NETWORK FILE SYSTEMS

U.SHANMUGARAJAN
I – M.Sc [Comp. Science]

CONTENTS

Topics

- What is Distributed File systems
- NFS : overview
- Communication in NFS
- Processes in NFS
- Naming in NFS
- Synchronization in NFS
- Catching and Replication
- Fault tolerance
- Secutiy

WHAT IS DISTRIBUTED FILE SYSTEM

- A **distributed file system** (**DFS**) is a **file system** with data stored on a server.
- The data is accessed and processed as if it was stored on the local client machine.
- The **DFS** makes it convenient to share information and **files** among users on a network in a controlled and authorized way.

EXAMPLES

NFS
CODA FILE SYSTEM
AFS
REDHAT GFS
Windows 2000

OVERVIEW OF NFS

Network File System (**NFS**) is a distributed **file system** protocol originally developed by Sun Microsystems in 1984.

The Network File System (NFS) allows users to access files and directories located on remote computers and treat those files and directories as if they were local.

Mostly used with UNIX OS and also Implement for other OS like Mac OS, Microsoft Windows.

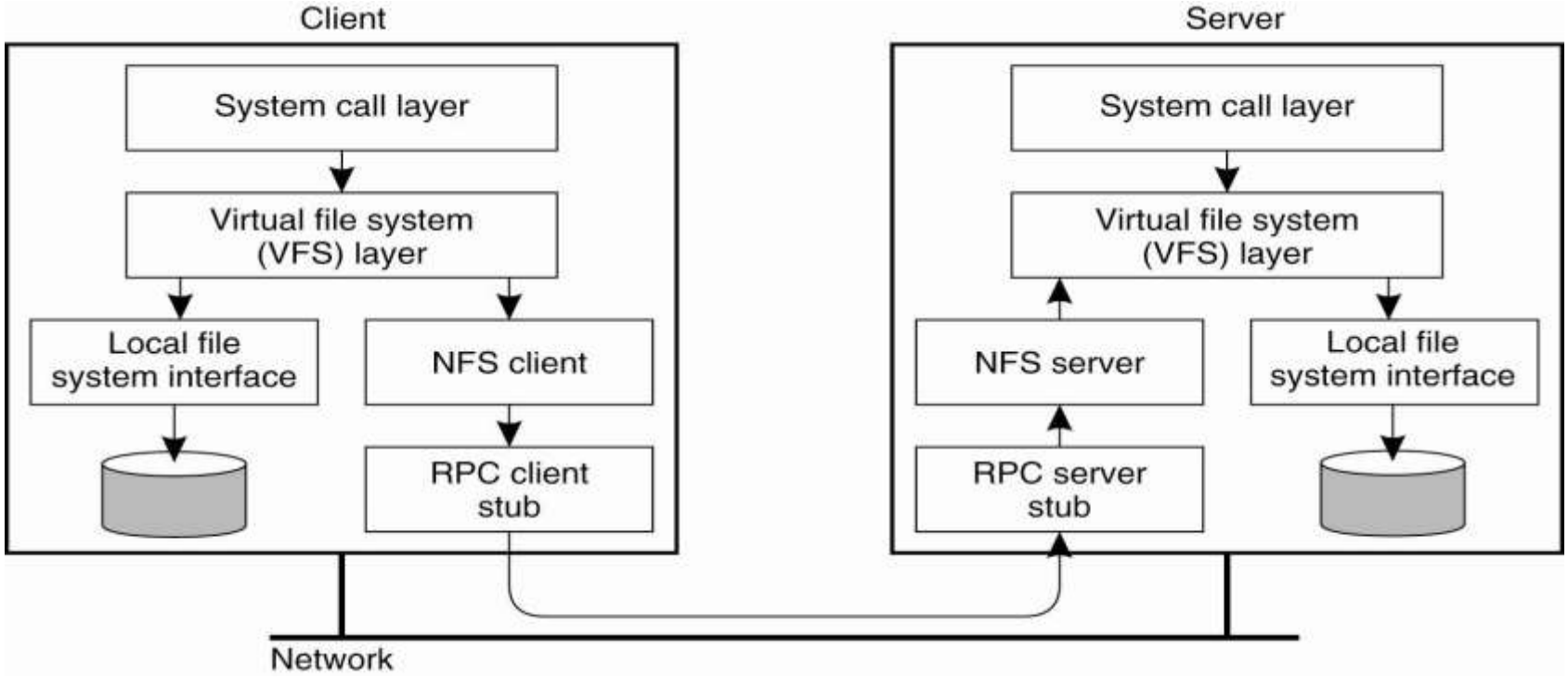
For example, users can use operating system commands to create, remove, read, write, and set file attributes for remote files and directories.

NFS VERSIONS

- The first version of NFS was never released and was kept internal to Sun micro systems.
- NFS v2 (released in March 1989) originally operated only over User Datagram Protocol(UDP). Its designers meant to keep the server side stateless, with locking.
- NFS v3 released in 1995 and it supports asynchronous writes on the server, to improve write performance.
- NFS v4 released in 2003 and it concentrates on strong security, and introduces a state full server.
- NFS version 4.1 (January 2010) aims to provide scalable parallel access to files distributed among multiple servers
- NFS version 4.2 is currently being developed.

NFS ARCHITECTURE

- NFS is implemented using the Virtual File System abstraction, which is now used for lots of different operating systems.
- VFS provides standard file system interface, and allows to hide difference between accessing local or remote file system.



FILE SYSTEM MODEL

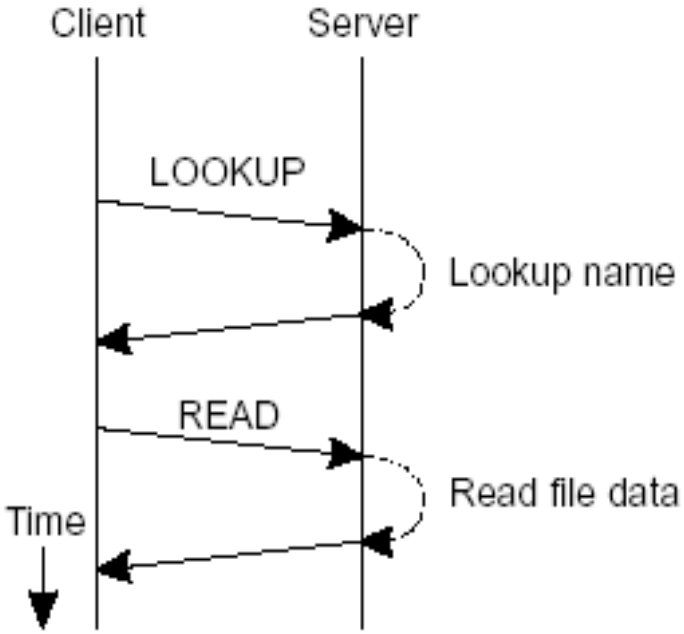
- The file system model offered by NFS is almost same as the one offered in UNIX-Based Systems.
- Files are treated as sequences of bytes.
- to access a file, a client must first look up its name in a naming service and obtain the associated file handle.

NFS FILE OPERATIONS

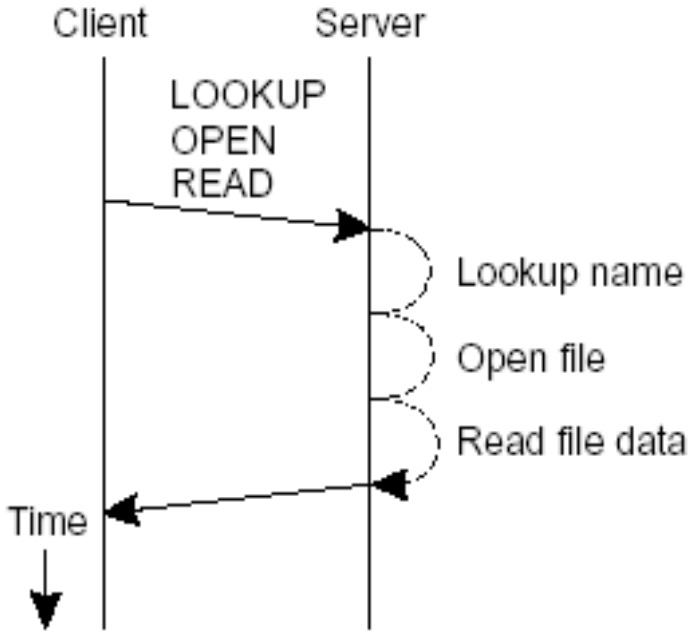
Oper.	v3	v4	Description
Create	Yes	No	Create a regular file
Create	No	Yes	Create a nonregular file
Link	Yes	Yes	Create a hard link to a file
Symlink	Yes	No	Create a symbolic link to a file
Mkdir	Yes	No	Create a subdirectory
Mknod	Yes	No	Create a special file
Rename	Yes	Yes	Change the name of a file
Remove	Yes	Yes	Remove a file from a file system
Rmdir	Yes	No	Remove an empty subdirectory
Open	No	Yes	Open a file
Close	No	Yes	Close a file
Lookup	Yes	Yes	Look up a file by means of a name
Readdir	Yes	Yes	Read the entries in a directory
Readlink	Yes	Yes	Read the path name in a symbolic link
Getattr	Yes	Yes	Get the attribute values for a file
Setattr	Yes	Yes	Set one or more file-attribute values
Read	Yes	Yes	Read the data contained in a file
Write	Yes	Yes	Write data to a file

COMMUNICATION IN NFS

- All communication is based on the Open Network Computing RPC (ONC RPC). Version 4 also supports **compound procedures** by which several RPCs can be grouped into a single request.



(a)



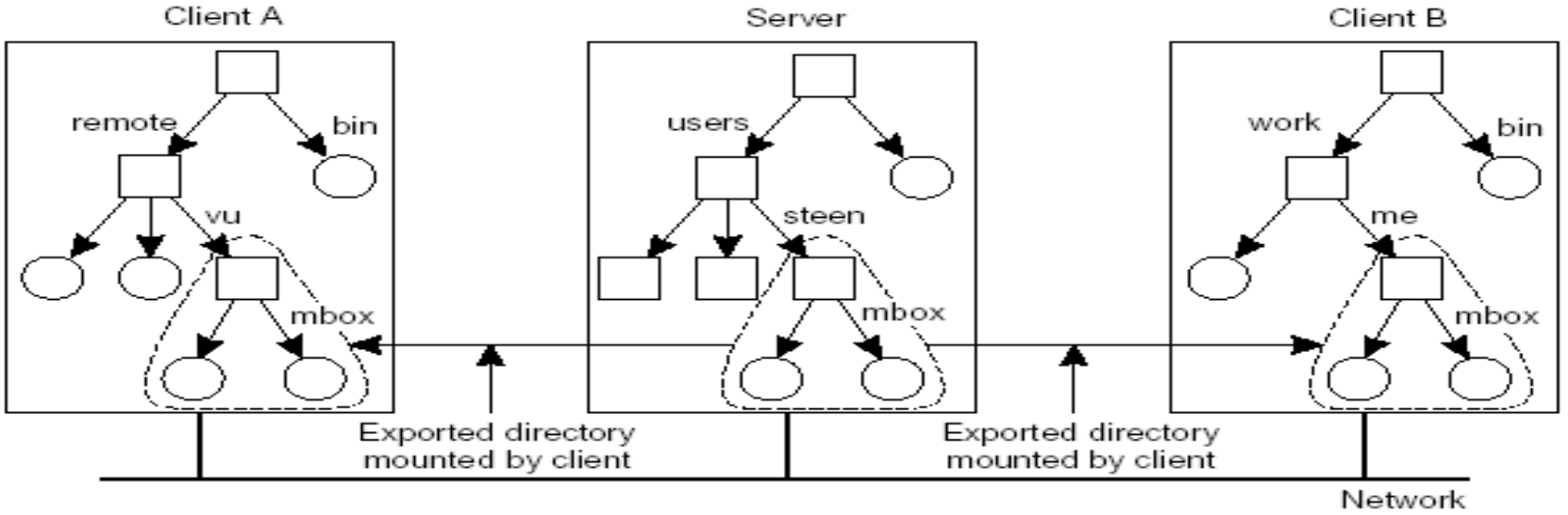
(b)

PROCESSES IN NFS

- NFS is a traditional client server system in which clients request a file server to perform operations on files.
- NFS servers are stateless (i.e.,) NFS protocol did not require that servers maintained any client's state.
- The main advantage of the stateless approach is simplicity.
- In NFS v4 , stateful servers are used.

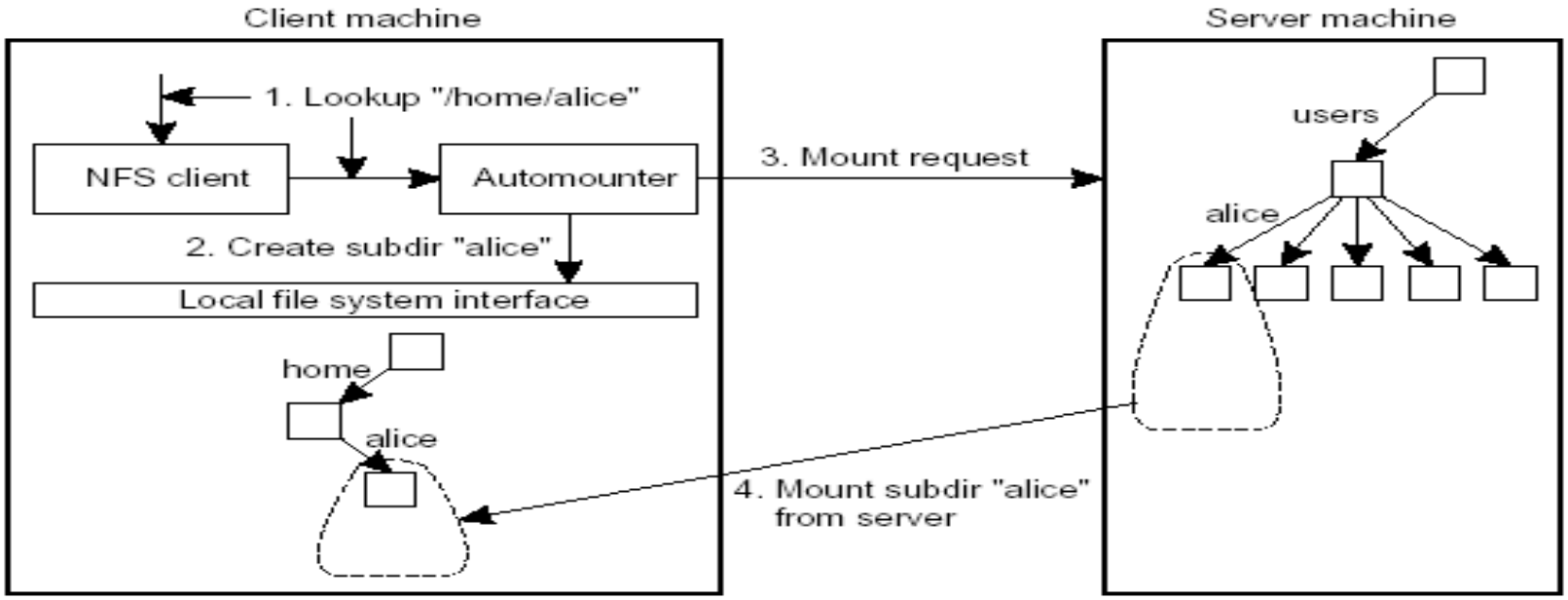
NAMING IN NFS

- NFS provides support for mounting remote file systems (and even directories) into a client's local name space.
- Instead of mounting an entire file system, NFS allows clients to mount only part of a file system.
- A server is said to **export a directory** when it makes that directory and its entries available to clients.



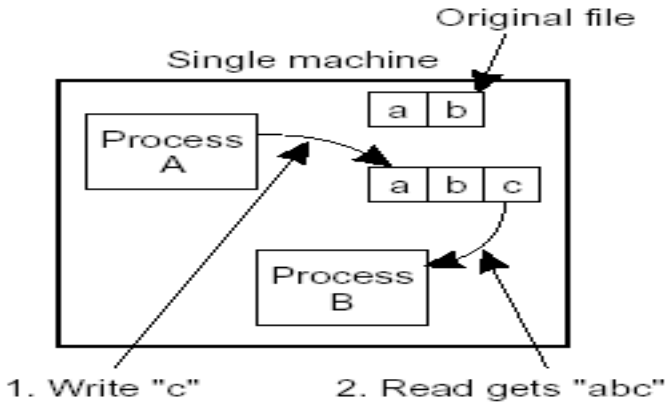
AUTOMOUNTING IN NFS

- **Problem:** To share files, we partly standardize local name spaces and mount shared directories. Mounting very large directories (e.g., all subdirectories in home/users) takes a lot of time
- **Solution: automounting -** The automounter maintains a table of mount points (pathnames) with a reference to one or more NFS servers.

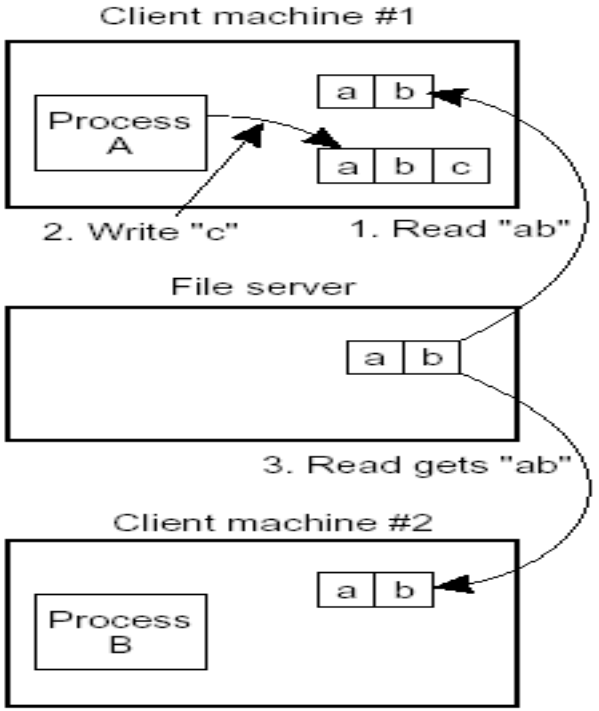


FILE SHARING SEMANTICS

- when two or more users share the same file ,it is necessary to define semantics of reading and writing precisely to avoid problems.



(a)



(b)

FILE SHARING SEMANTICS (CONT.,)

UNIX semantics: a read operation returns the effect of the last write operation ⇨ Every operation on a file is instantly visible to all processes.

Transaction semantics: the file system supports transactions on a single file ⇨ issue is how to allow concurrent access.

Session semantics: the effects of read and write operations are seen only to the client that has opened (a local copy) of the file ⇨No changes are visible to other processes until the file is closed.

FILE LOCKING IN NFS

- in NFS, file locking has been handled by means of a separate protocol, implemented by a lock manager.

  LOCK,LOCKT,LOCKU

  Locks are granted for a specific time and we use RENEW fo extend the lock.

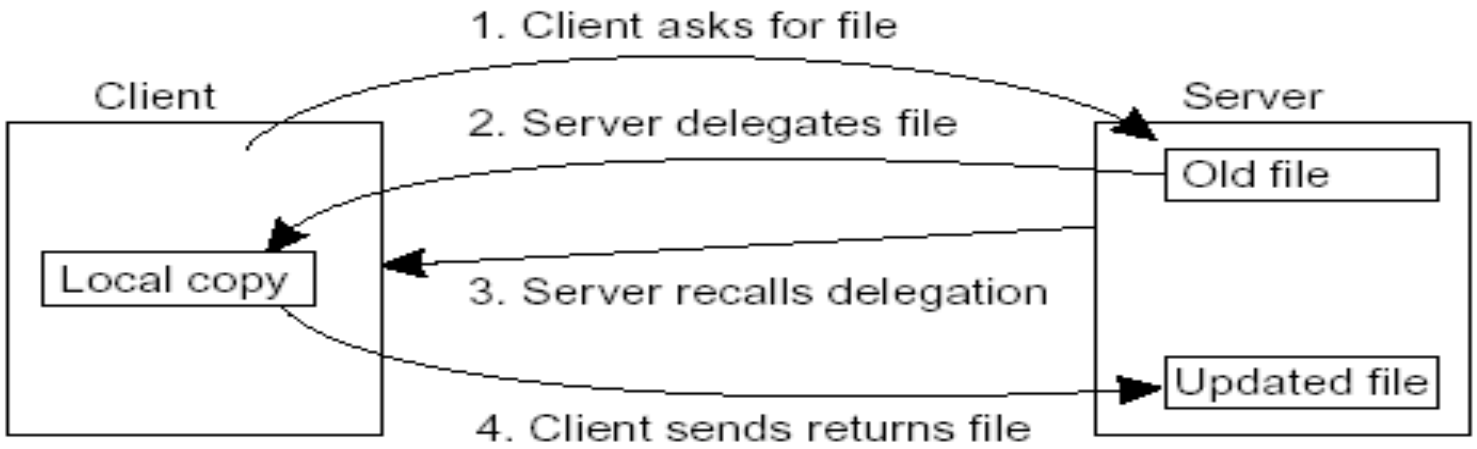
- it also an implicit **share reservation** approach(in windows system).
- In NFS v4 ,locking integrated into file access protocol.
- When a client opens a file, it specifies the type of access it requires (READ,WRITE,BOTH) and which type of access the server deny other clients (NONE,READ,WRITE,BOTH).

CACHING & REPLICATION

- caching refers to the ability to access data from within the distributed system itself instead of relying on a separate system of record.
- On client *open*(), client asks server if its cached attribute blocks are up to date. Once file is *open*, different client processes can write it and get inconsistent data.
- Modified data is flushed back to the server every 30 seconds
- Data is said to be *replicated* when a distributed caching system proactively makes multiple copies of that data for achieving some combination of availability.

CACHING & REPLICATION (CONT.,)

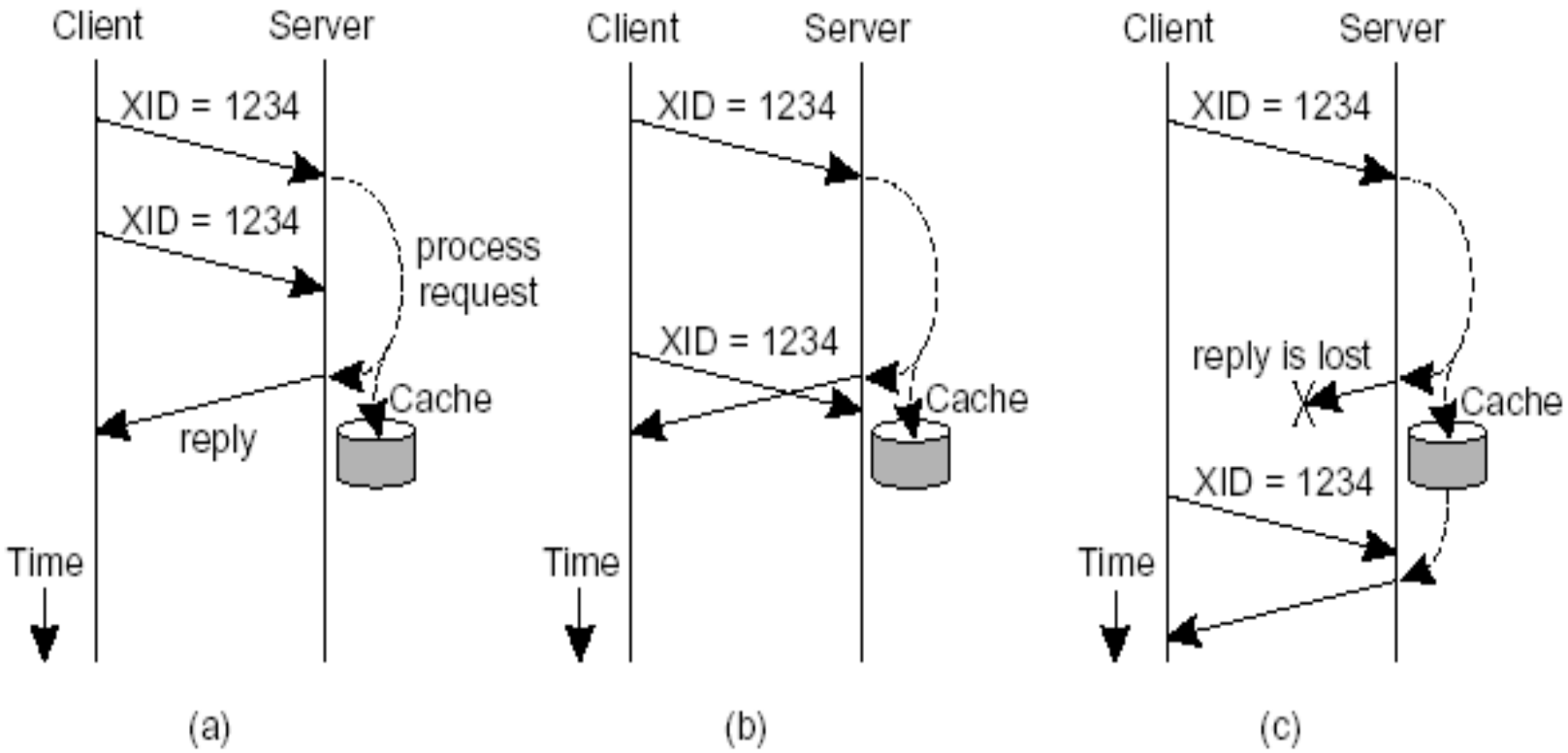
- **Open delegation:** Server will explicitly permit a client machine to handle local operations from other clients on that machine.
- An important consequence of delegating a file to a client is that the server need to be able to recall the delegation.



FAULT TOLERANCE

- Until v4, fault tolerance was easy due to the stateless servers. Now, problems come from the use of an unreliable RPC mechanism, but also stateful servers that have delegated matters to clients.
- **RPC FAILURES –** when reply is lost, retransmission may trigger multiple invocations of request.
- The above problem solved with **duplicate-request cache** and **transaction identifier.**
- **Locking/Open delegation:** Essentially, recovered server offers clients a grace period to **reclaim** locks. When period is over, the server starts its normal local manager function again.

FAULT TOLERANCE (CONT.,)



SECURITY

Secure RPC

there are three ways for authentication

1.System authentication

passes its user ID and group ID to server.

2.Diffe-Helman key exchange

establish session key .

3.kerberos

works on the basis of tickets

RPCSEC GSS

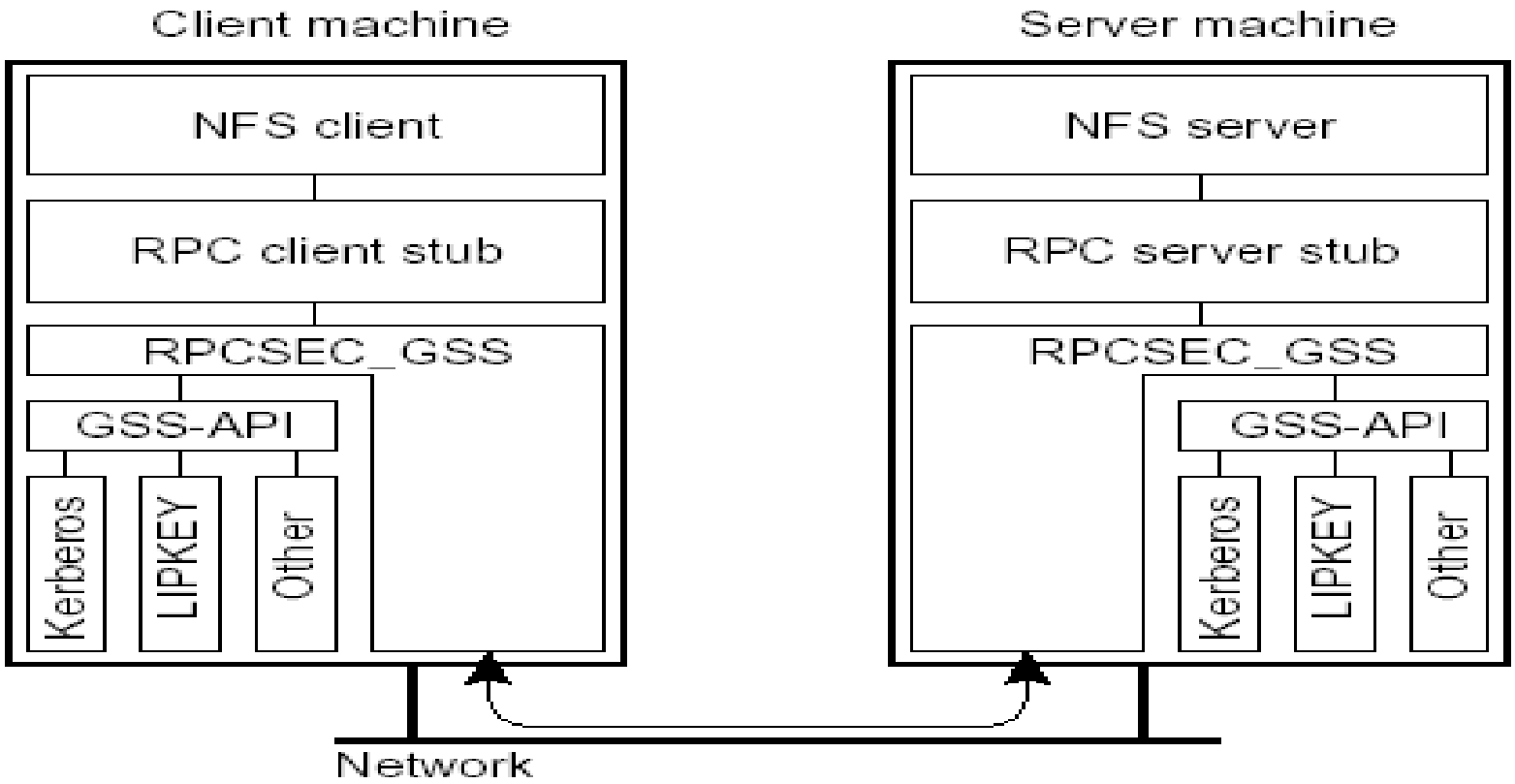
A standard interface that allows integration with existing security services

Access Control

secure RPC provides the mechanisms for security but does not specify any particular policy.

maintains ACL file attribute.

SECURITY


Client machine
Server machine
NFS client
RPC client stub
RPCSEC_GSS
GSS-API
Kerberos
LIPKEY
Other
NFS server
RPC server stub
RPCSEC_GSS
GSS-API
Kerberos
LIPKEY
Other
Network

THANK YOU