Distributed Systems
Distributed File System

[2] Distributed File System

1. Sun Network File System
2. The Coda File System
3. Plan 9: Resources Unified to Files

[3] Network File System (NFS)

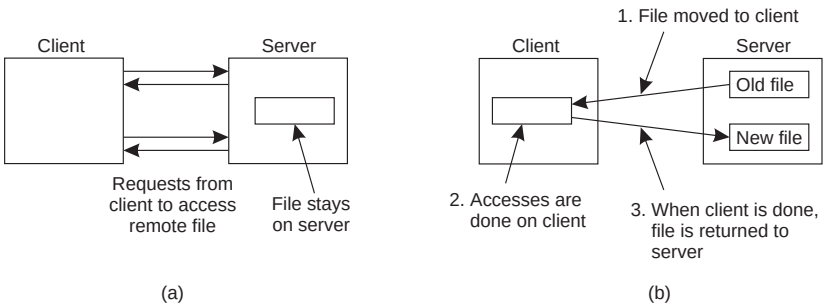
NFS, basic idea: each file server provides a standardized view of its local files system,

History of NFS:

- the 1st version internal to Sun,
- the 2nd version incorporated into SunOS 2.0,
- the 3rd (current) version – now undergoing major revisions.

NFS – not so much a true file system but a collection of protocols.

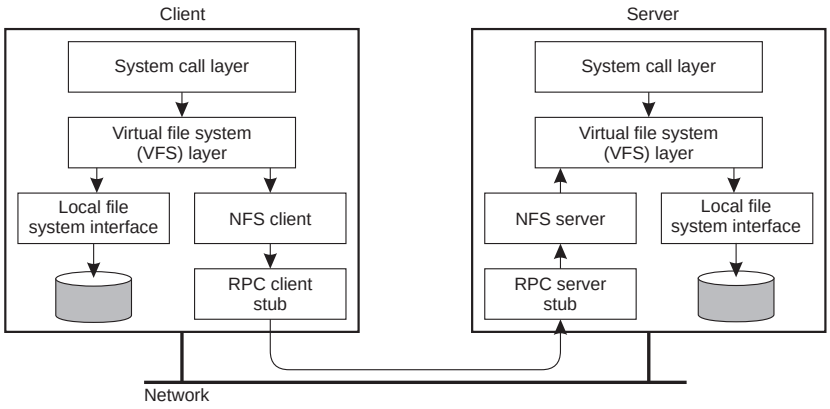
[4] NFS Architecture (1)



a. the remote access model,

b. the upload/download model.

[5] **NFS Architecture (2)**



The basic NFS architecture for UNIX systems.

[6] **NFS Features**

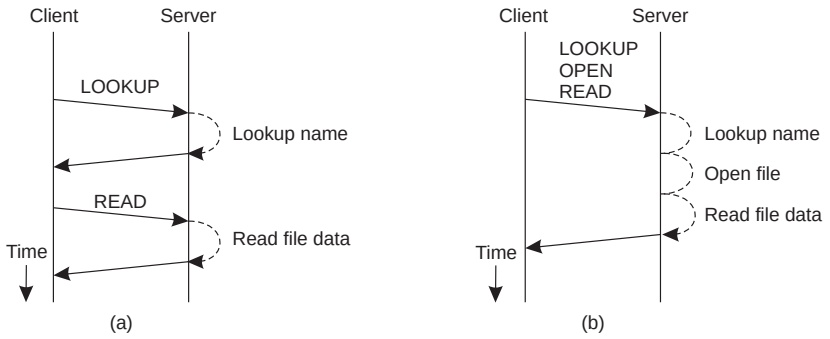
- NFS largely independent of local file system,
- supports hard and symbolic links,
- files named, accessed by means of Unix-like file handles,
- **version 4**
  - *create* used for creating non-regular files,
  - regular files created by *open*,
  - server generally maintains state between operations on the same file,
  - *lookup* attempts to resolve the entire name, also if it means crossing mount points,
  - supports *compound procedures*.

[7] **File System Model**

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
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

An incomplete list of file system operations supported by NFS.

[8] **Communication**



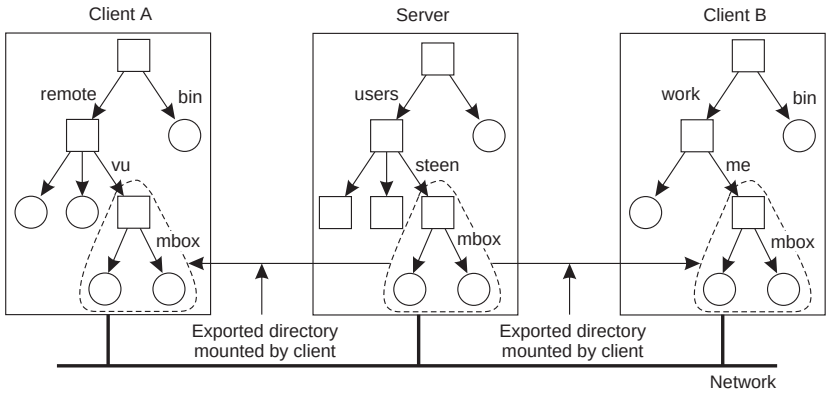
a. Reading data from a file in NFS version 3.

b. Reading data using a compound procedure in version 4.

[9] **Stateless vs. Stateful Server**

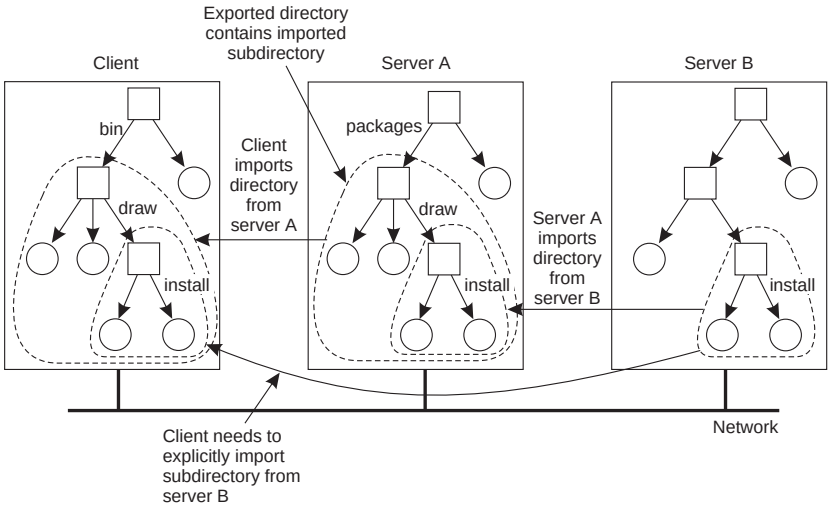
- NFS version 3:
  - simplicity as the main advantage of the stateless approach,
  - locking a file cannot be easily done,
  - certain authentication protocols require maintaining state of clients.
- NFS version 4:
  - expected to work across wide area network,
  - clients can make effective use of caches requiring cache consistency protocol,
  - support for callback procedures by which a server can do an RPC to a client.

[10] **NFS - Naming (1)**



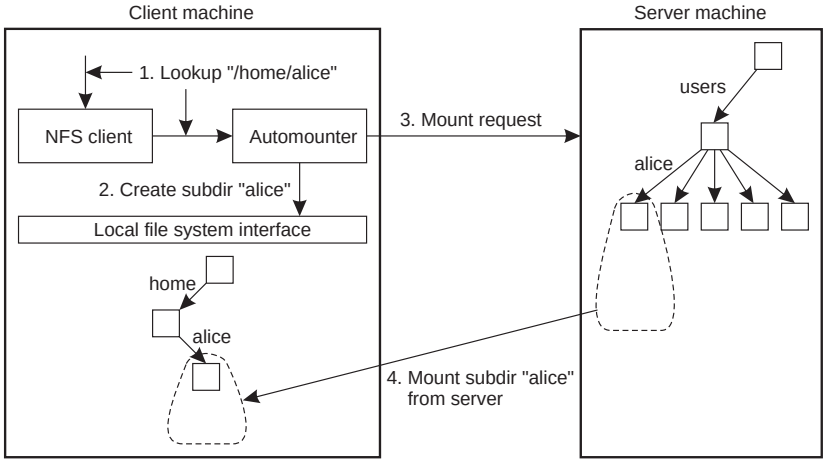
Mounting (part of) a remote file system in NFS.

[11] **NFS - Naming (2)**



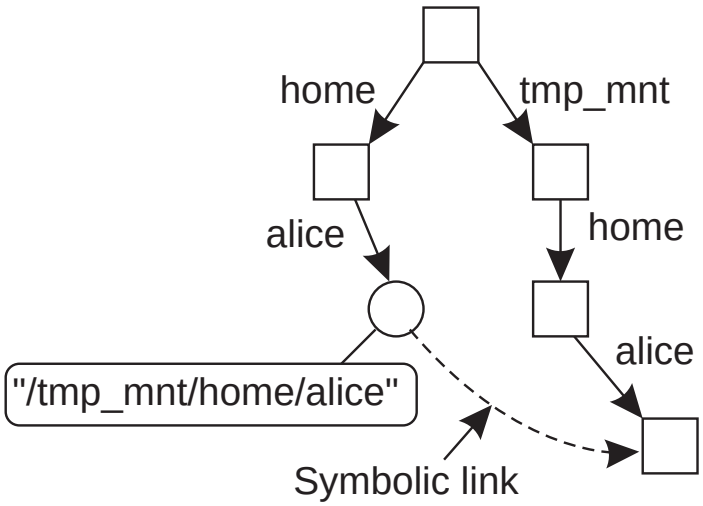
Mounting nested directories from multiple servers in NFS.

[12] **Automounting (1)**



A simple automounter for NFS.

[13] **Automounting (2)**



Using symbolic links with automounting.

Whenever command `ls -l /home/alice` is executed, the NFS server is contacted directly without involvement of the automounter.

[14] **File Attributes**

Attribute	Description
TYPE	The type of the file (regular, directory, symbolic link)
SIZE	The length of the file in bytes
CHANGE	Indicator for a client to see if and/or when the file has changed
FSID	Server-unique identifier of the file's file system

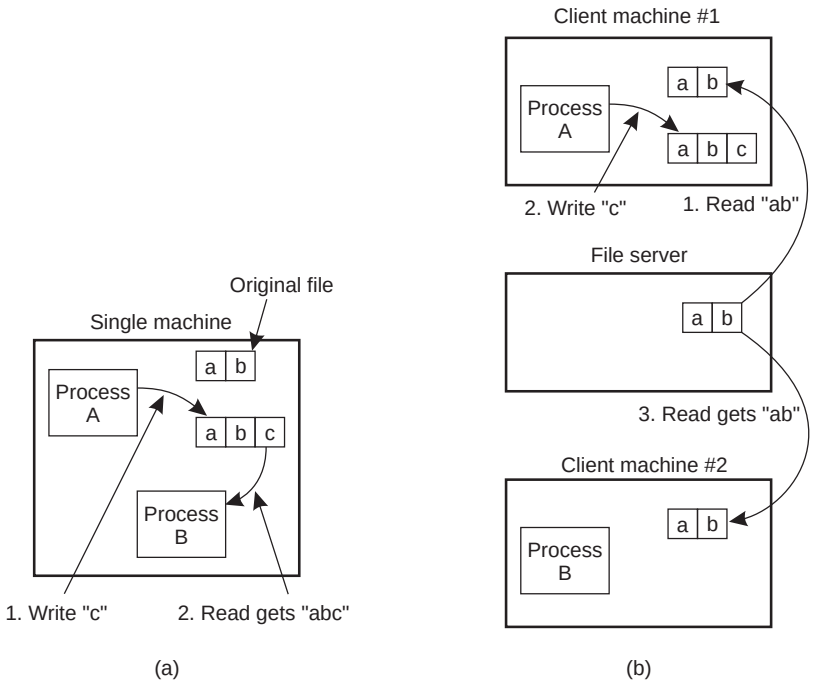
(a)

Attribute	Description
ACL	An access control list associated with the file
FILEHANDLE	The server-provided file handle of this file
FILEID	A file-system unique identifier for this file
FS_LOCATIONS	Locations in the network where this file system may be found
OWNER	The character-string name of the file's owner
TIME_ACCESS	Time when the file data were last accessed
TIME_MODIFY	Time when the file data were last modified
TIME_CREATE	Time when the file was created

(b)

Some general **mandatory** (a) and **recommended** (b) file attributes in NFS. Moreover one may have **named** attributes – an array of pairs *(attribute, value)*.

[15] **Semantics of File Sharing (1)**



- On a single processor, when a read follows a write, the value returned by the read is the value just written.
- In a distributed system with caching, obsolete values may be returned.

[16] **Semantics of File Sharing (2)**

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

Four ways of dealing with the shared files in a distributed system.

- NFS implements session semantics.

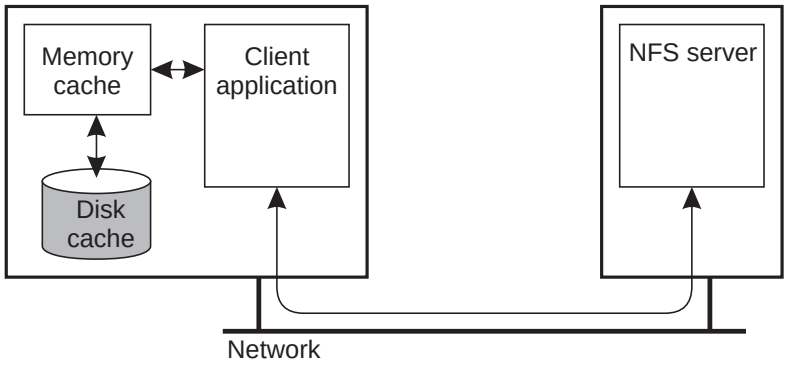
[17] **File Locking in NFS**

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

NFS version 4 operations related to file locking.

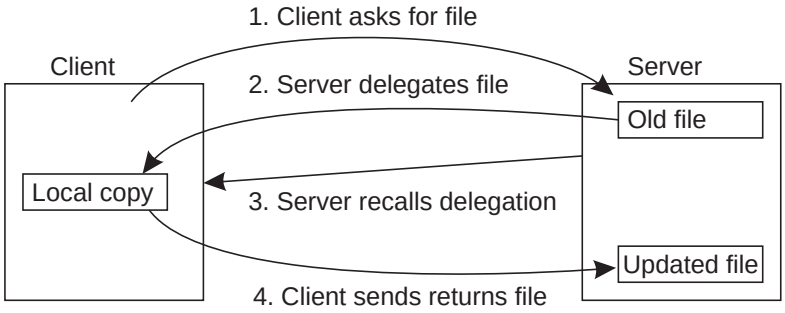
- v4: file locking integrated into file access protocol,
- lock failed ⇒
  - error message and polling **or**
  - client can request to be put on a FIFO-ordered list maintained by the server (and still polling).

[18] **Client Caching (1)**



Client-side caching in NFS.

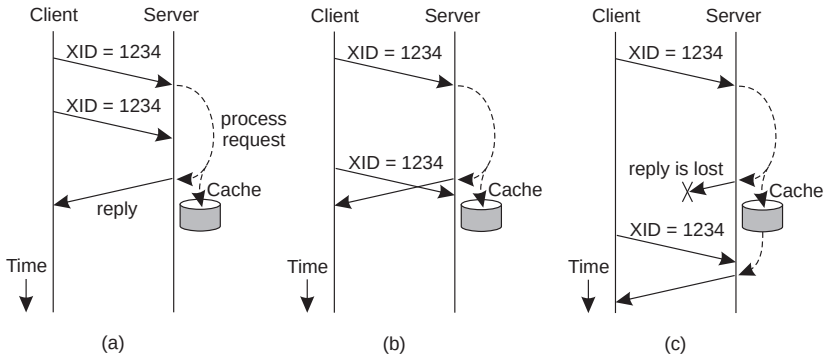
[19] **Client Caching (2)**



Using the NFS version 4 callback mechanism to recall file delegation.

- **open delegation** takes place when the client machine is allowed to locally handle open and close operations from other clients on the same machine,
- recalling delegation requires **callback** support,
- NFS uses **leases** on cached attributes, file handles and directories.

[20] **RPC Failures**



Three situations for handling retransmissions (XID = transaction identifier).

a. the request is still in progress,

b. the reply has just been returned,

c. the reply has been some time ago, but was lost.

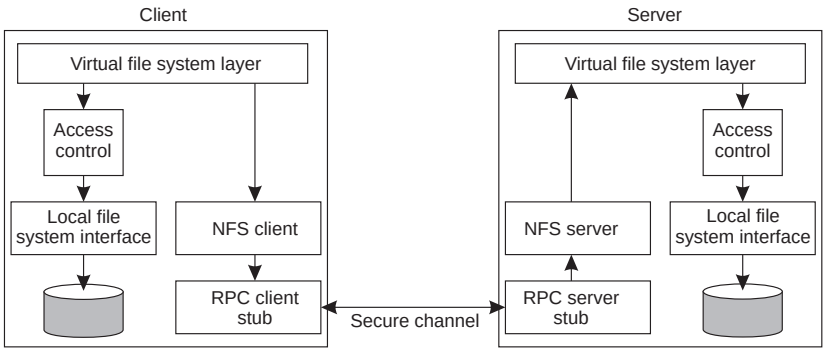
[21] **File Locking in the Presence of Failures**

Server crashes and subsequently recovers, than:

- **grace period**:
  - a client can reclaim locks that were previously granted to it,
  - normal lock requests may be refused until the grace period is over.

Notice: leasing requires synchronization of client's and server's clocks.

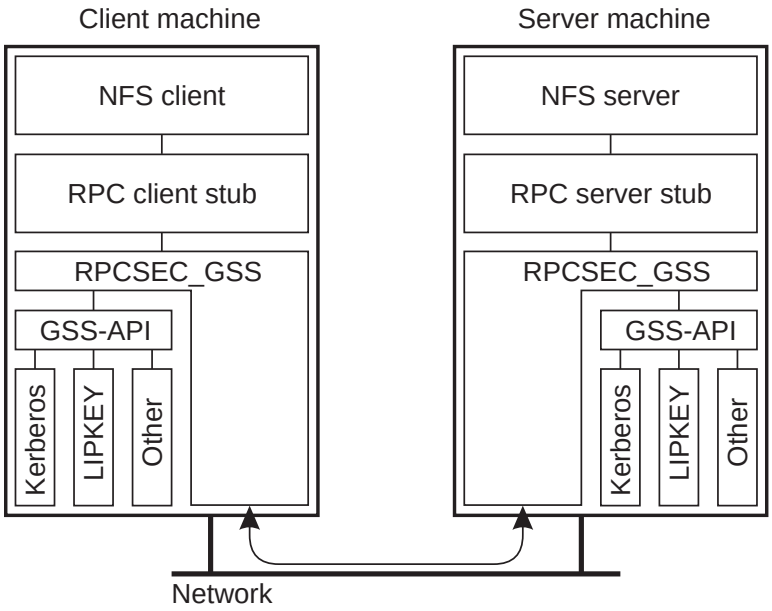
[22] **Security**



The NFS security architecture (version 3).

- system authentication,
- Diffie-Hellman key exchange (a public key cryptosystem), but only 192 bits in NFS,
- Kerberos.

[23] **Secure RPCs**



Secure RPC in NFS version 4 (GSS - general security framework):

- LIPKEY - a public key system,
- clients to be authenticated using passwords,
- servers can be authenticated using a public key.

[24] **Access Control**

Operation	Description
Read_data	Permission to read the data contained in a file
Write_data	Permission to modify a file's data
Append_data	Permission to append data to a file
Execute	Permission to execute a file
List_directory	Permission to list the contents of a directory
Add_file	Permission to add a new file to a directory
Add_subdirectory	Permission to create a subdirectory to a directory
Delete	Permission to delete a file
Delete_child	Permission to delete a file or directory within a directory
Read_acl	Permission to read the ACL
Write_acl	Permission to write the ACL
Read_attributes	The ability to read the other basic attributes of a file
Write_attributes	Permission to change the other basic attributes of a file
Read_named_attrs	Permission to read the named attributes of a file
Write_named_attrs	Permission to write the named attributes of a file
Write_owner	Permission to change the owner
Synchronize	Permission to access a file locally at the server with synchronous reads and writes

The classification of operations recognized by NFS with respect to access control.

[25] Users/ Processes by 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

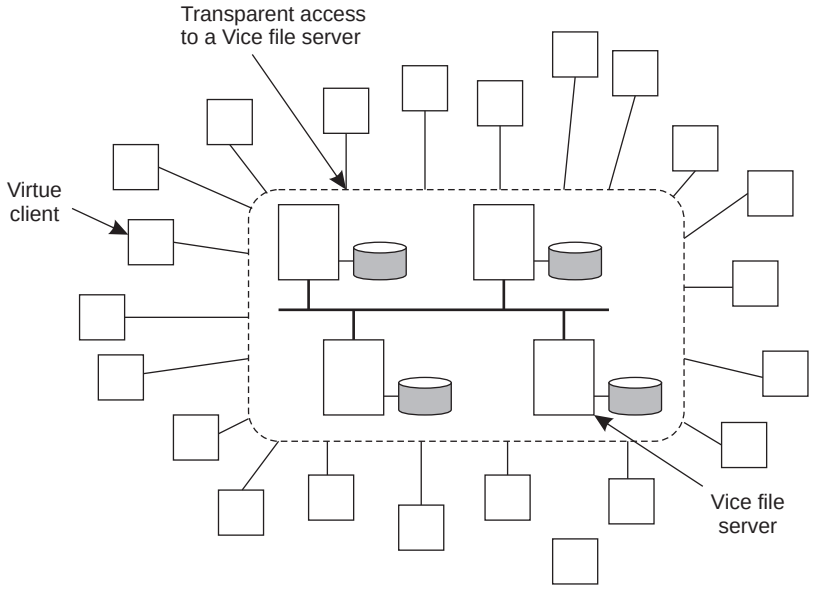
The various kinds of users and processes distinguished by NFS with respect to access control.

[26] The Coda File System

- developed at Carnegie Mellon University, main goal: high availability,
- advanced caching allows a client to continue operation despite being disconnected from a server,

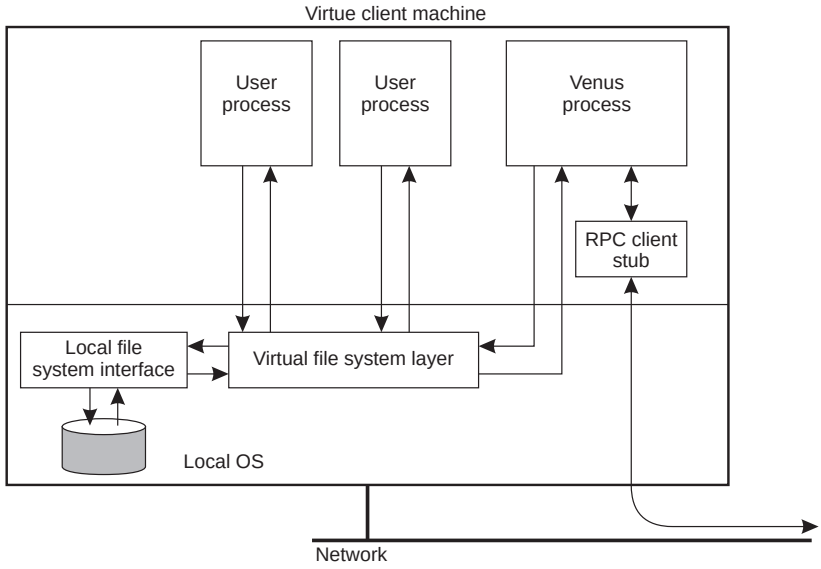
- descendant of version 2 of the Andrew File System (AFS),
- **Vice** file servers and **Virtue** workstations with **Venus** processes,
- both Vice file server processes and Venus processes run as user-level processes,
- a user-lever RPC on top of UDP providing *at-most-once* semantics,
- trusted Vice machines run authentication servers,
- Coda appears as a traditional UNIX-based file system.

[27] **Overview of Coda (1)**



The overall organization of AFS.

[28] **Overview of Coda (2)**

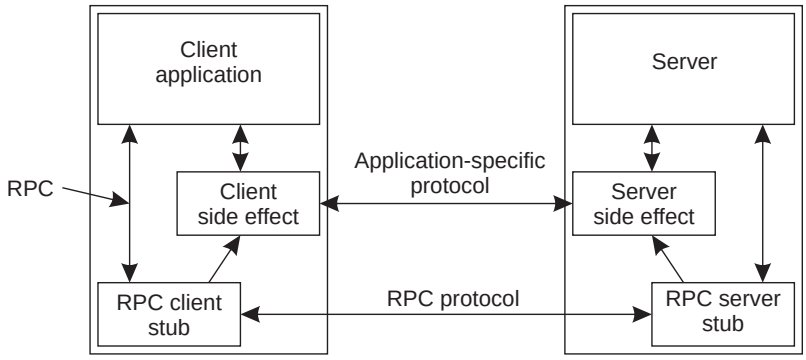


The internal organization of a Virtue workstation.

[29] **Coda - communication**

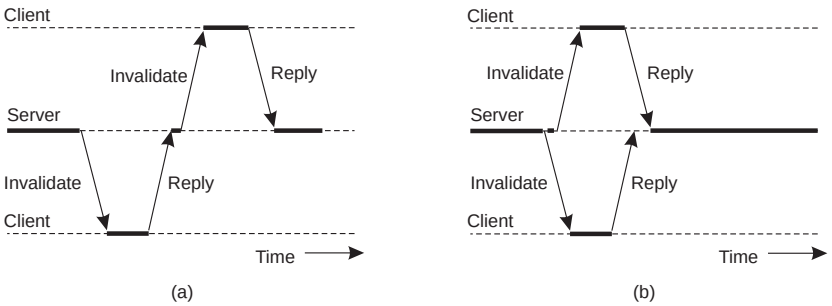
- RPC2 different to ONC RPC used by NFS,
- offers reliable communication on top of the UDP protocol,
- thread per each RPC request,
- back messages regularly sent by the server to the client,
- support for side effects – mechanisms for communication using an application-specific protocols,
- support for multicasting, parallel RPC implemented by means of **MutliRPC**, fully transparent to callees,
- threads in Coda non-preemptive and entirely in user space,
- separate thread to handle all I/O operations with low-level asynchronous I/O emulating synchronous I/O without blocking an entire process.

[30] **Communication (1)**



Side effects in Coda's RPC2 system.

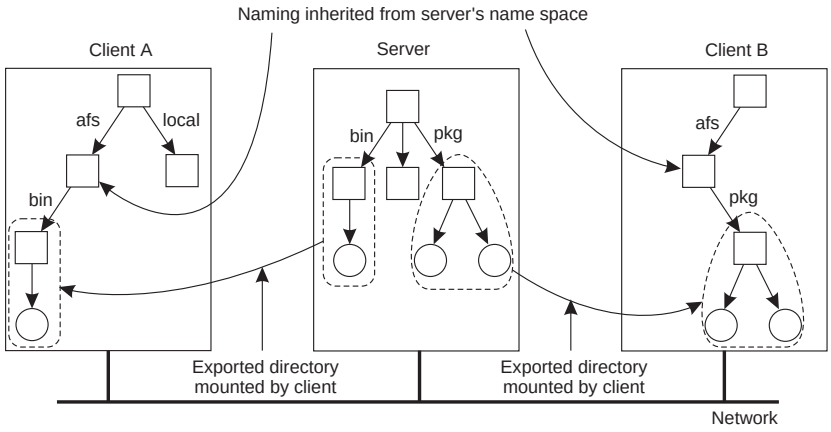
[31] **Communication (2)**



a. sending an invalidation message one at a time,

b. sending invalidation messages in parallel.

[32] **Naming**

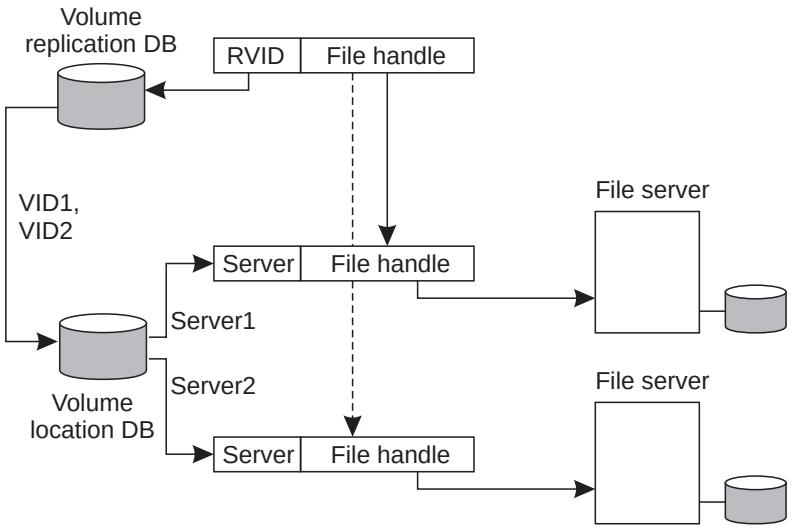


Clients in Coda have access to a single shared name space.

[33] **Volumes and File Identifiers**

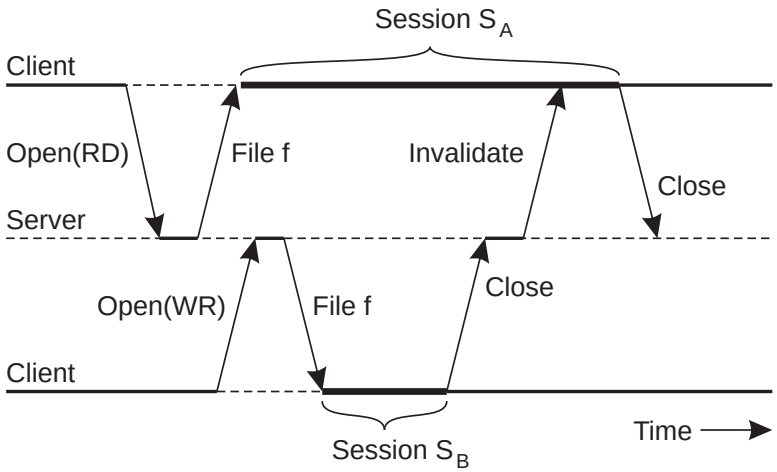
- volumes,
- only root nodes can act as mounting points,
- shared name space,
- file identifiers,
- **RVID** – replicated volume identifier,
- **VID** – volume identifier,
- volume replication database,
- volume location database,
- 64-bit handle identifying the file within the volume.

[34] **File Identifiers**



The implementation and resolution of a Coda file identifier.

[35] **Sharing Files in Coda**

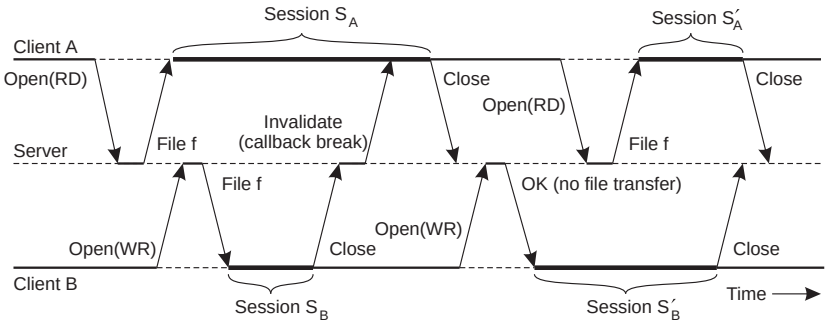


The transactional behavior in sharing files in Coda.

[36] **Transactional Semantics**

- **partition** - part of the network isolated from the rest,
- recognition of different types of session (like the store session type),
- usage of versioning scheme,
- update from a client accepted only when the update lead to the next version of a file,
- when conflict occurs, the updates from the client's session undone and client forced to save its local version of a file for manual reconciliation
- cache coherence maintained by means of callbacks,
- *callback promise*,
- *callback break*.

[37] **Client Caching**



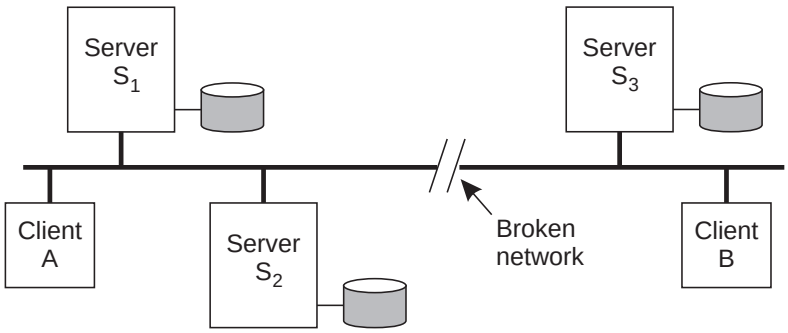
The use of local copies when opening a session in Coda.

[38] **Server Replication**

- file servers may be replicated,
- Volume Storage Group (**VSG**),

- client's Accessible VSG (**AVSG**),
- if the AVSG is empty, the client is said to be **disconnected**,
- consistency: Read-One, Write-All (**ROWA**),
- optimistic strategy for file replication,
- version vectors for conflicts detection.

[39] **Server Replication**



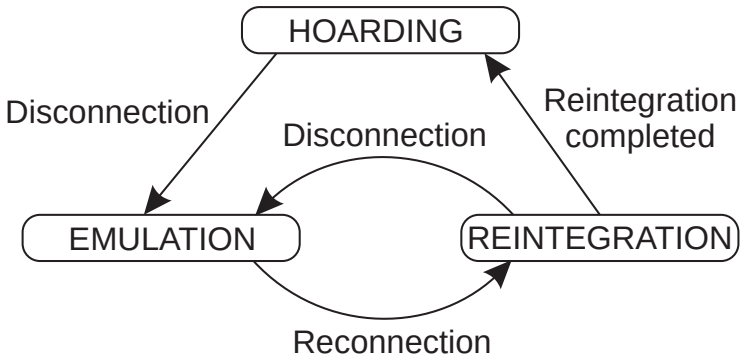
Two clients with different AVSG for the same replicated file.

[40] **Coda - Hoarding**

- **hoarding** – filling the cache in advance with the appropriate files,
- priority mechanism to ensure caching of useful data:
  - user may store paths in hoard database (one per workstation),
  - priority for each file based on the hoard database and last references,
- hoard walk invoked once every 10 minutes,
- cache in **equilibrium**, if:
  - no uncached file with a higher priority,
  - cache full or no uncached files with nonzero priority,

  - each cached file is a copy of the one from client's AVSG.
- anyway no guarantee.

[41] **Disconnected Operation**



The state-transition diagram of a Coda client with respect to a volume.

- `http://www.coda.cs.cmu.edu/`

[42] **Access Control**

Operation	Description
Read	Read any file in the directory
Write	Modify any file in the directory
Lookup	Look up the status of any file
Insert	Add a new file to the directory
Delete	Delete an existing file
Administer	Modify the ACL of the directory

Classification of file and directory operations recognized by Coda with respect to access control.

– also: useful support for the listing of negative rights.

[43] **Plan 9**

– bringing back the idea of having a few centralized servers and numerous client machines,

– Unix at Bell Labs team,

– *file-based distributed system*,

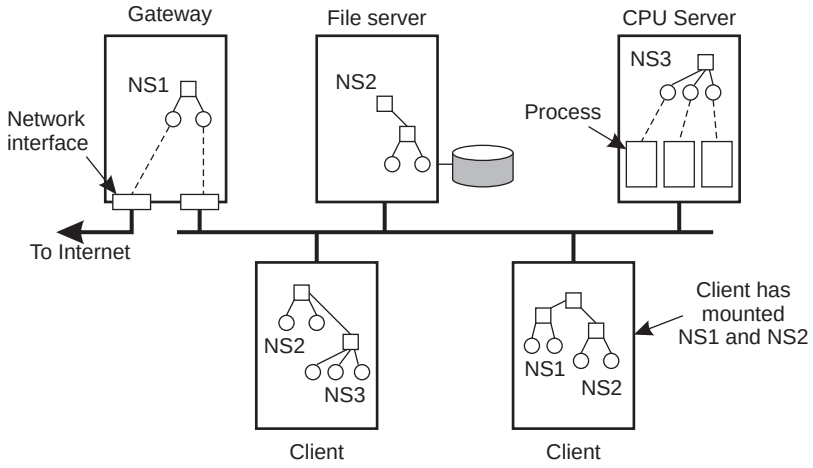
– all resources accessed in the same way (as files), including processes and network interfaces,

– each server offers a hierarchical name space to the resources it controls,

– communication through the protocol **9P**, tailored to file-oriented operations,

– for LAN **Internet Link** (**IL**) reliable datagram protocol, TCP for WAN.

[44] **Plan 9: Resources Unified to Files**



General organization of Plan 9.

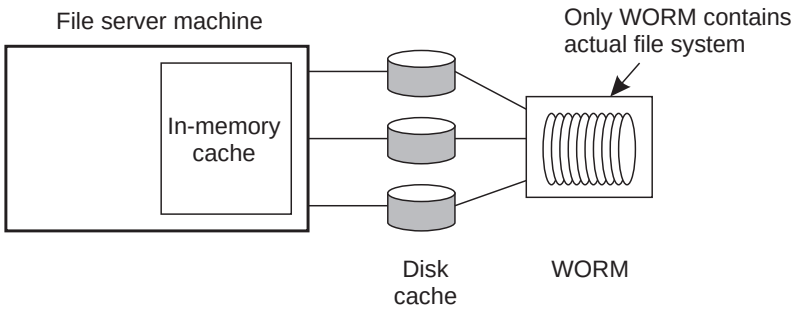
[45] **Communication**

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

Files associated with a single TCP connection in Plan 9.

- opening a telnet connection requires writing a special string to the **ctl** file "connect 192.31.231.42!23".

[46] **Processes**



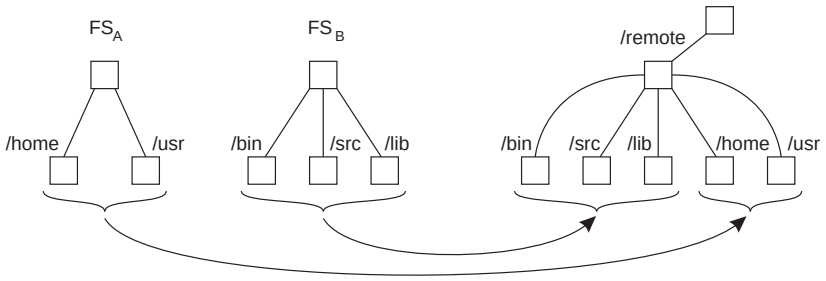
The Plan 9 file server.

[47] **Resource Management**

- let /net/inet denote the network interface,
- if M exports /net, a client can use M as a gateway by locally mounting /net and subsequently opening /net/inet.

- multiple name spaces can be mounted at the same mount point, leading to **union directory**,
- file systems appear to be Boolean or-ed,
- mounting order is important.
- Plan 9 implements UNIX file sharing semantics,
- all update operations always forwarded to the server.

[48] **Naming**



A union directory in Plan 9.

- `http://cm.bell-labs.com/plan9/`
- `http://www.vitanuova.com/inferno/`