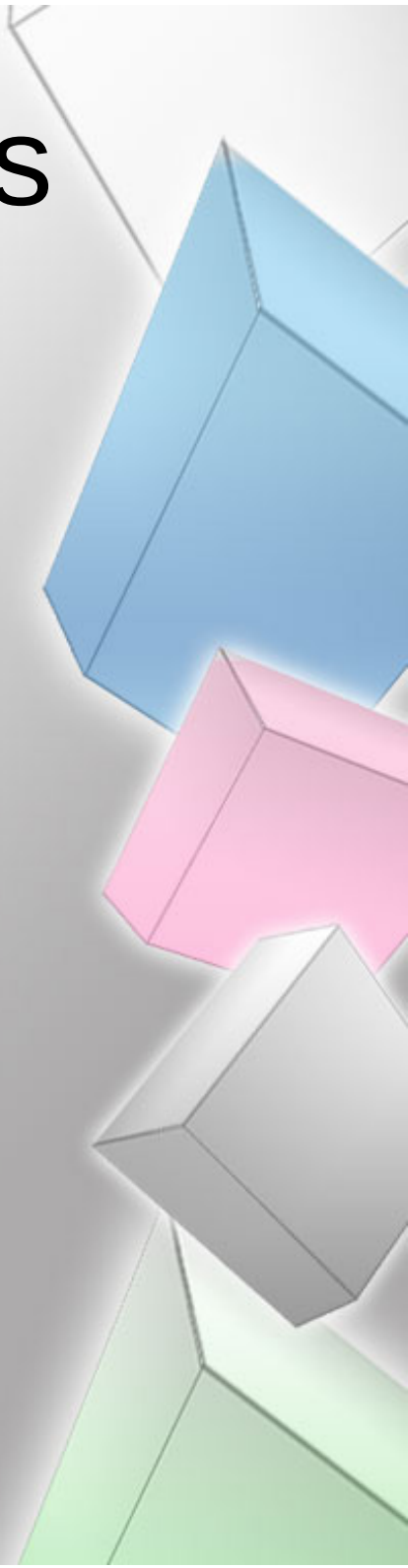


Figure 11-25. Balancing load in a peer-to-peer system by replication.

Distributed File Systems

- Architecture
- Processes
- Communication
- Naming
- Synchronization
- Consistency and Replication
- Security



Security in NFS

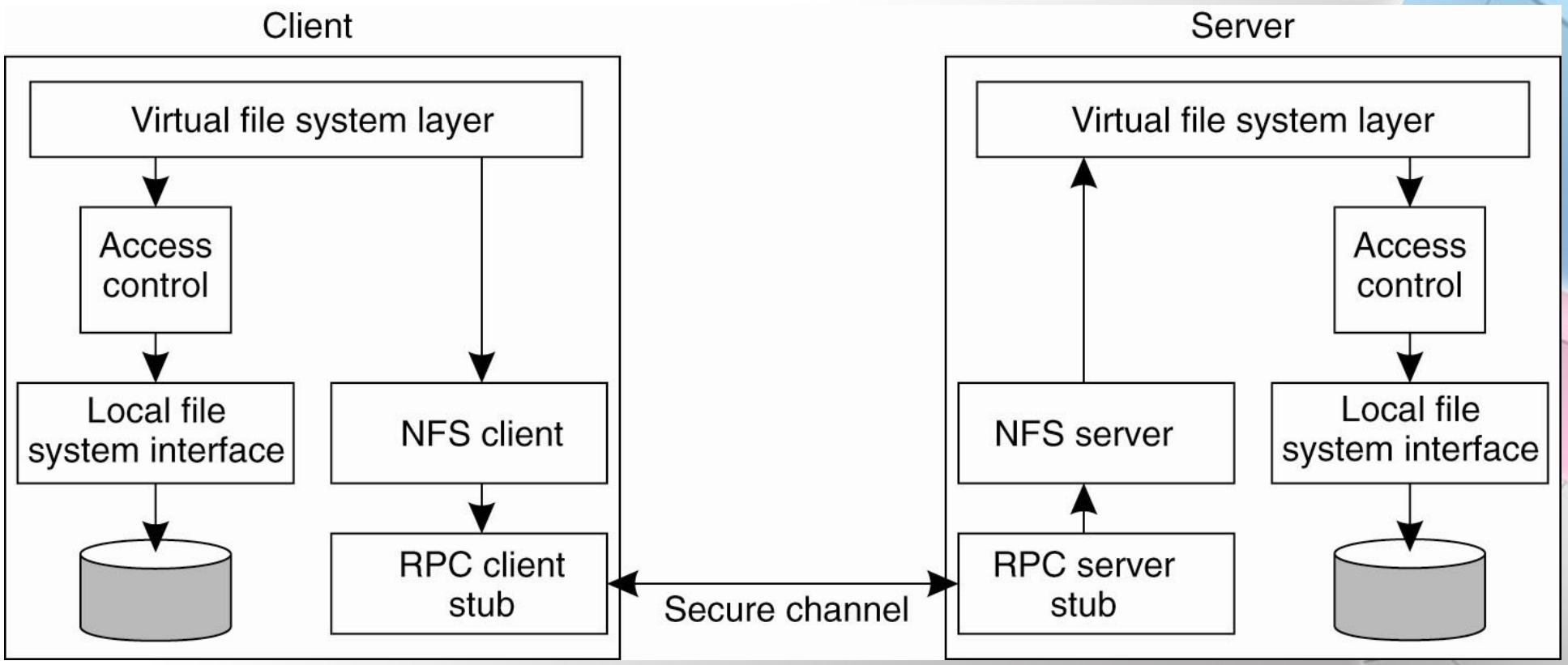


Figure 11-28. The NFS security architecture.

Secure RPCs

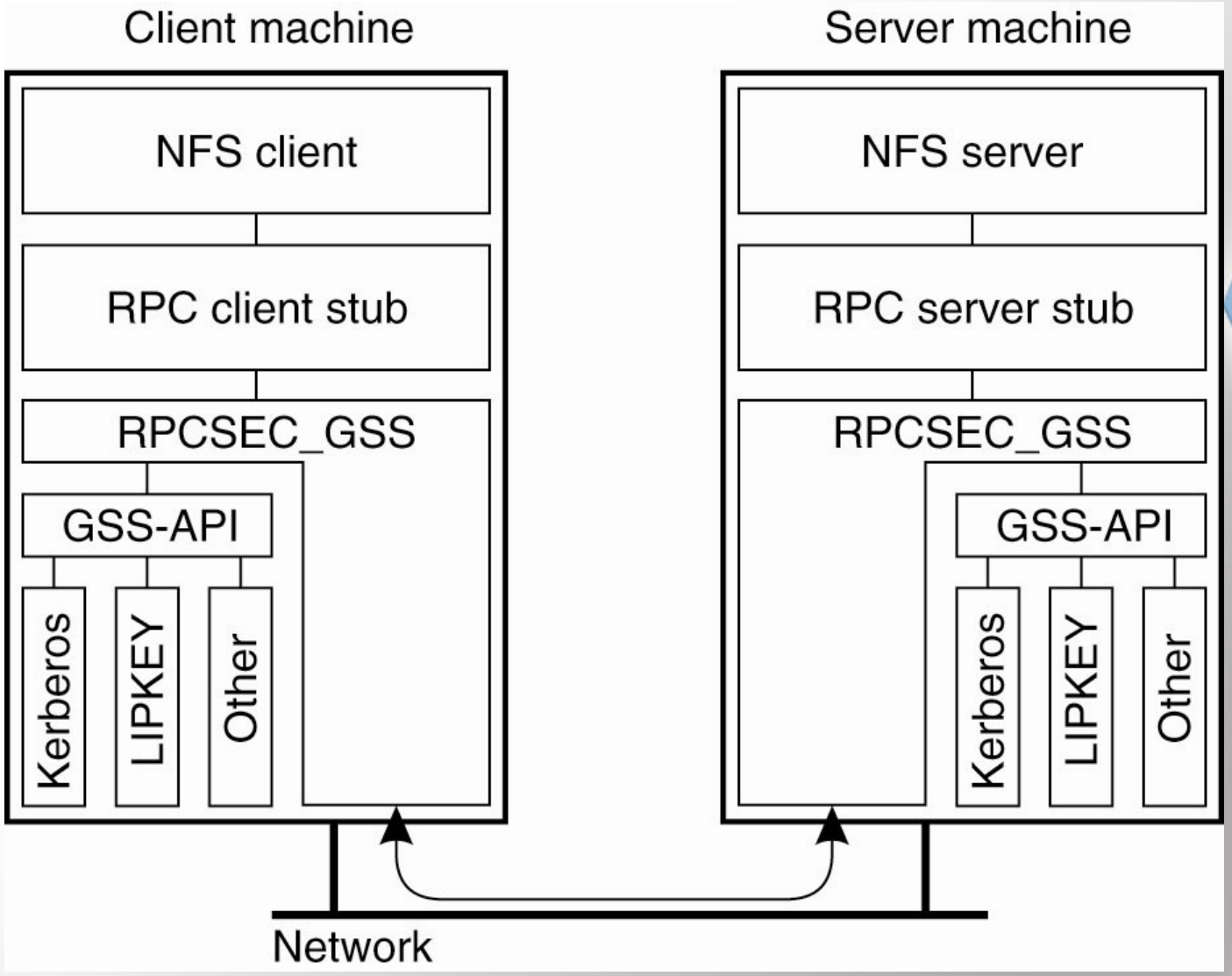


Figure 11-29. Secure RPC in NFSv4. RPCSEC_GSS is a general security framework that can support a myriad of security mechanisms.

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)

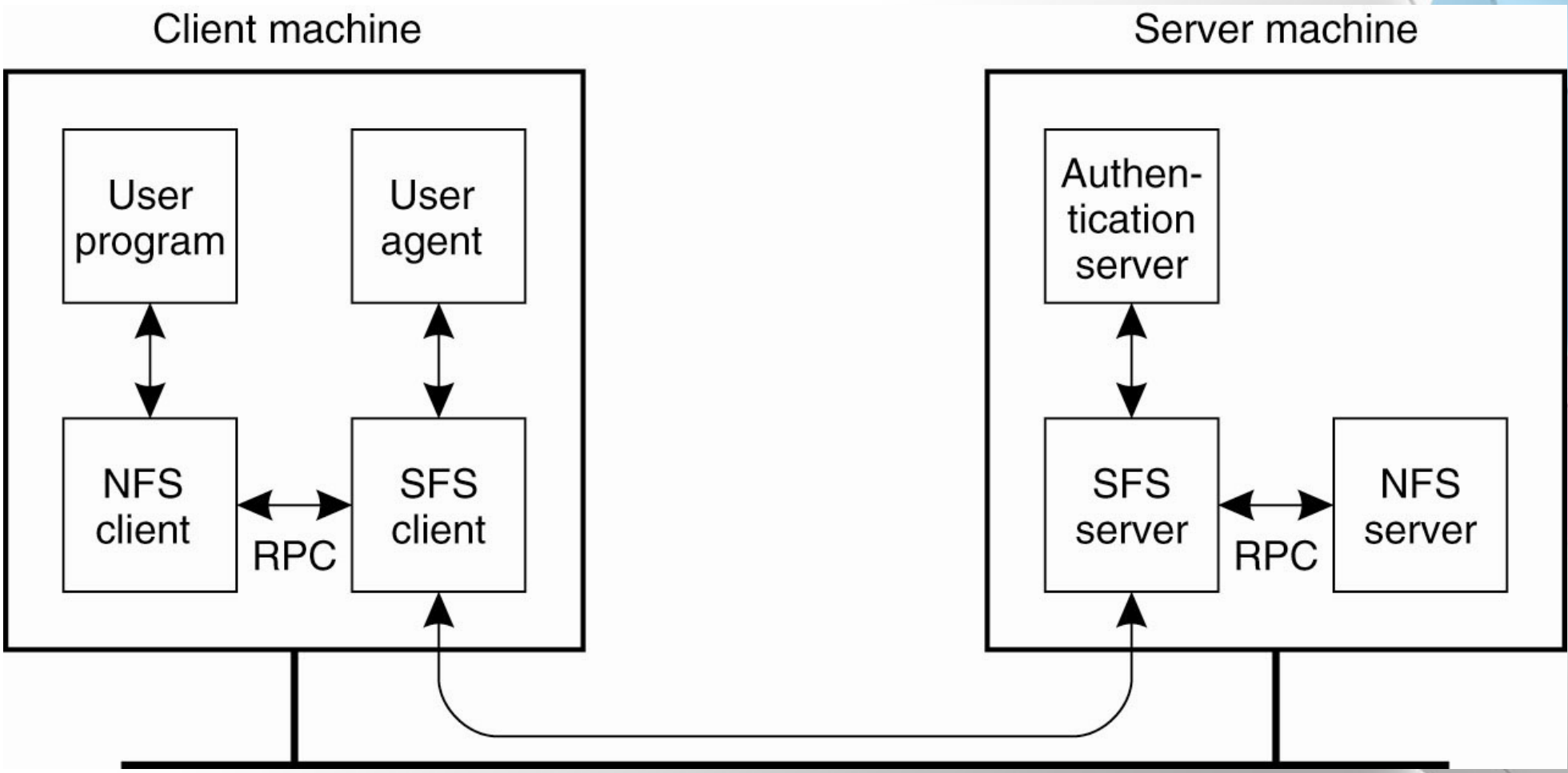


Figure 11-31. The organization of Secure File System (SFS).

Decentralized Authentication (2)

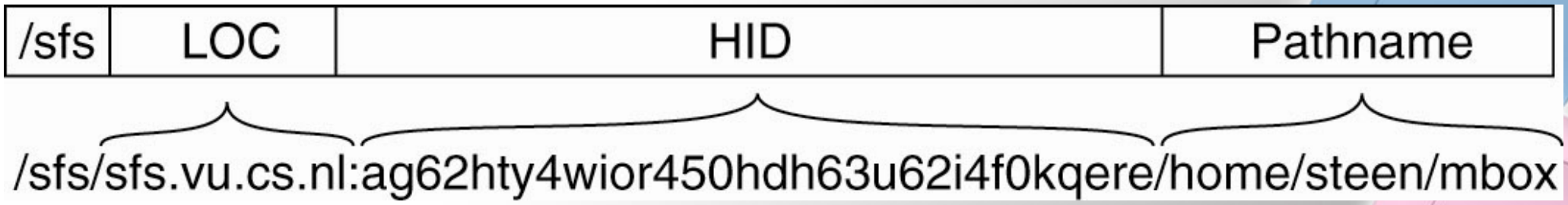
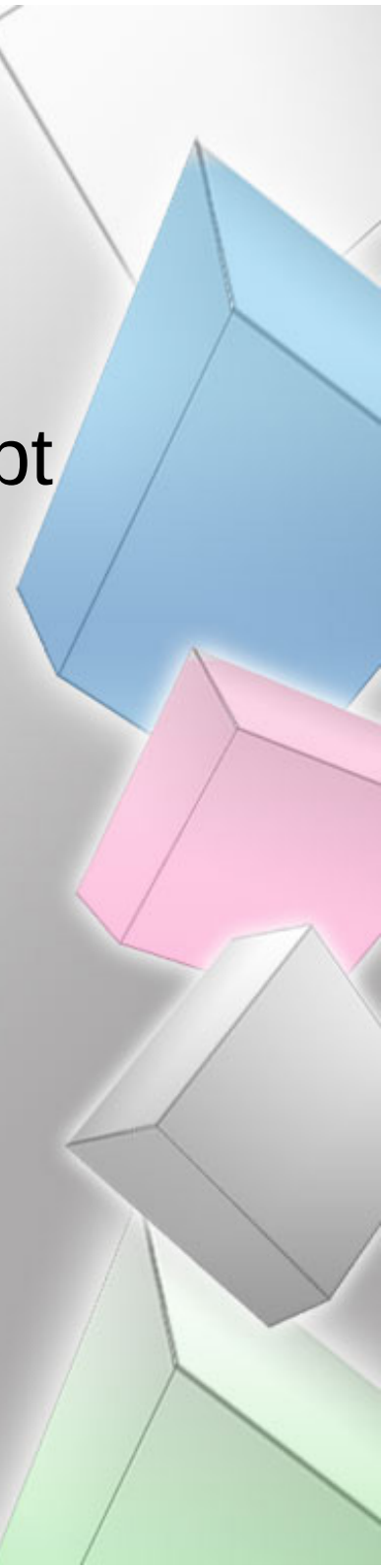


Figure 11-32. A self-certifying pathname in SFS. HID is a 32-digit number in base 32. HID is computed by taking a cryptographic hash H over the server's location and its public key.

End of Lesson 11

- Readings
  - Distributed Systems, Chapter 10, Except 11.4.2, 11.7 and 11.8.3.

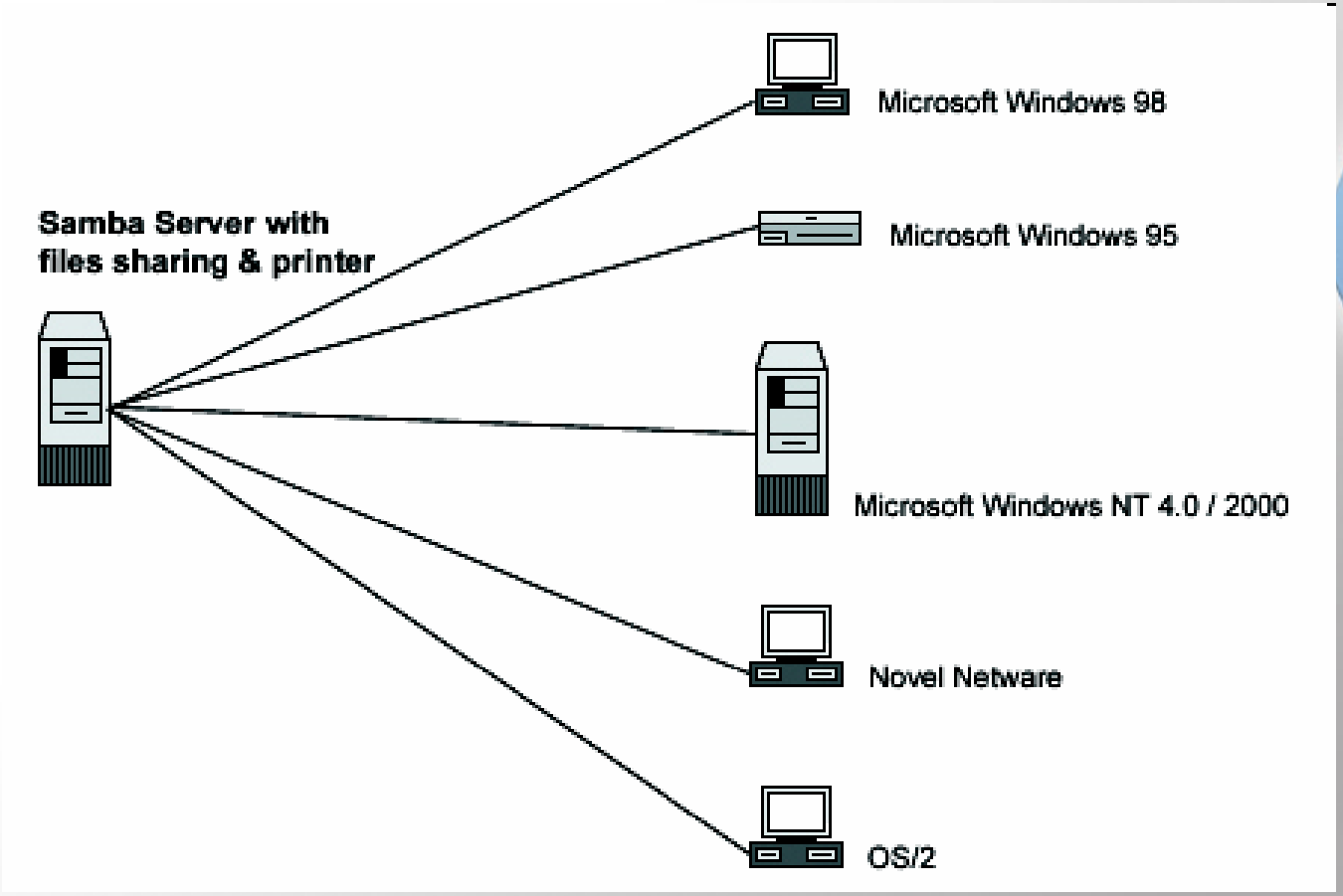


Samba

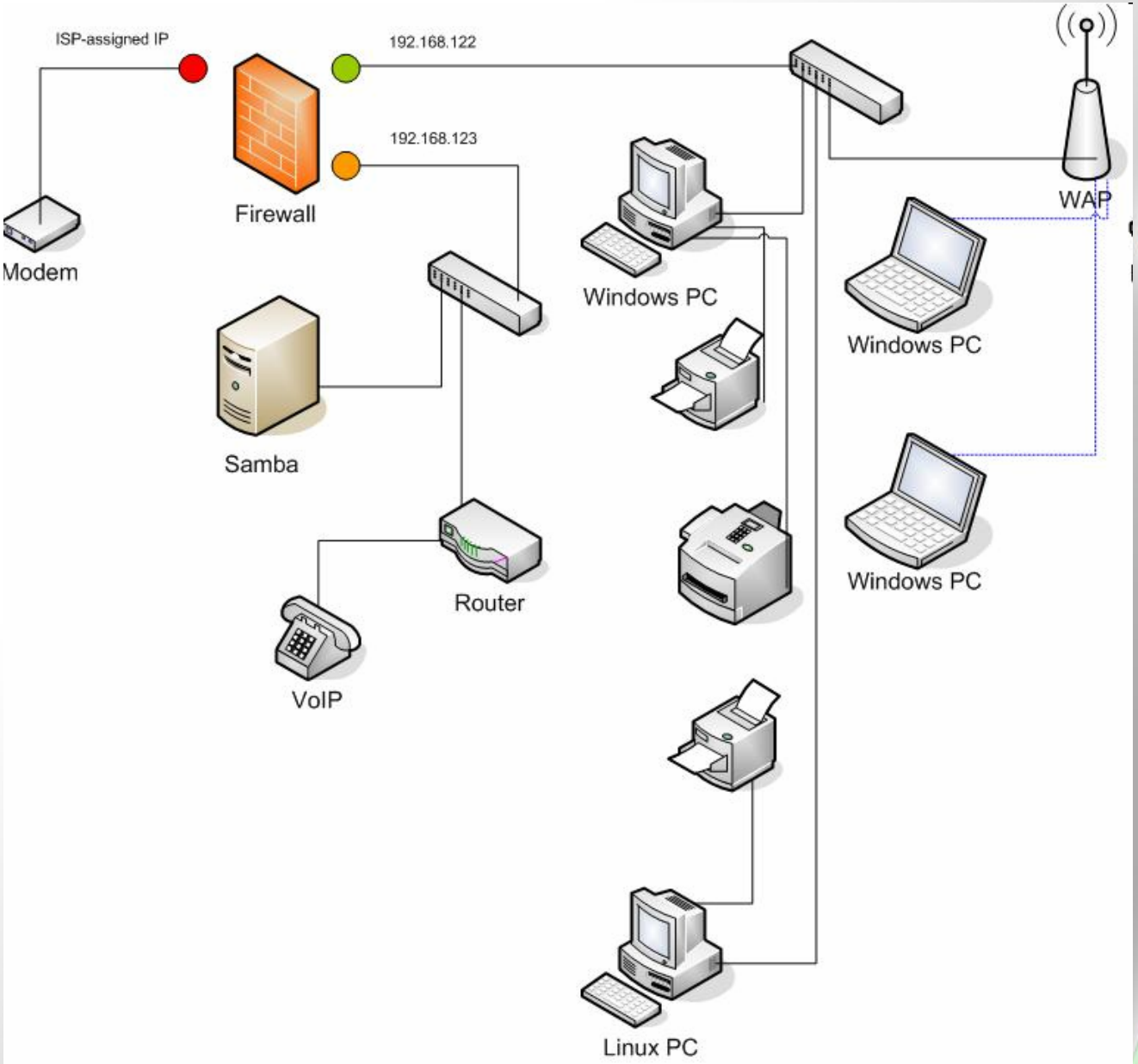
- Samba is software that can be run on a platform other than Microsoft Windows, for example, UNIX, Linux, IBM System 390, OpenVMS, and other operating systems.

- Samba uses the TCP/IP protocol that is installed on the host server. When correctly configured, it allows that host to interact with a Microsoft Windows client or server as if it is a Windows file and print server.

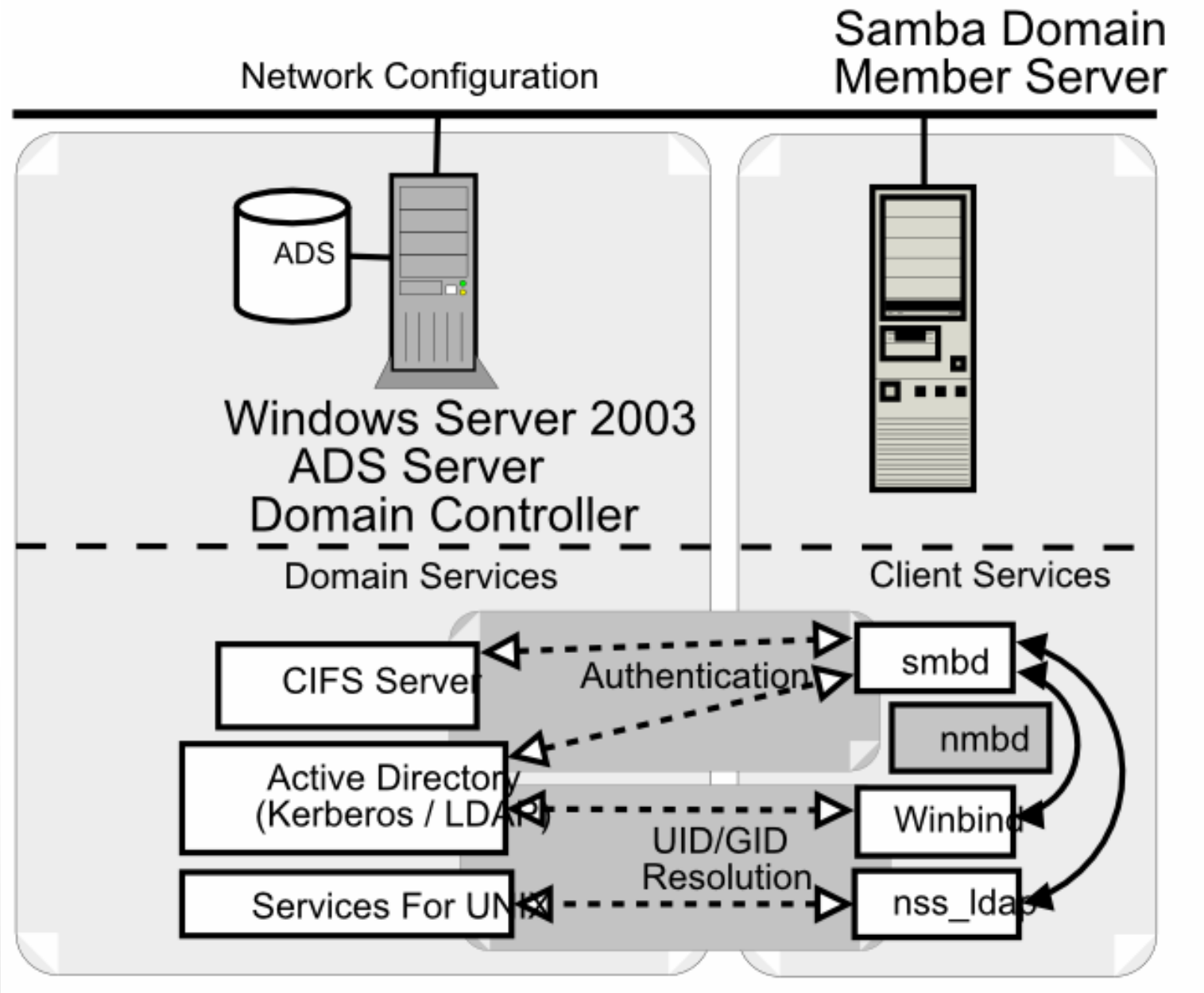
Samba Server



Windows systems have access to Linux file system



Samba and Active Directory



Project Proposal 5

- Set up and configure the Samba Server under Linux.