第十章　分布式文件系统

- 简介
- 文件服务系统结构
- **NFS**网络文件系统
- **Coda**文件系统

简 介

- 文件系统
  - 持久存储（persistent storage）
- 分布式文件系统
  - 持久存储
  - 信息共享
  - 类似（有时更好）的性能和可靠性

文件系统的特点

- 功能
  - 组织, 存储, 检索, 命名, 共享和保护
- 文件相关的重要概念
  - 文件：包含数据和属性
  - 目录：提供从文件名到内部文件标识映射的特殊文件
  - 元数据：额外的管理信息，包括属性、目录等
- 文件系统结构
- 文件系统操作

文件属性

- 文件长度
- 创建时间戳
- 读时间戳
- 写时间戳
- 属性时间戳
- 引用计数
- 所有者
- 文件类型
- 访问控制列表ACL

文件系统结构

Directory module:	relates file names to file IDs
File module:	relates file IDs to particular files
Access control module:	checks permission for operation requested
File access module:	reads or writes file data or attributes
Block module:	accesses and allocates disk blocks
Device module:	disk I/O and buffering

分布式文件系统还需要附加模块实现客户服务器通信、分布式命名和文件定位

UNIX文件系统操作

filedes = open(name, mode) *filedes = create(name, mode)*	打开一个名字为 *name*的文件 创建一个名字为 *name*的文件 以上两个操作都返回一个打开文件的描述符 *mode* 可以是 *read*, *write* or both.
status = close(filedes)	关闭打开文件 *filedes*
count = read(filedes, buffer, n) *count = write(filedes, buffer, n)*	从被*filedes*引用的文件中传输*n*字节到缓冲区 从缓冲区传输*n*字节到被*filedes*引用的文件中 两个操作都返回实际的传输值，并移动读写指针
pos = lseek(filedes, offset, whence)	移动读写指针到指定的位移， *whence*决定是相对位移还是绝对位移
status = unlink(name)	删除一个名字为 *name*的文件
status = link(name1, name2)	为file (*name1*)增加一个新名 *(name2)*
status = stat(name, buffer)	读名字为 *name*的文件属性到缓冲区

分布式文件系统的需求（1）

- 透明性
  - 访问透明性
  - 位置透明性
  - 移动透明性
  - 性能透明性：当服务负载在一定范围内变化时，客户程序可以保持满意的性能
  - 扩展透明性：文件服务可以扩充，以满足负载和网络规模的增长
- 并发文件更新
  - 并发控制：客户改变文件的操作不影响其他用户访问或改变同一文件的操作
- 文件复制：多个副本
  - 更好的性能与容错
- 硬件和操作系统异构性：文件服务的接口必须有明确的定义。在不同操作系统和计算机上实现客户和服务器软件

分布式文件系统的需求（2）

- 容错
  - 为了处理暂时的通信错误，容错设计可以基于最多一次性语义
  - 对于幂等操作：支持最少一次性语义
  - 无状态的服务器: 崩溃重启时不需恢复
- 一致性
  - Unix提供单一副本更新语义：一个文件上的每个操作都对所有的进程都是瞬间可见的
  - 当文件在不同地点被复制和缓存时，可能会偏离单一副本更新语义
- 安全性
  - 身份验证，访问控制，安全通道
- 效率
  - 应提供比传统文件系统相同或更强的性能和可靠性

分布式文件系统的关键目标

- 如何保证透明性？
- 如何保证性能？
- 如何保证容错？
- 如何保证并发操作？

SUN网络文件系统NFS

- NFS简介
- 通 信
- 进程
- 命名
- 同步
- 缓存和复制
- 容错性
- 安全性

NFS简介

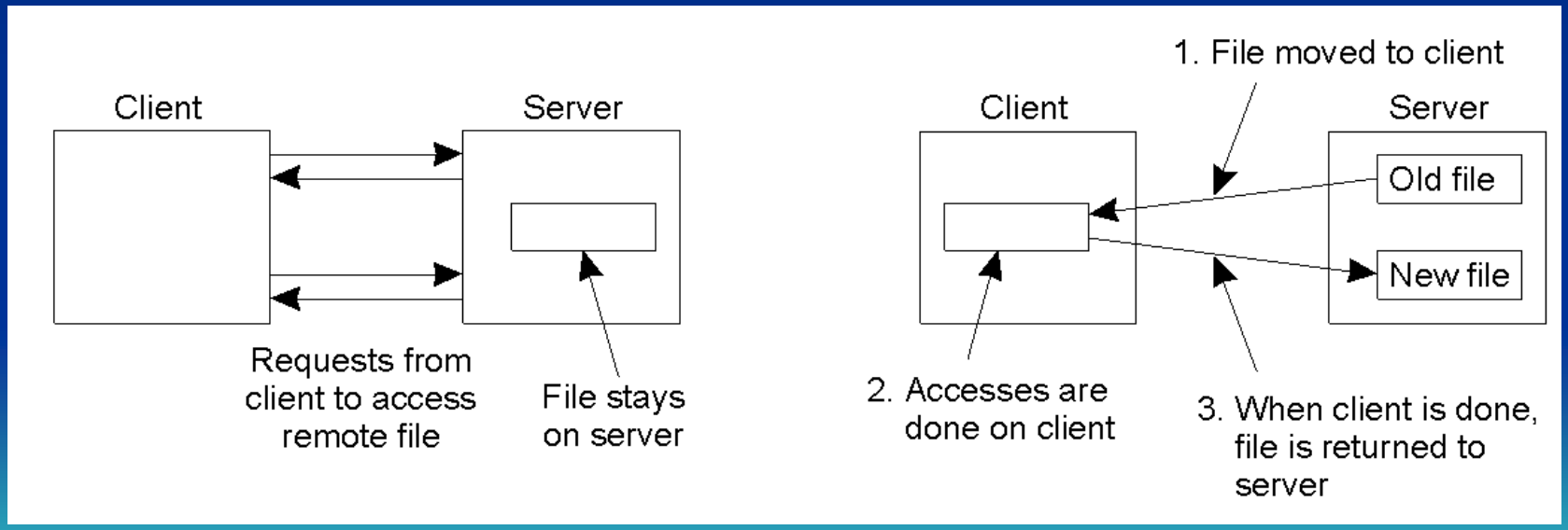
- NFS是Sun Microsystem公司1985年研制的网络FS，它是基于UNIX的，是第一个形成产品的DFS。
- NFS正在努力成为工业标准。
- NFS支持不同类型的系统，每台计算机在系统内安装NFS的客户组件和服务器组件。
- NFS的实现思想、协议、实现都非常有特色

实现思想

- 基本思想：让客户集和服务器的任何一个集合共享一个公用FS，NFS允许一台计算机既是客户机又是服务器。
- 当一个服务器输出某个目录时，该目录为根的子目录树同时被输出。服务器输出的目录列标记在文件的/etc/export中。
- 客户通过安装方式访问服务器的输出目录。
- 基本特征：服务器输出目录，客户从远处安装它们。
- 优点：当多个客户机同时安装同一个目录时，它们可以通过共享公用目录者的文件来进行通信。

NFS 体系结构 (1)

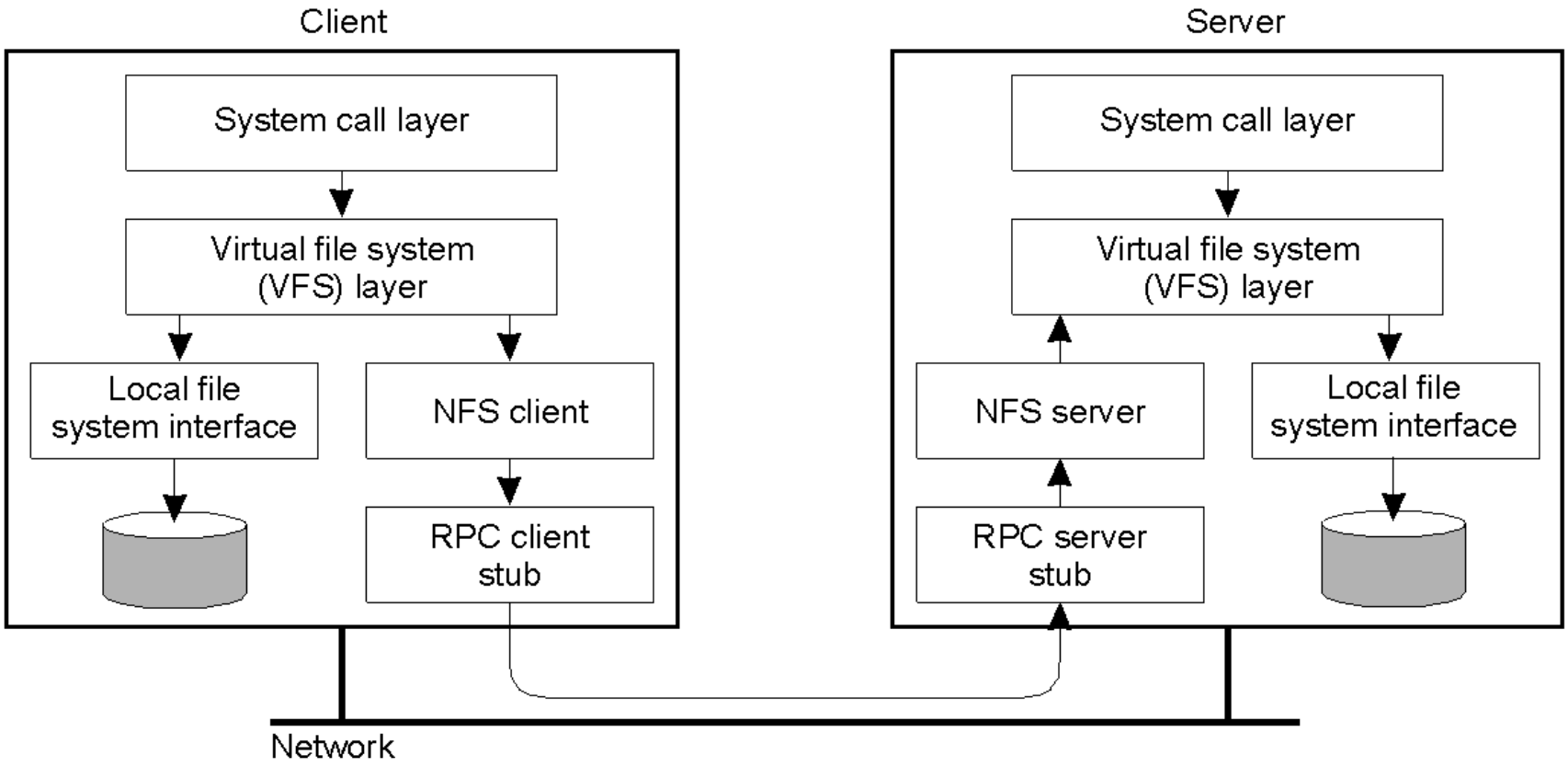
- 远程访问模型：客户被提供文件操作的接口，服务器负责实现这些操作。
  - 服务器实现的功能较复杂，容易造成服务器瓶颈。
- 上传/下载模型：客户从服务器下载文件后，在本地访问该文件；完成对该文件的访问后，再将该文件上传回服务器。
  - 服务器功能简单，对文件内部的操作全部由客户端自行完成。所以，要求客户端有较大的空间。
  - 每次进行的是整个文件的传送，从而给网络带来许多不必要的压力。



a) 远程访问模型
b) 上传/下载模型

NFS体系结构 (2)

NFS在很大程度上独立于本地文件系统



UNIX系统的基本NFS 结构

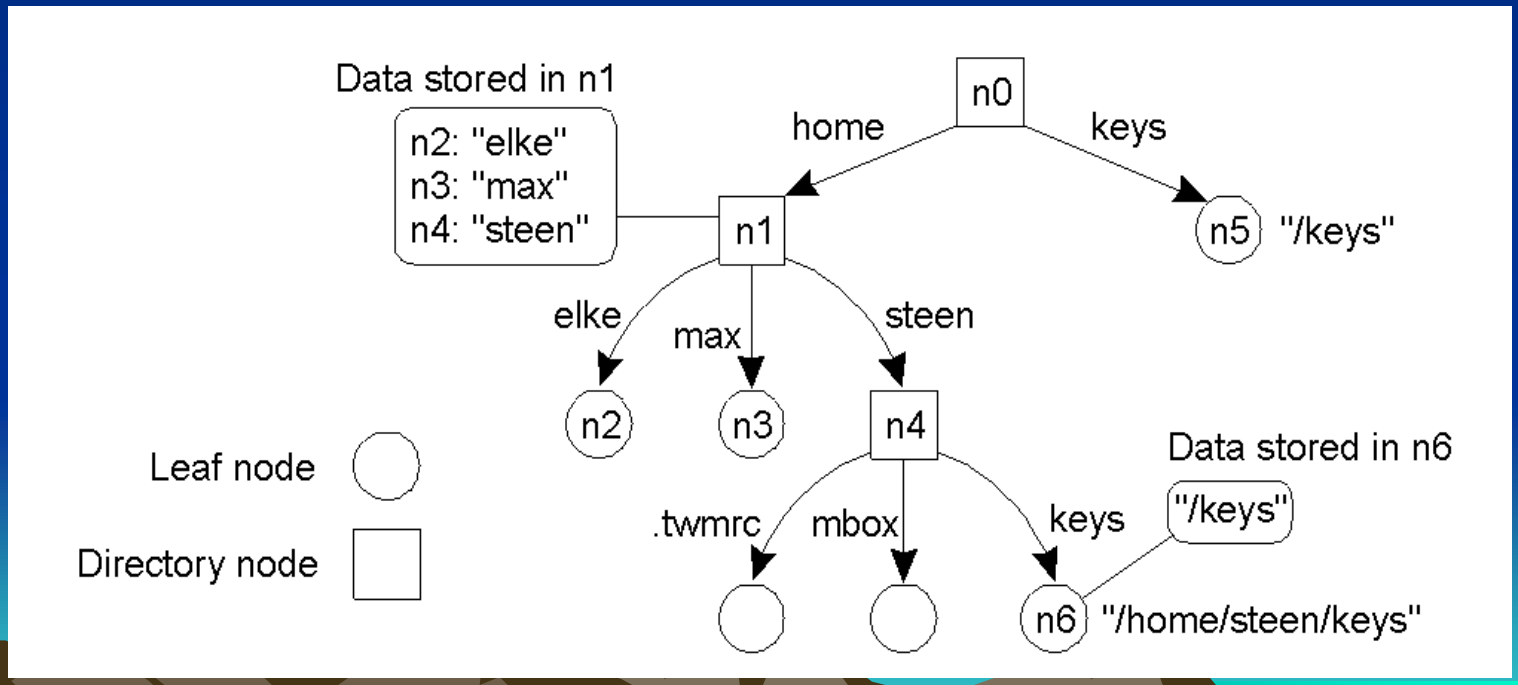
虚拟文件系统VFS

- VFS对每个文件系统的所有细节进行抽象，使得不同的文件系统在操作系统核心以及系统中运行的其他进程看来，都是相同的。
- VFS并不是一种实际的文件系统。它在系统启动时建立，在系统关闭时消亡。只存在于内存中，不存在于任何外存空间。
- VFS拥有关于各种特殊文件系统的公共界面，如超级块、inode、文件操作函数入口等。

文件系统模型

NFS文件系统模型与UNIX系统提供的文件系统模型很类似：

- 文件是字节流，层次化组织在命名图中
- 每个文件有文件句柄
- 支持硬链接和符号链接



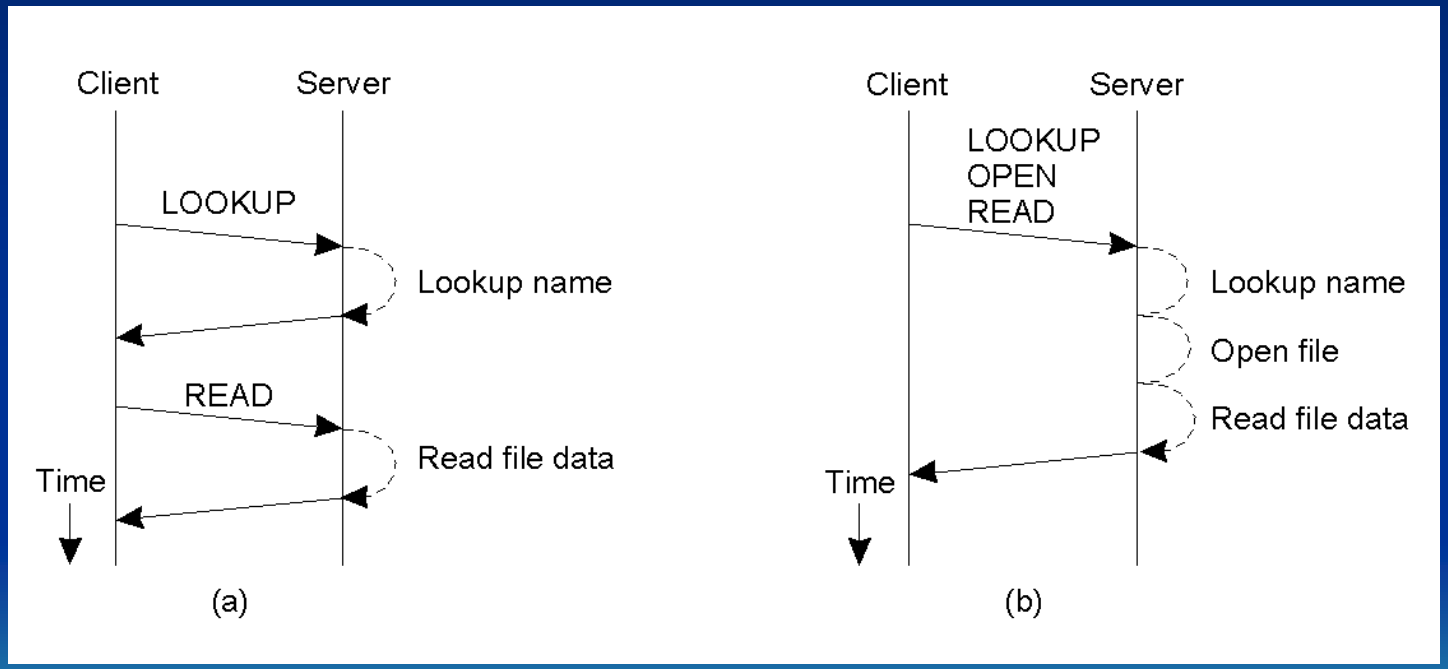
文件系统模型

NFS支持的文件系统操作的不完全列表.

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
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	Read 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

通 信

每个NFS操作都可以被实现为文件服务器的单一远程过程调用RPC，可以通过支持复合过程来提高性能，复合过程并不包含事务处理语义



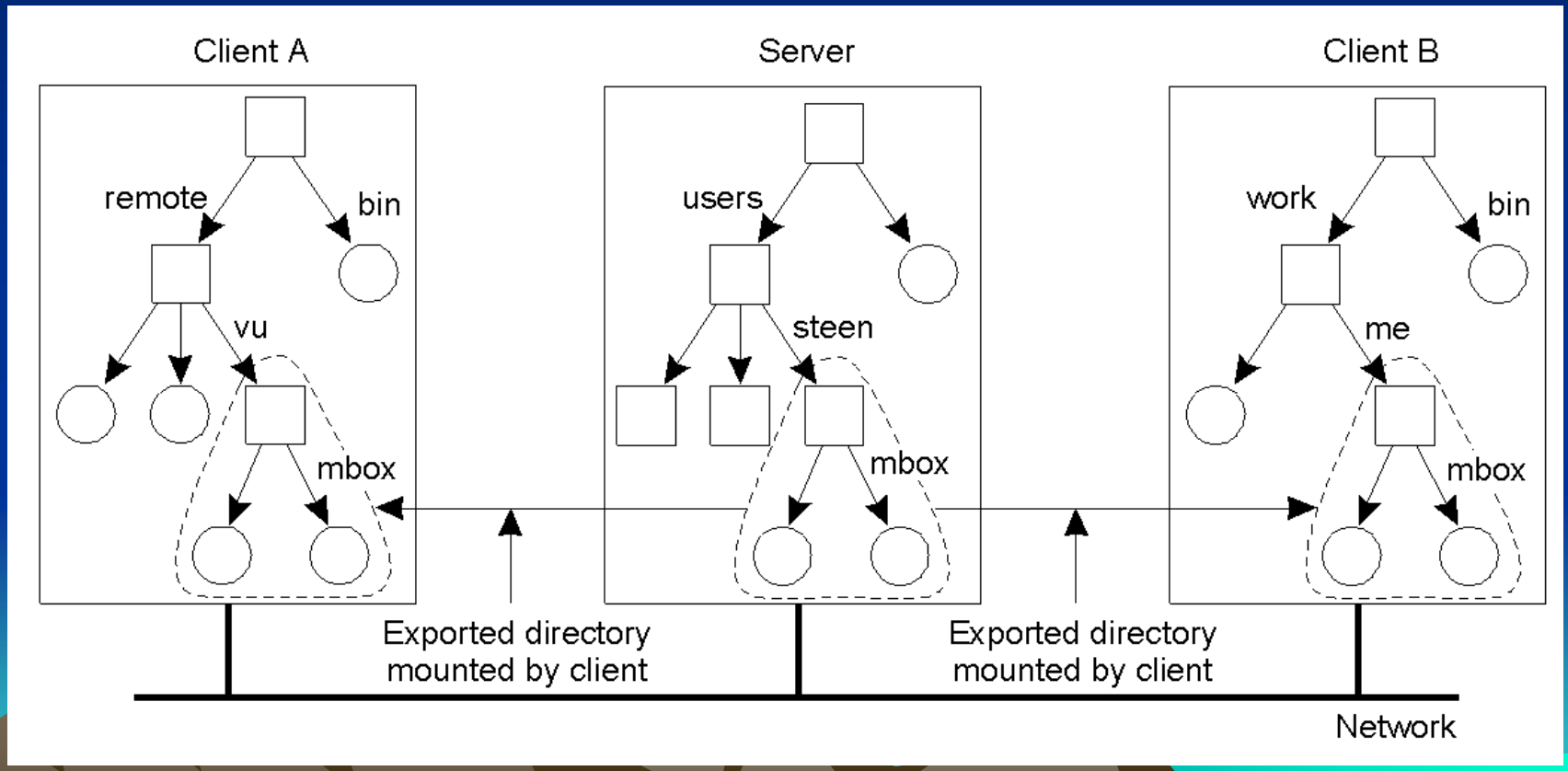
a) 在NFS version 3中从文件读取数据
b) 在NFS version 4中使用复合过程从文件读取数据

进程-无状态和有状态的方法

- 在NFS version 3 中NFS协议是无状态的，也就是说，服务器不必保持关于它的客户端的任何协议状态信息
- 优点是简单性：在失败的事件发生的时候，不需要进入回到原先状态的恢复阶段。
- 缺点是客户端得不到任何关于请求是否得以执行的保证。
- 在NFS version 4 中NFS放弃无状态的方法
  - 无状态的模型难以实现文件锁
  - 某些身份验证需要服务器保留客户的状态
  - 需要高速缓存协议，这些协议与保留客户文件信息的服务器合作的最好（如租用）

命名 (1)

- NFS命名模型为客户提供完全透明的，访问服务器保存的远程文件系统的机制
- 服务器可以输出（export）某个目录
- 用户不共享名称空间；解决办法：提供一个部分标准化的名称空间



NFS中装入（部分）远程文件系统.

命名 (2)

- 文件句柄
  - 对文件系统内文件的引用
  - 对文件是不变的，可以缓存在客户端
- 文件属性

文件属性 (1)

Attribute	Description
TYPE	文件类型 (regular, directory, symbolic link)
SIZE	文件长度
CHANGE	向用户指示文件是否已修改
FSID	服务器上文件所属文件系统的唯一标识

NFS中的强制文件属性

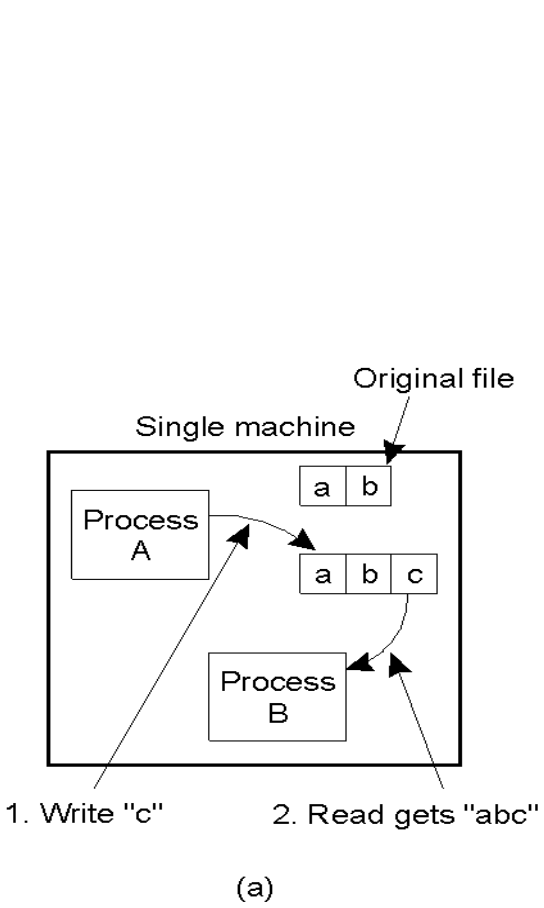
文件属性 (2)

Attribute	Description
ACL	文件关联的访问控制列表
FILEHANDLE	服务器提供的文件句柄
FILEID	文件系统对该文件的唯一标识
FS_LOCATIONS	可能找到该文件系统的网络地址
OWNER	文件所有者的字符串名称
TIME_ACCESS	文件最后访问时间
TIME_MODIFY	文件最后修改时间
TIME_CREATE	文件创建时间

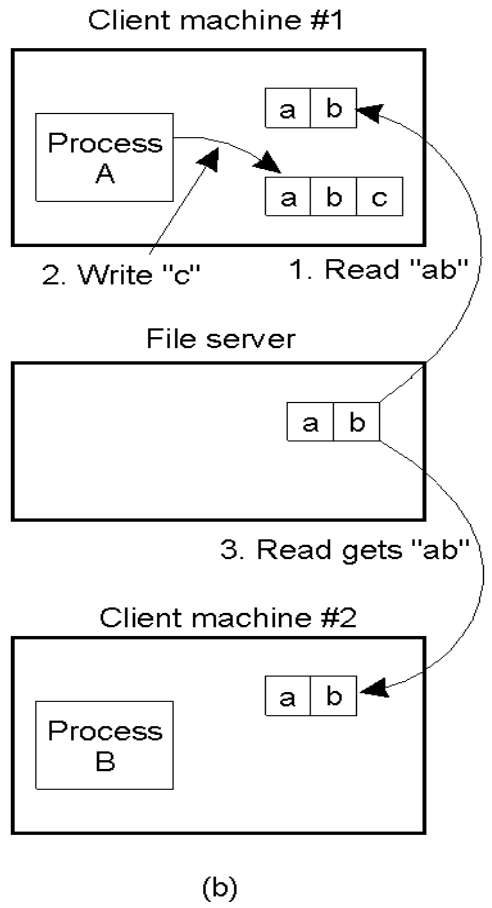
NFS中的推荐文件属性

同步-NFS中文件共享的语义

a) 在单处理器上，read操作出现在write操作之后时， read操作返回的是刚写入的值。
b) 使用缓存的分布式系统可能返回过时的值



(a)



(b)

同步-NFS中的文件锁定

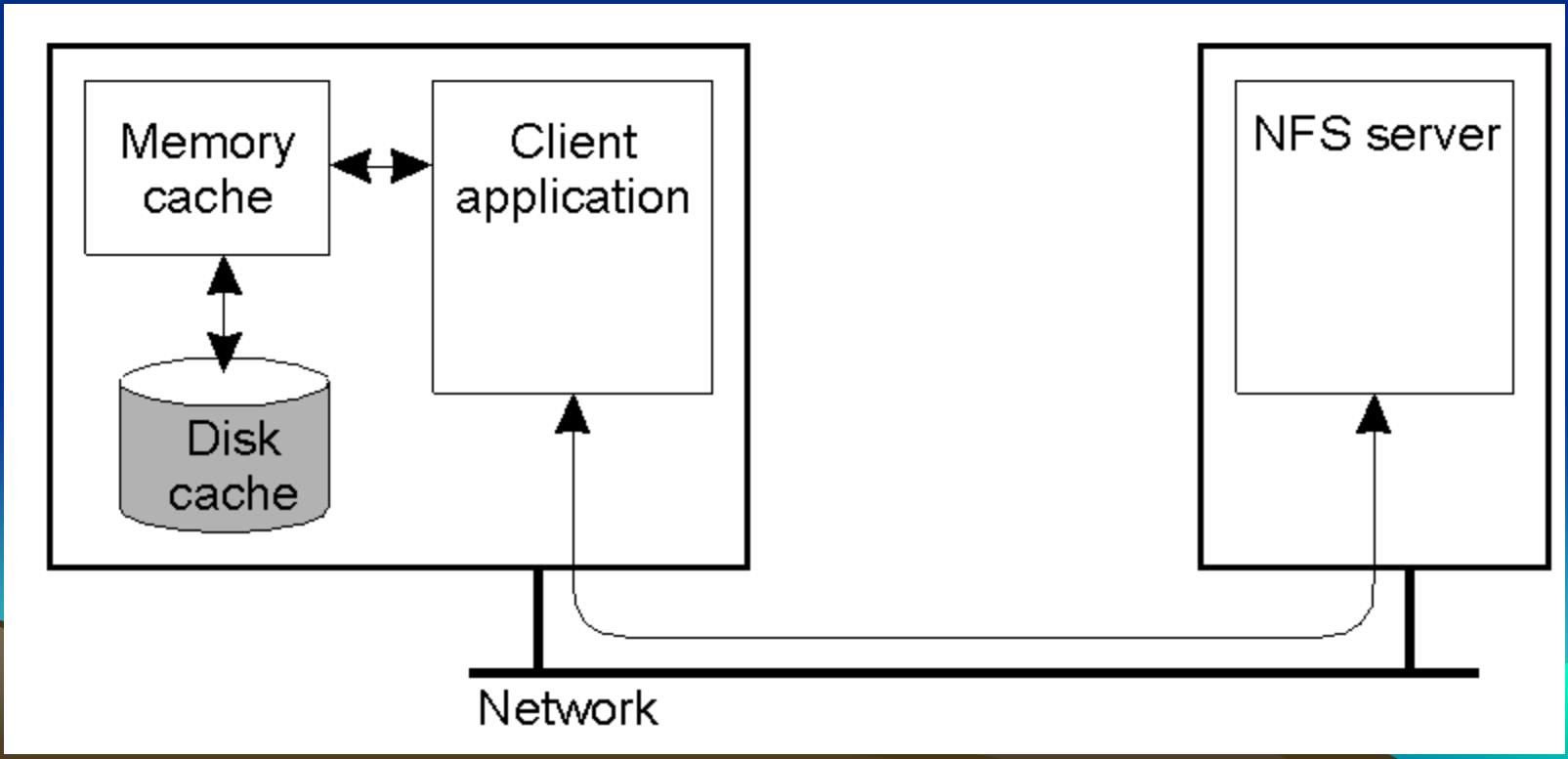
操作	描述
Lock	为一定范围的字节创建锁
Lockt	测试是否已授予冲突的锁
Locku	删除一定范围的字节上的锁
Renew	更新一个指定锁上的租用

NFS version 4 上关于文件锁定的操作

缓存和复制-客户端缓存

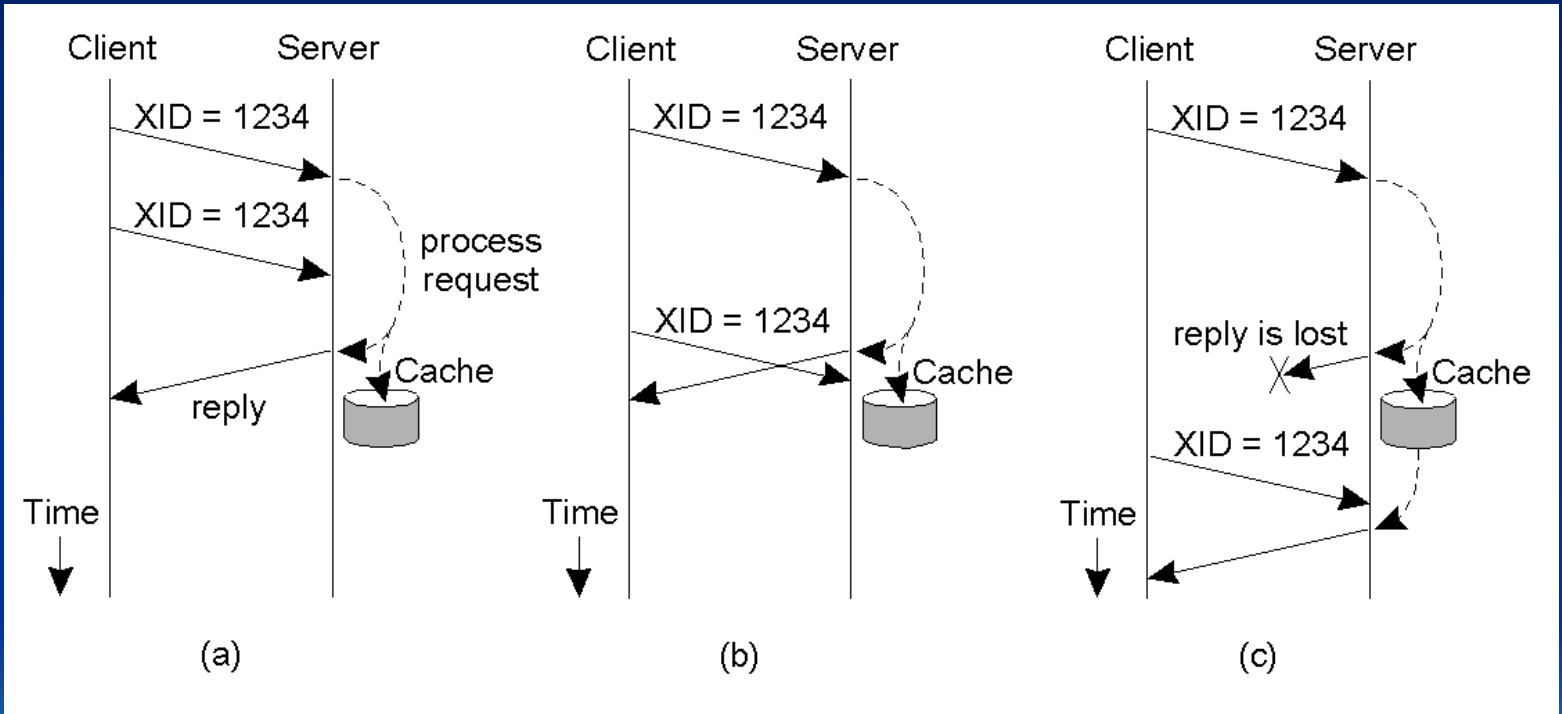
NFS中的客户端缓存：

- 读操作缓存和写操作缓存
- 同一机器上的多客户可以共享一个高速缓存



容错性-RPC 故障

- NFS的底层RPC不能保证可靠性，而且缺乏对重复请求的检测
- NFS 服务器提供重复请求高速缓存解决：XID事务处理标识符



处理重传的三种情况

a) 请求正在处理

b) 刚返回响应

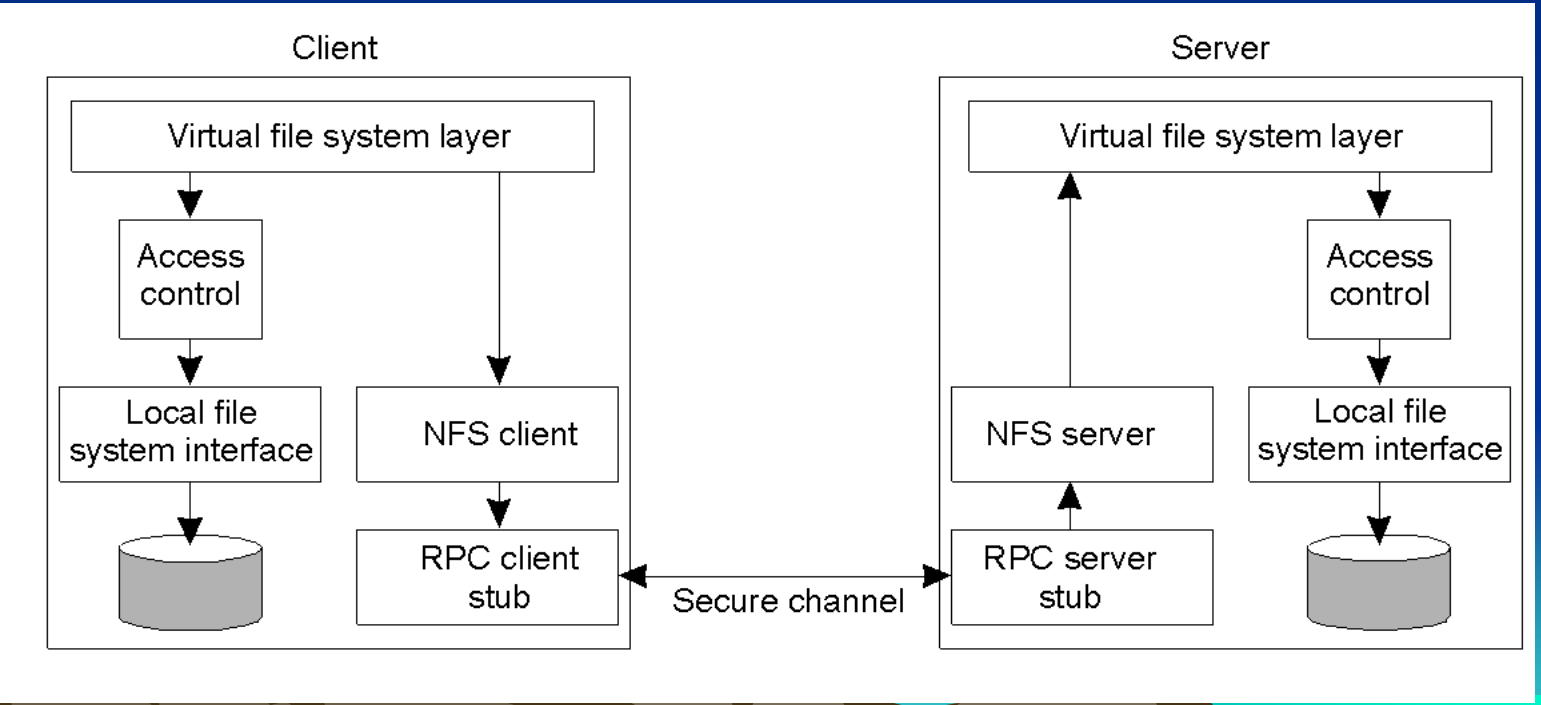
c) 早已返回响应，但响应丢失

容错性

- 出现故障的文件锁定
  - 客户端崩溃：使用租用
    - 要考虑时钟同步
    - 发送租用更新消息的可靠性
  - 服务器崩溃：恢复后进入宽限期，只接收要求归还锁的请求

安全

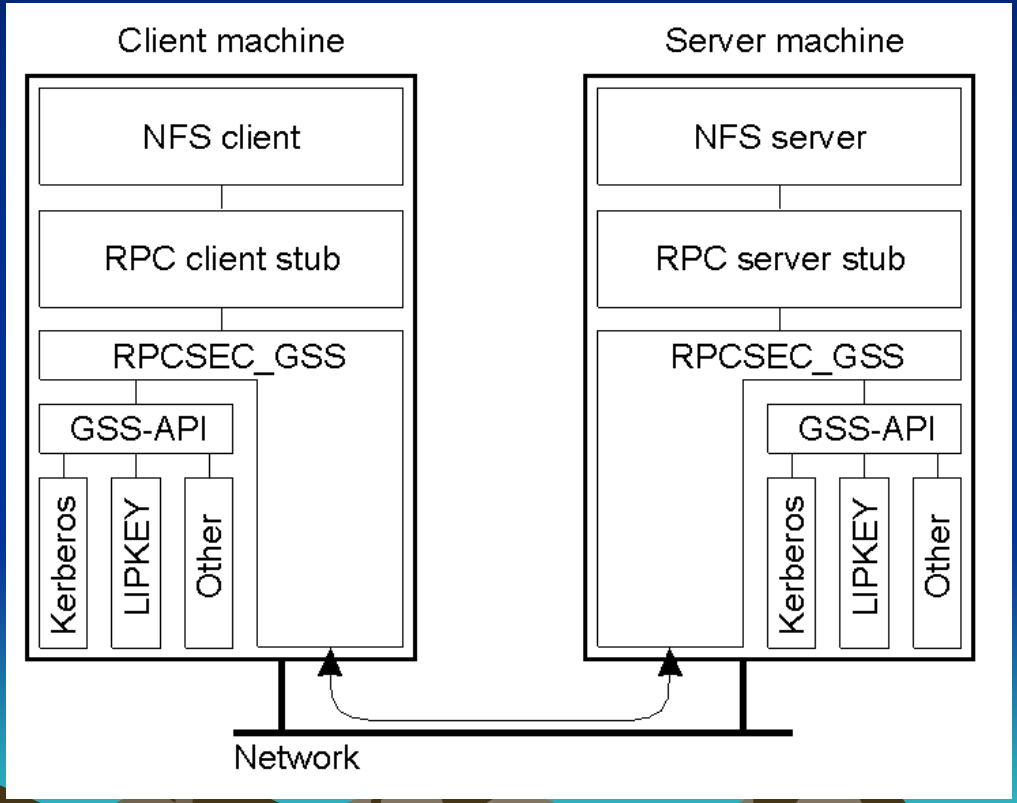
使用安全RPC建立安全通道
在NFS version 4 之前只支持身份验证：
•系统身份验证：客户向服务器以明文发送用户ID、组ID，服务器假设客户已通过登录过程，且信任客户所在的机器
•安全NFS： 使用Diffie-Hellman方法建立共享会话密钥
•Kerberos协议



NFS 安全性架构

安全 RPCs

- NFS version 4支持通用的安全框架RPCSEC_GSS
- RPCSEC_GSS不仅支持不同的身份验证系统，而且支持消息的机密性和完整性
- NFS安全设计为不仅支持自己的安全性机制，而且支持处理安全性的标准方法



NFS version 4 中的安全RPC

访问控制(1)

- NFS访问控制的操作分类

操作	描述
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 t5o 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

访问控制 (2)

- NFS中关于访问控制的各种用户和进程

用户类型	描述
Owner	The owner of a file
Group	The group of users associated with a file
Everyone	Any user of a 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 a batch job
Anonymous	Anyone accessing the file without authentication
Authenticated	Any authenticated user of a process
Service	Any system-defined service process

Coda 文件系统

- Coda 概述
- 通 信
- 命名
- 缓存和复制
- 容错性
- 安全性

Coda概述 (1)

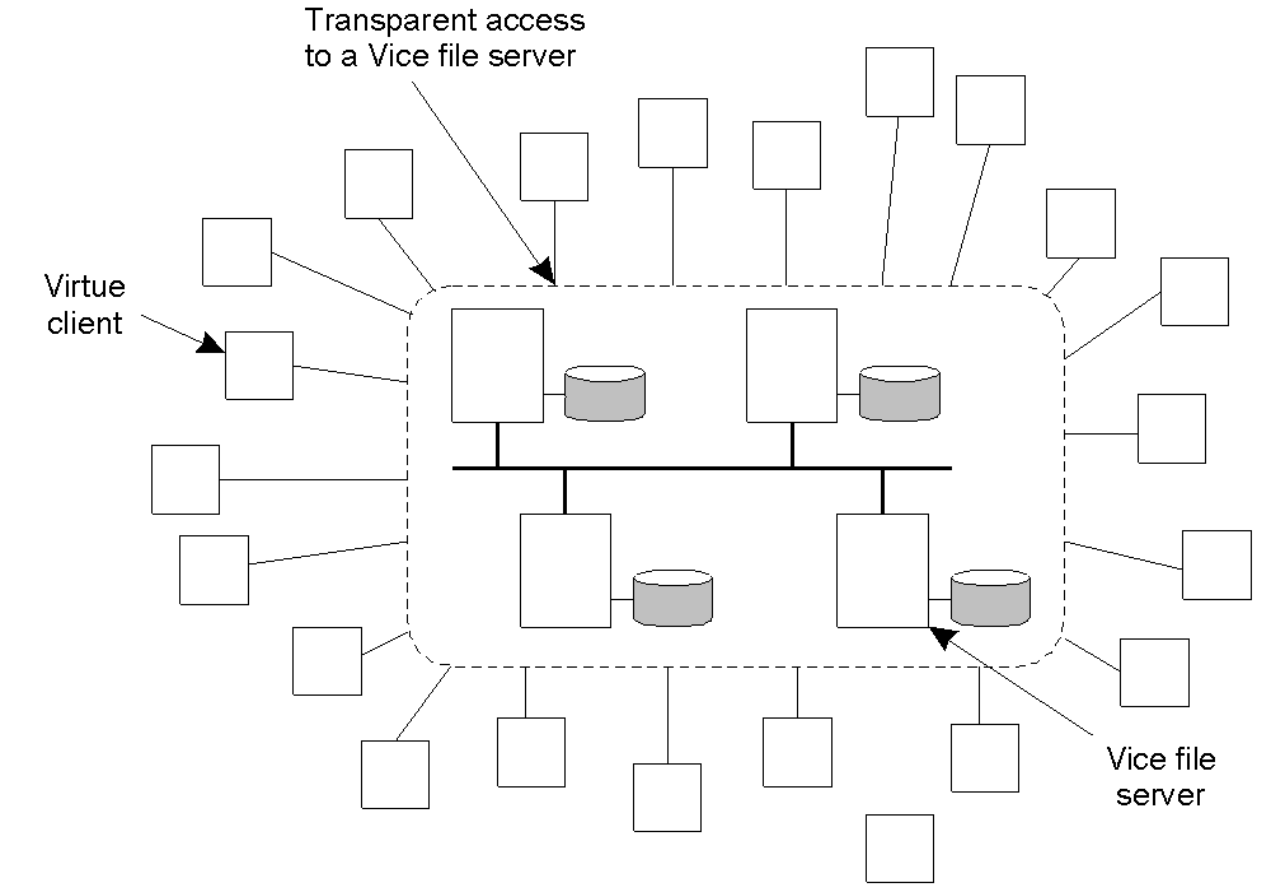
Coda分布式文件系统是由卡耐基·梅隆大学**1990**年开发的一个分布式文件系统。源于**AFS**（**Andrew file system**）第二版，在移动计算处理上具有很多别的系统没有的先进特性（如高可用性）。

具有以下特性：

- 离线状态下移动客户端仍可操作
  - 保持离线状态客户端的数据一致性
- 错误恢复
  - 解决服务器之间的冲突
  - 处理服务器之间的网络故障
  - 处理断开的客户端
- 性能和可靠性
  - 客户端可靠性能、持久的保存文件、目录以及属性。
  - 回写式缓存
- 安全性
  - **Kerberos**方式身份验证
  - 访问控制列表
- 可以免费获取源代码

Coda概述 (2)

AFS节点分成两组：Vice服务器和Virtue工作站

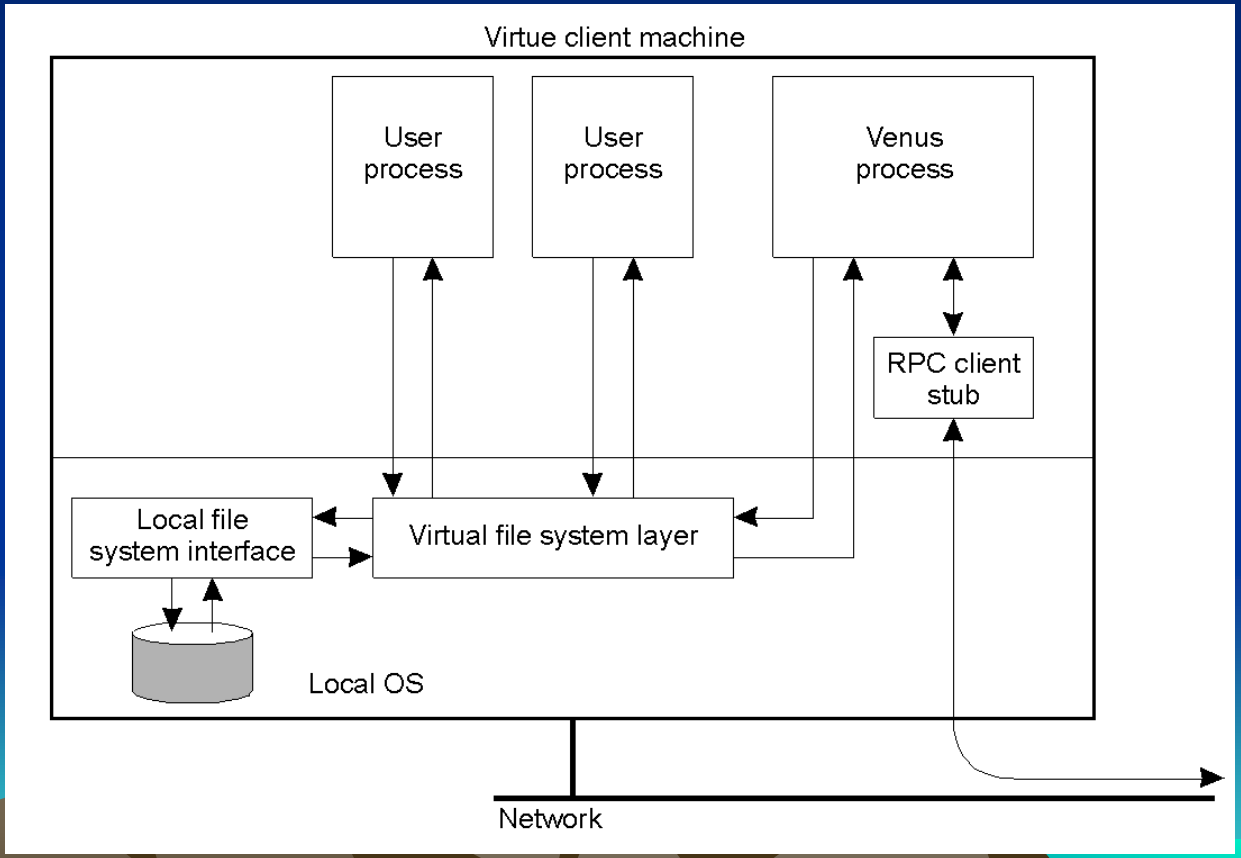


AFS的总体组织结构

Coda概述 (3)

每个Virtue 工作站都有一个Venus用户级进程：

- 作用类似于NFS客户，负责维护访问Vice服务器文件的机制
- 还负责使客户即使不能访问文件服务仍可继续执行



Virtue 工作站的内部组织

Coda概述 (4)

Vice服务器上的进程

- Vice文件服务器：维护本地文件，以用户态运行
- 身分验证服务器
- 更新进程：保持文件系统上的元信息在每台Vice服务器上保持一致

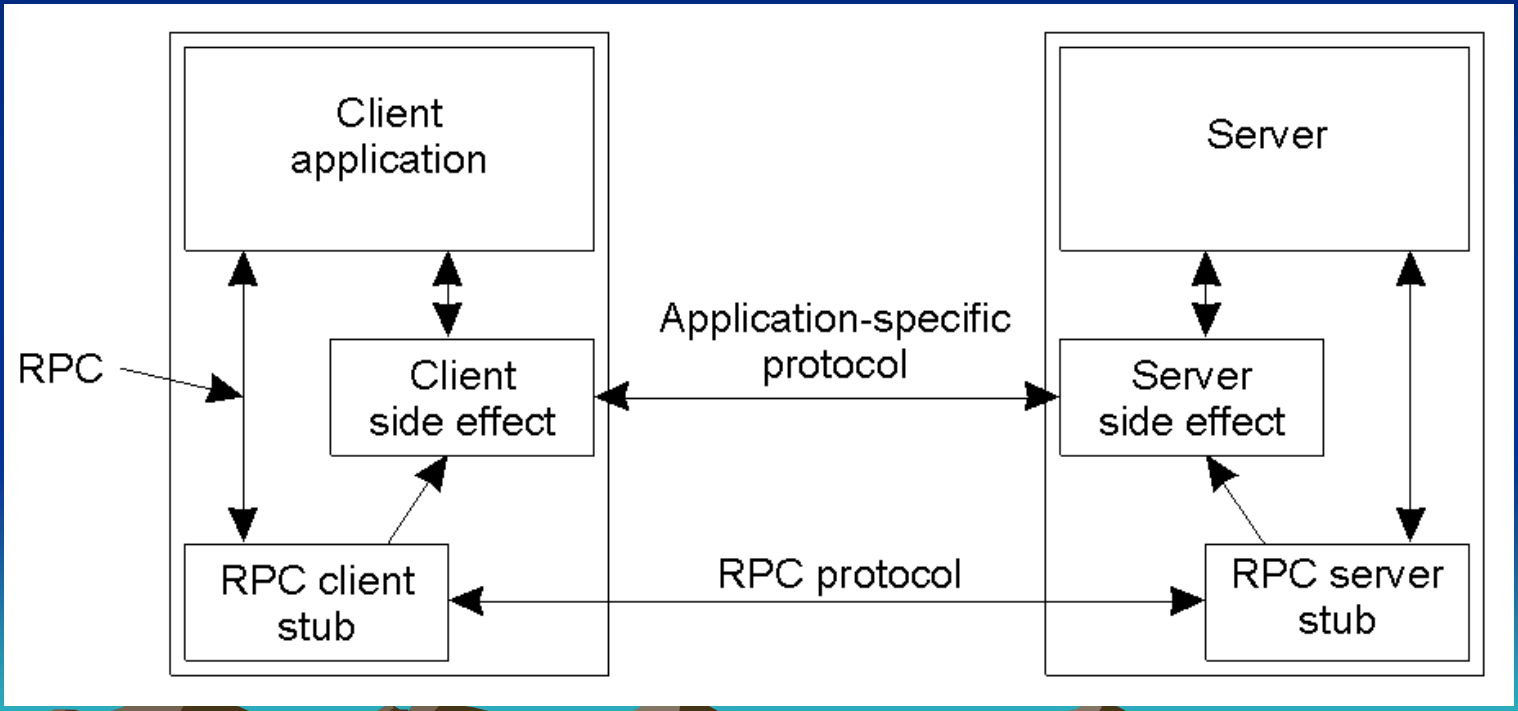
通信 (1)

RPC2要比传统**RPC**复杂得多：

•在**UDP**协议上提供可靠的**RPC**：调用远程过程时，客户端启动新线程负责与服务器的通信；服务器有规律的向客户返回消息，表示仍在处理请求。

•支持副作用（ Side effects ）：提供副作用例程接口，客户和服务器可以使用针对应用程序的协议进行通信的机制

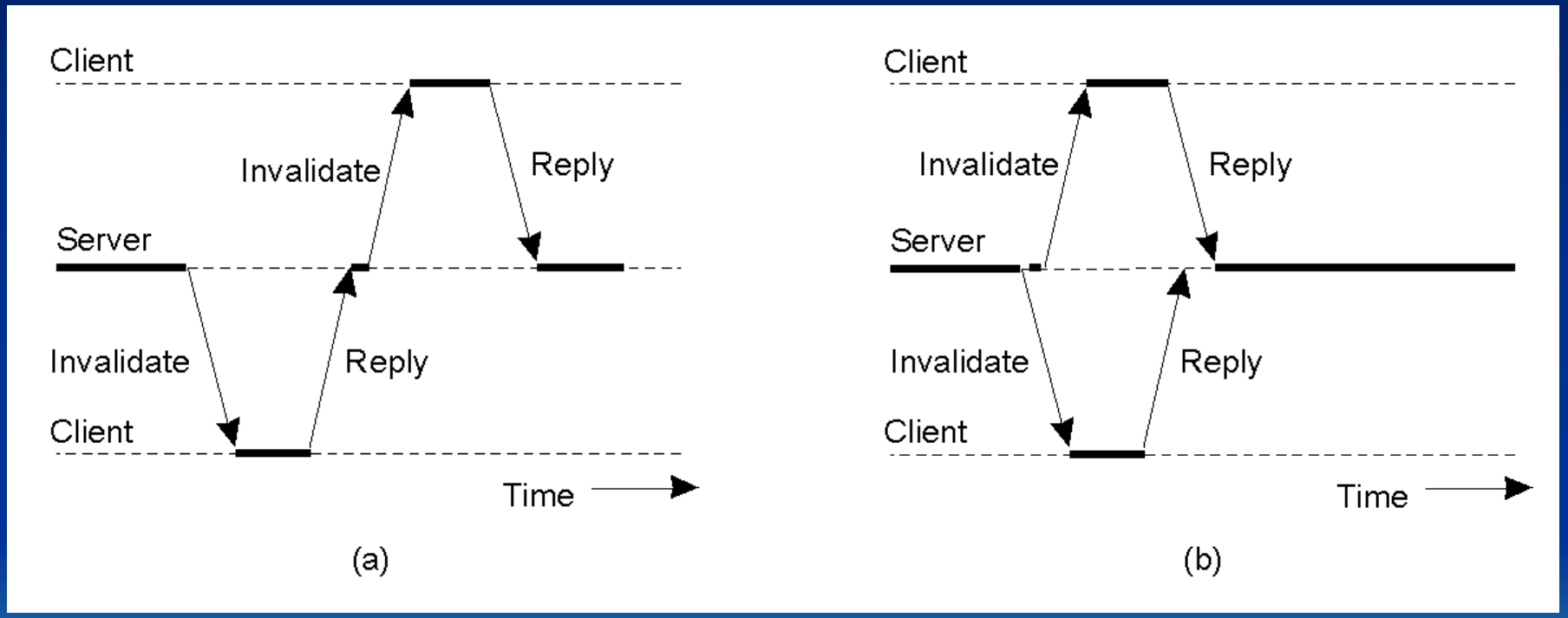
•支持多播



Coda RPC2 系统中的副作用

通信 (2)

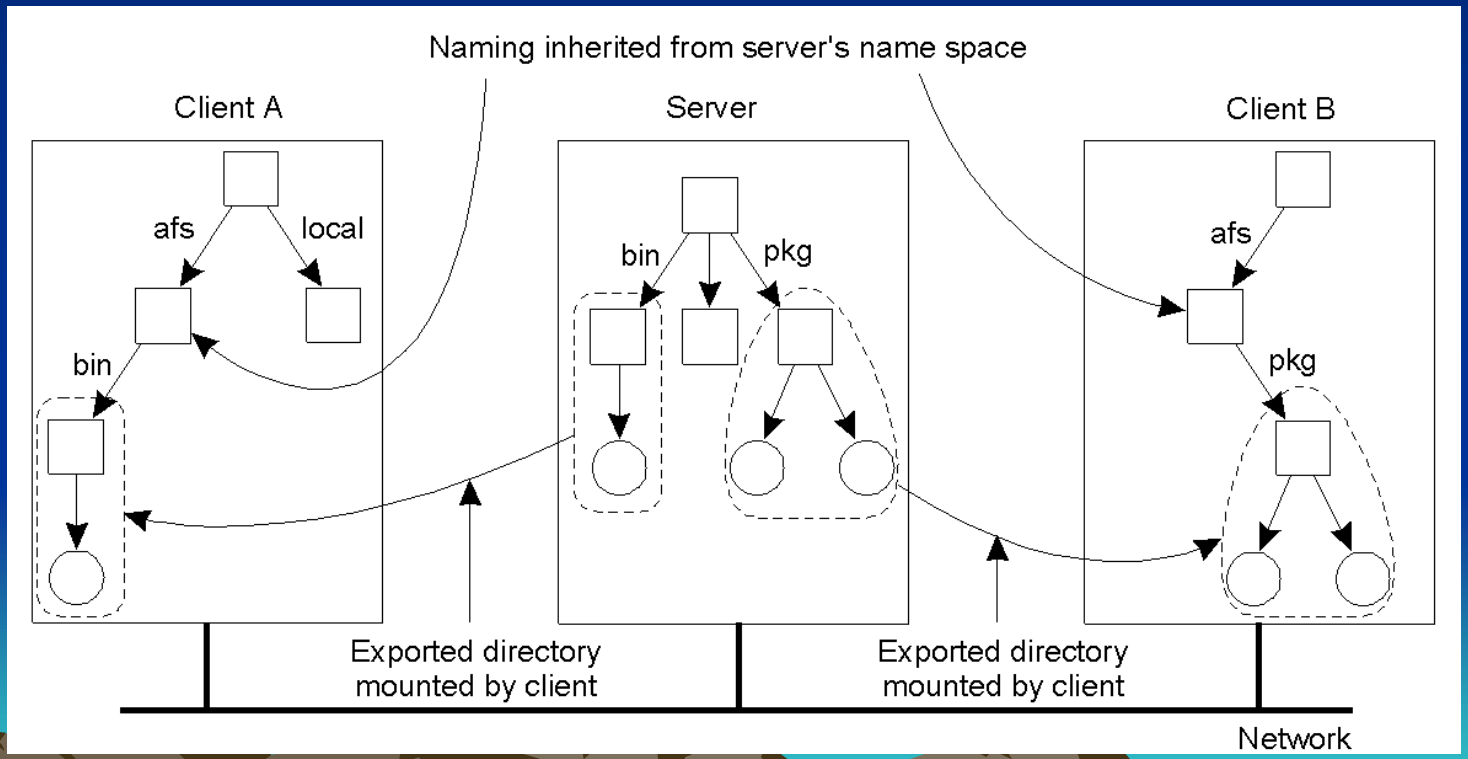
Coda中服务器需要记录哪些客户具有文件的本地拷贝，当文件被更改时，需要通过**RPC**通知这些客户，使其本地拷贝失效。可以使用多播。



a) 一次发送一个无效化消息
b) 并行地发送无效化消息

命名

Coda的命名系统类似UNIX。
使用卷（volume），类似磁盘分区
•是构造整个名称空间的基本单元
•是服务器端复制的单元，当装入一个卷时，每个Venus进程确保以/afs为根的命名图总是Vice服务器共同维护的完整名称空间的一个子图



Coda客户访问统一的共享名称空间

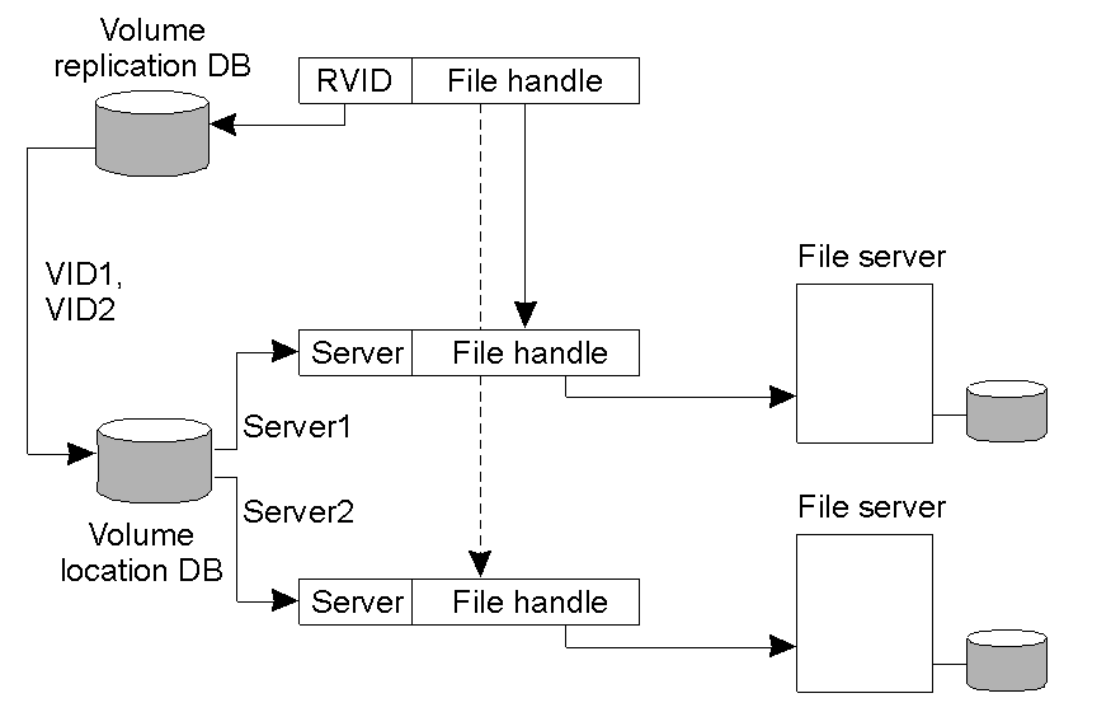
文件标识符

Coda的每个文件包含于唯一的一个卷中，该卷可能复制于多个服务器。

物理卷：有自己的**VID(**卷标识符**)**

逻辑卷：被复制的物理卷，关联同一个**RVID(**复制卷标识符**)**。

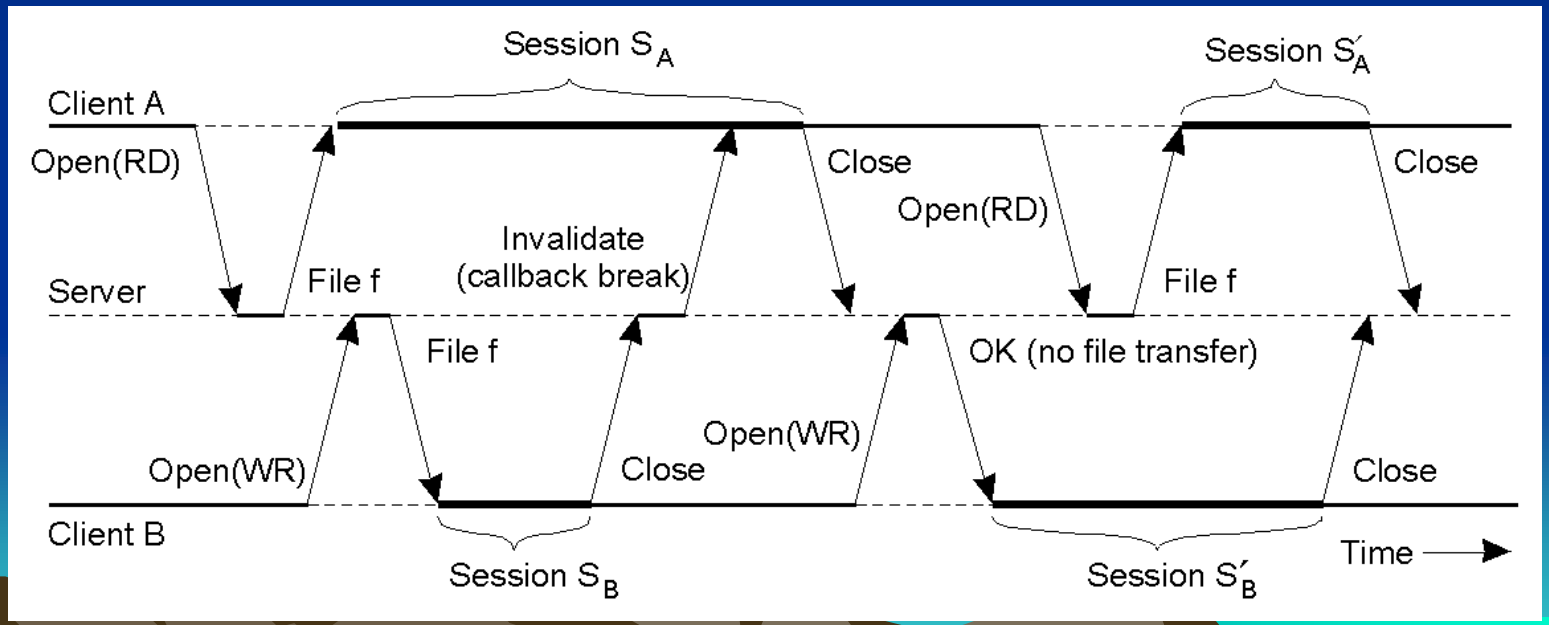
一个文件对应**96**位的文件标识符。



缓存和复制-客户端缓存

客户端缓存的重要性：

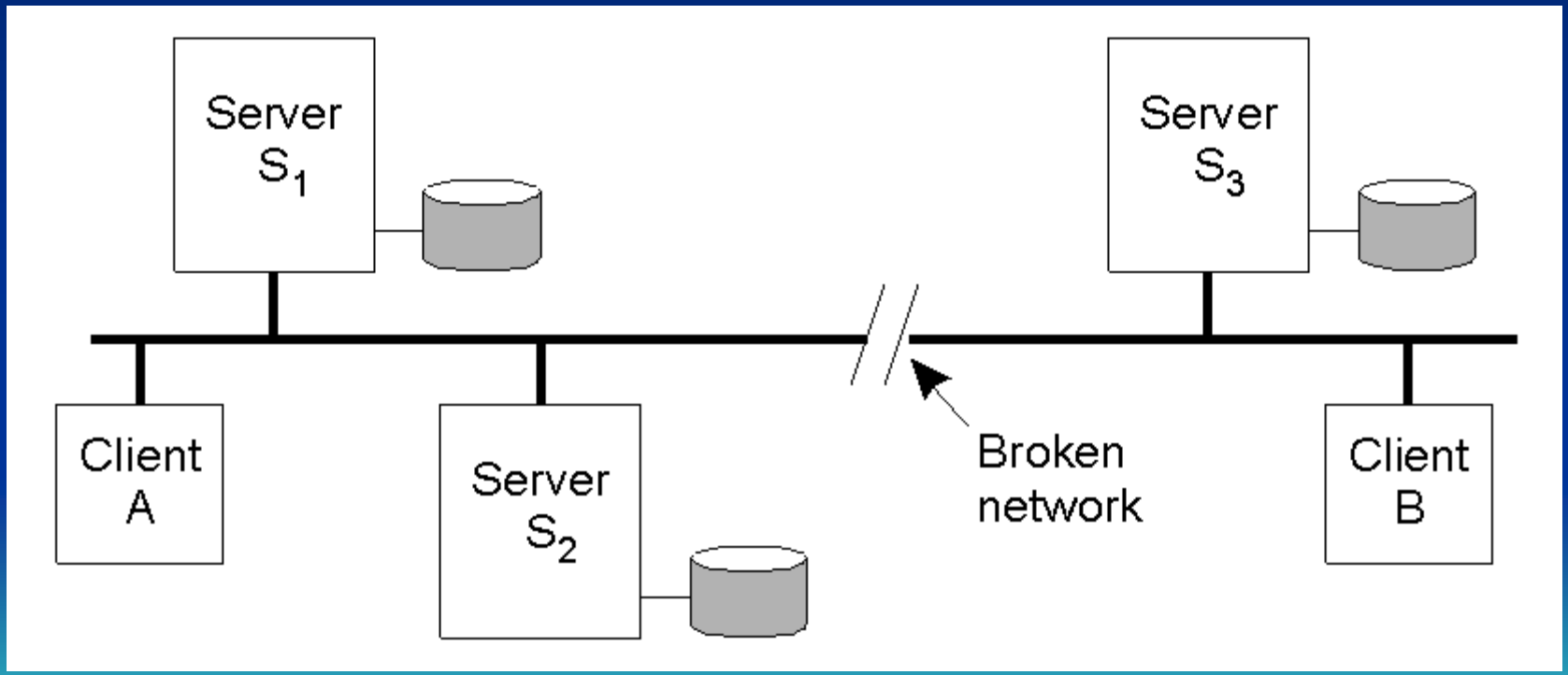
- 可扩展性
- 容错性
- Coda高速缓存整个文件
- 回叫承诺：服务器记录哪些客户在本地缓存了文件的拷贝
- 如果文件被客户更改，会通知服务器，后者向其他客户发无效化消息（回叫中断：服务器废弃回叫承诺）
- 如果客户在服务器上有未被废弃的回叫承诺，它就可以安全地在本地访问文件。



Coda中打开会话时本地备份的使用

缓存和复制-服务器复制

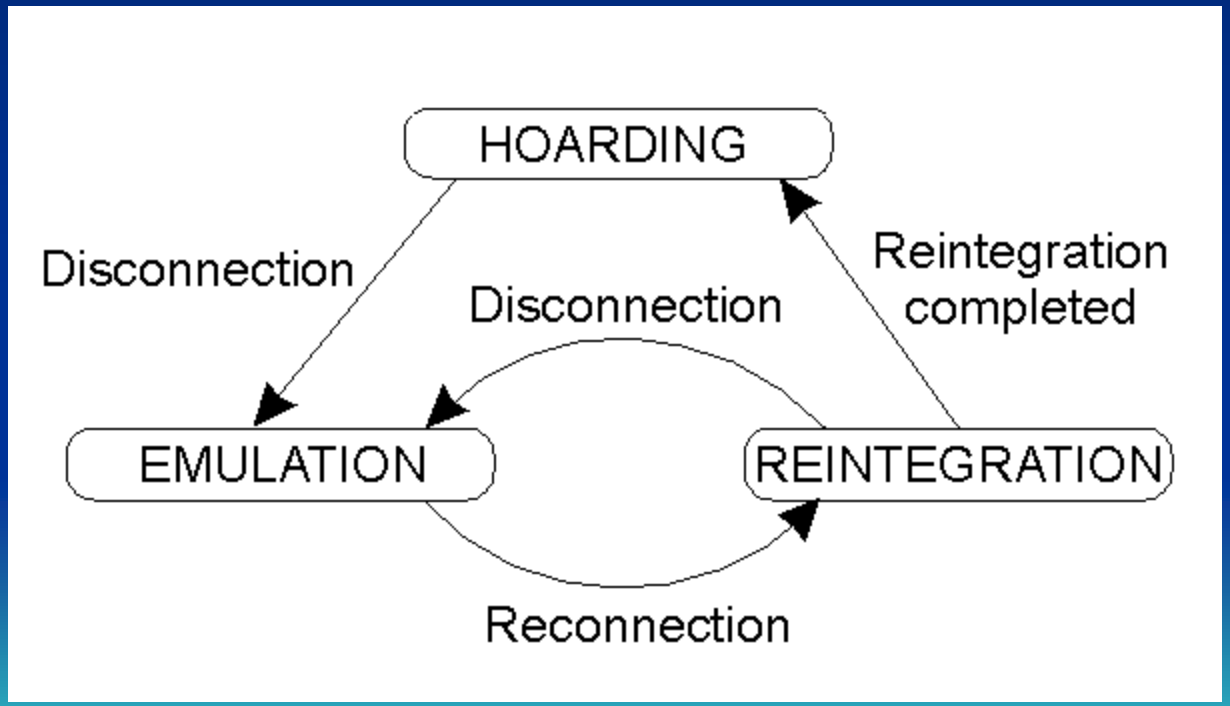
- VSG (Volume Storage Group)是一个保存同一复制卷的服务器集合。
- 可用的VSG成员叫做AVSG (Available VSG members)。
- Coda使用ROWA（read one, write all）来维护复制卷的一致性
- 使用版本记录方案检测不一致性



对于同一复制文件，具有两个不同AVSG的客户

容错性-断开连接的操作（1）

Coda允许客户在断开连接时（AVSG为空）的继续操作。
使用本地备份，再次连接后回传服务器。
基于事实：两个进程打开相同的文件进行写操作很罕见。
使用储藏技术（hoarding）。



对于一个卷， Coda客户的状态转换图

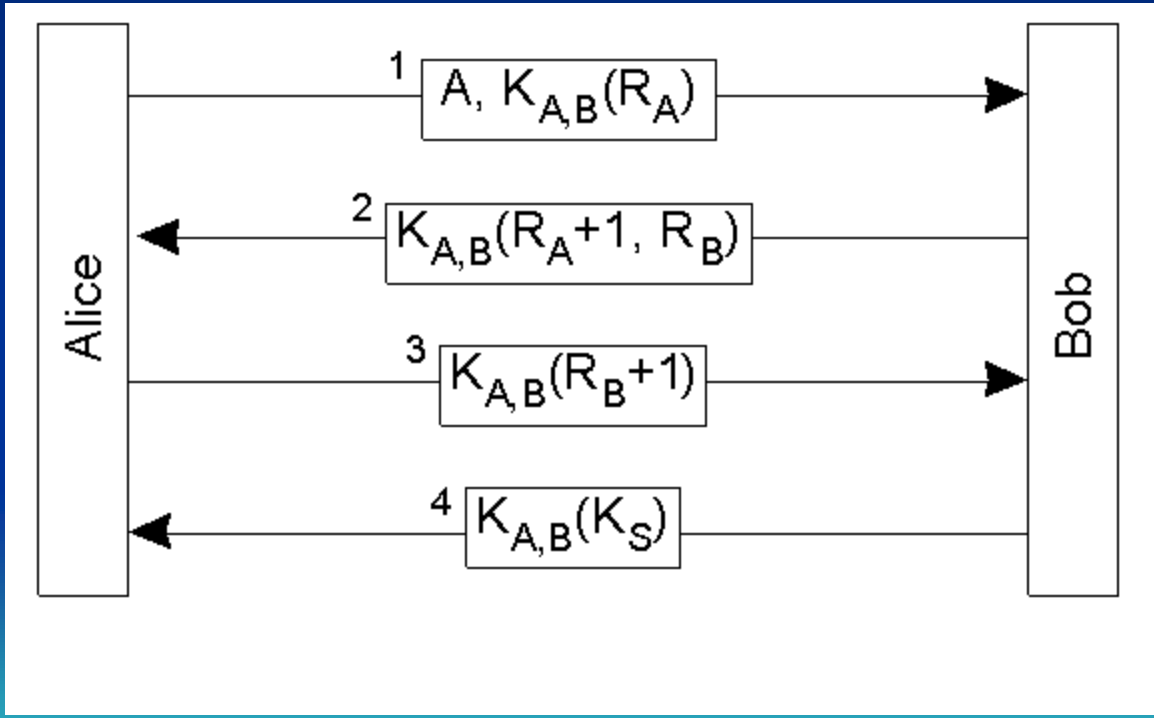
容错性-断开连接的操作（2）

- Coda使用复杂的优先级技术确保有用的数据已被缓存
- 计算优先级
  - 用户可以在储藏数据库存储路径，声明重要的文件户目录
  - 最近的文件引用信息
- 根据优先级获取文件，满足平衡状态的三个条件
  - 未缓存的文件优先级不比任何已缓存的文件优先级高
  - 高速缓存已满或没有任何未缓存的文件具有非零的优先级
  - 每个缓存的文件都是客户AVSG中维护的文件拷贝
- 储藏走查（hoard walk）:重新组织高速缓存以保持平衡

安全性-安全通道 (1)

Coda安全体系结构：

- 使用安全RPC和系统级的身份验证建立安全通道
- 处理文件的访问控制

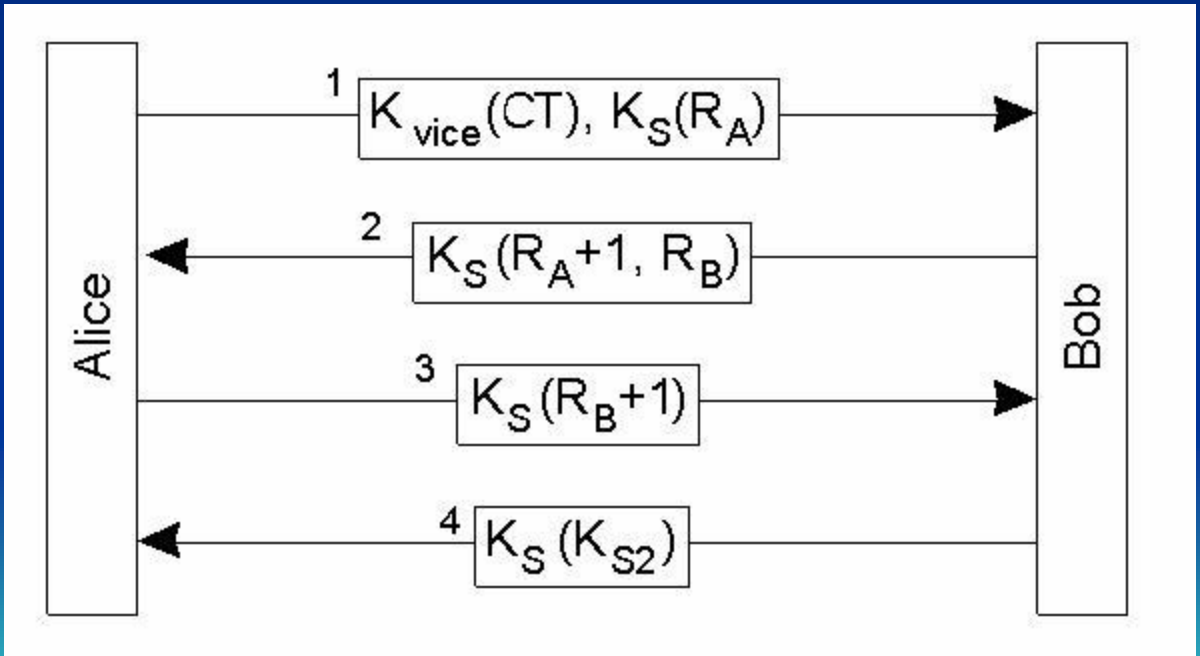


RPC2中的相互身份验证

安全性-安全通道 (2)

系统级的身份验证，类似Kerberos方式身份验证：

- 客户利用密码生成$K_{A,AS}$,从身份验证服务器AS获得身份验证令牌CT=[A,TID,Ks（会话密钥）, T_{start}, T_{end}] 和密文令牌$ST=K_{vice}[CT]$ (K_{vice}是服务器共享的密钥)
- 客户想与服务器建立安全通道时，发送密文令牌$ST=K_{vice}[CT]$
- 服务器使用K_{vice}，得到CT，从而得到Ks



在Coda 中建立 Venus client 和 Vice server 间的安全通道

安全性-存取控制

操作	描述
Read	读取目录中的文件
Write	修改目录中的文件
Lookup	查询文件的状态
Insert	向目录添加文件
Delete	删除一个已有文件
Administer	修改目录的 ACL

Coda 中的文件和目录操作分类

出于简单性与扩展性的原因，Vice服务器只将目录与访问控制列表关联

小结

- NFS和 Coda 间的比较

	NFS	Coda
设计目标	访问透明性	高可用性
访问模型	远程	上传/下载
通信	RPC	RPC
客户进程	Thin/Fat（V4）	Fat
服务器组	无	有
装入粒度	目录	文件系统（卷）
命名空间	用户私有	全局
文件 ID 范围	文件服务器	全局
共享语义	会话	事务

Issue	NFS	Coda
高速缓存一致性	回写	回写
复制	最低限度	**ROWA**
容错	可靠通信	复制与缓存
安全通道	已有的机制	**Needham-Schroeder** 基于密钥分发中心的身份验证
访问控制	大量的操作（包括文件）	目录操作

小 结

- **NFS**的共享预约。
- **NFS** 服务器的重复请求高速缓存。
- Coda的回叫承诺。
- Coda的储藏技术。