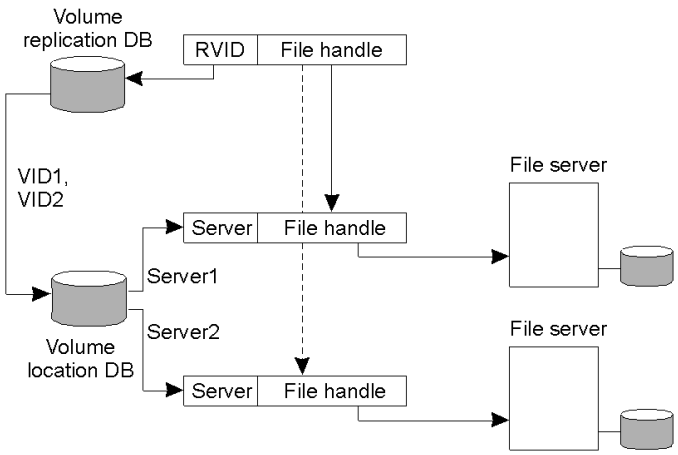
Today: Coda, xFS

- Case Study: Coda File System

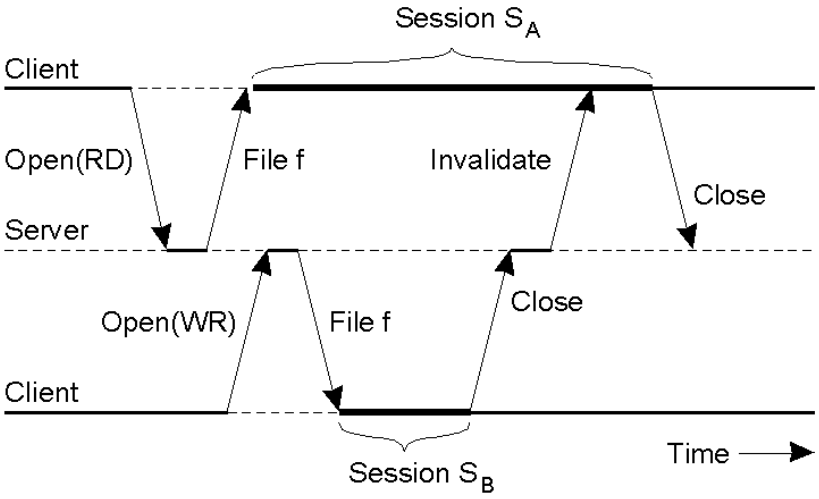
- Brief overview of other recent file systems
  - xFS
  - Log structured file systems

File Identifiers



- Each file in Coda belongs to exactly one volume
  - Volume may be replicated across several servers
  - Multiple logical (replicated) volumes map to the same physical volume
  - 96 bit file identifier = 32 bit RVID + 64 bit file handle

Sharing Files in Coda



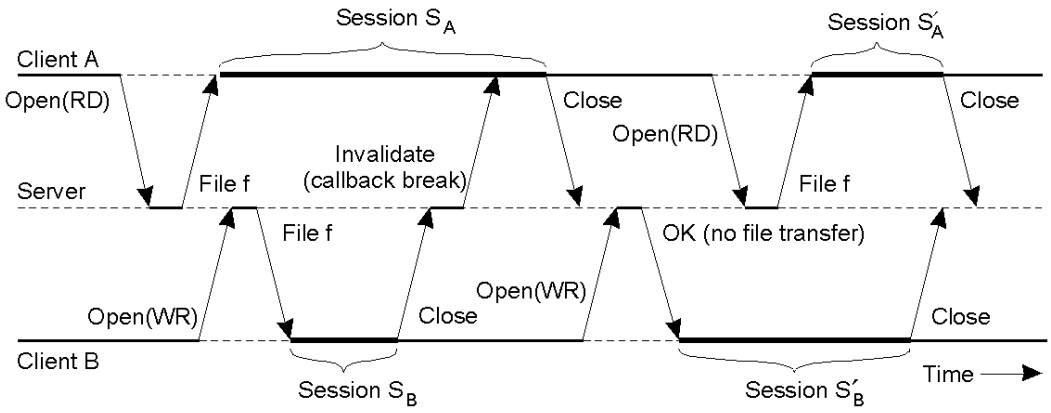
- Transactional behavior for sharing files: similar to share reservations in NFS
  - File open: transfer entire file to client machine [similar to delegation]
  - Uses session semantics: each session is like a transaction
    - Updates are sent back to the server only when the file is closed

Transactional Semantics

File-associated data	Read?	Modified?
File identifier	Yes	No
Access rights	Yes	No
Last modification time	Yes	Yes
File length	Yes	Yes
File contents	Yes	Yes

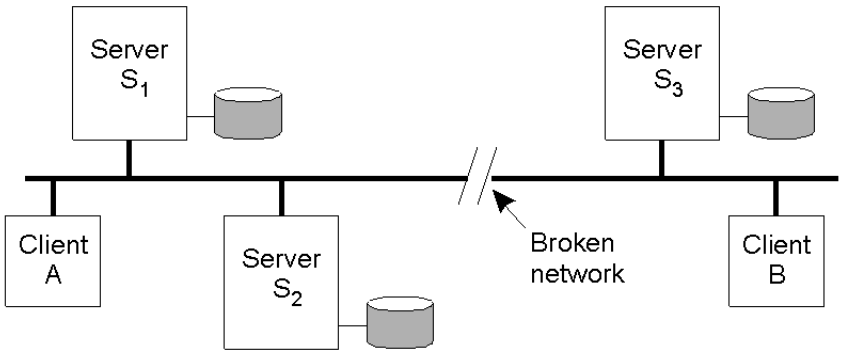
- Network partition: part of network isolated from rest
  - Allow conflicting operations on replicas across file partitions
  - Reconcile upon reconnection
  - Transactional semantics => operations must be serializable
    - Ensure that operations were serializable *after thay have executed*
  - Conflict => force manual reconciliation

Client Caching



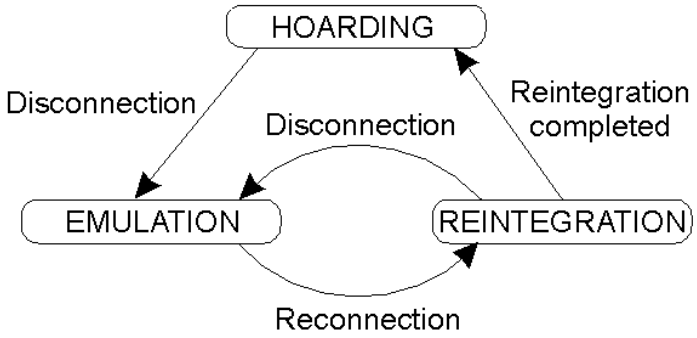
- Cache consistency maintained using callbacks
  - Server tracks all clients that have a copy of the file [provide *callback promise*]
  - Upon modification: send invalidate to clients

Server Replication



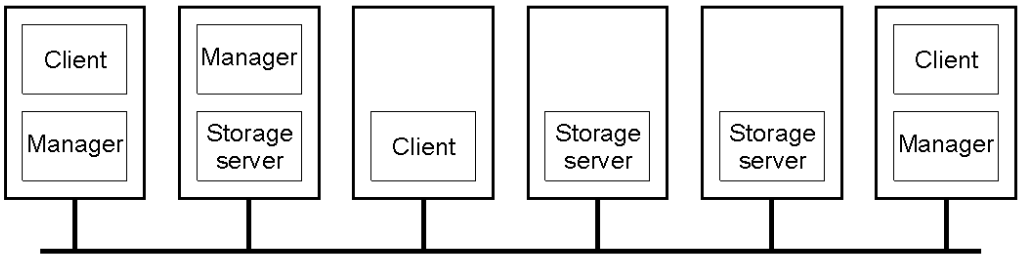
- Use replicated writes: read-once write-all
  - Writes are sent to all AVSG (all accessible replicas)
- How to handle network partitions?
  - Use optimistic strategy for replication
  - Detect conflicts using a Coda version vector
  - Example: [2,2,1] and [1,1,2] is a conflict => manual reconciliation

Disconnected Operation



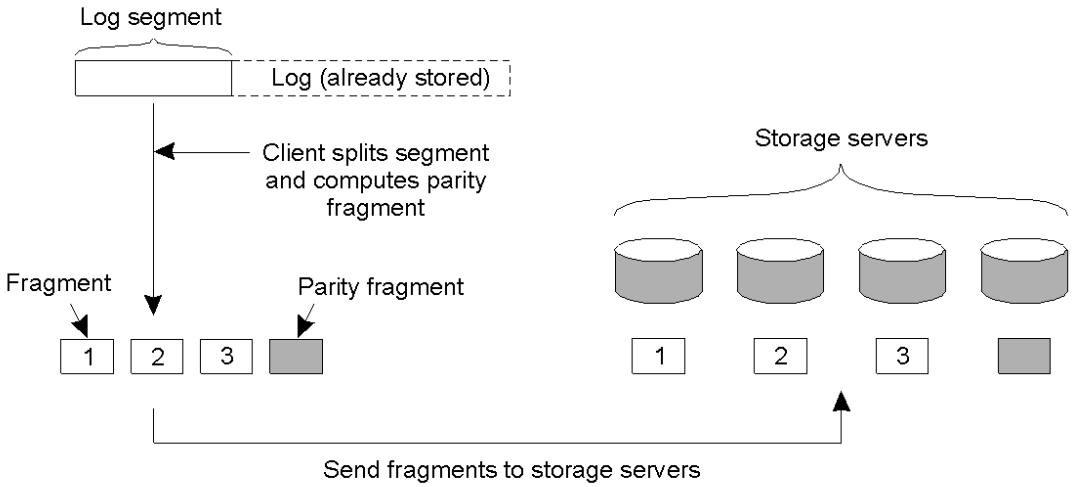
- The state-transition diagram of a Coda client with respect to a volume.
- Use hoarding to provide file access during disconnection
  - Prefetch all files that may be accessed and cache (hoard) locally
  - If AVSG=0, go to emulation mode and reintegrate upon reconnection

Overview of xFS.



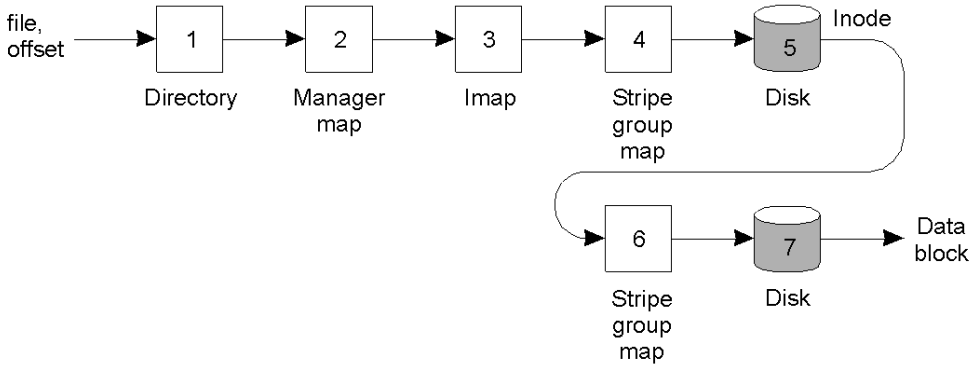
- Key Idea: fully distributed file system [*serverless* file system]
- xFS: x in "xFS" => no server
- Designed for high-speed LAN environments

Processes in xFS



- The principle of log-based striping in xFS
  - Combines striping and logging

Reading a File Block



- Reading a block of data in xFS.

xFS Naming

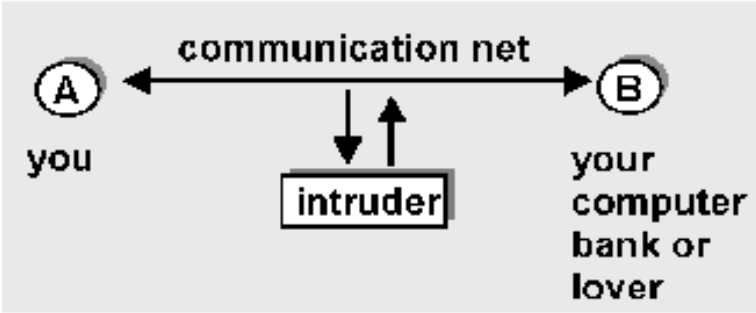
Data structure	Description
Manager map	Maps file ID to manager
Imap	Maps file ID to log address of file's inode
Inode	Maps block number (i.e., offset) to log address of block
File identifier	Reference used to index into manager map
File directory	Maps a file name to a file identifier
Log addresses	Triplet of stripe group, ID, segment ID, and segment offset
Stripe group map	Maps stripe group ID to list of storage servers

- Main data structures used in xFS.

Security in Distributed Systems

- Introduction
- Cryptography
- Authentication
- Key exchange
- Readings: Tannenbaum, chapter 8

Network Security



Intruder may

- eavesdrop
- remove, modify, and/or insert messages
- read and playback messages

Issues

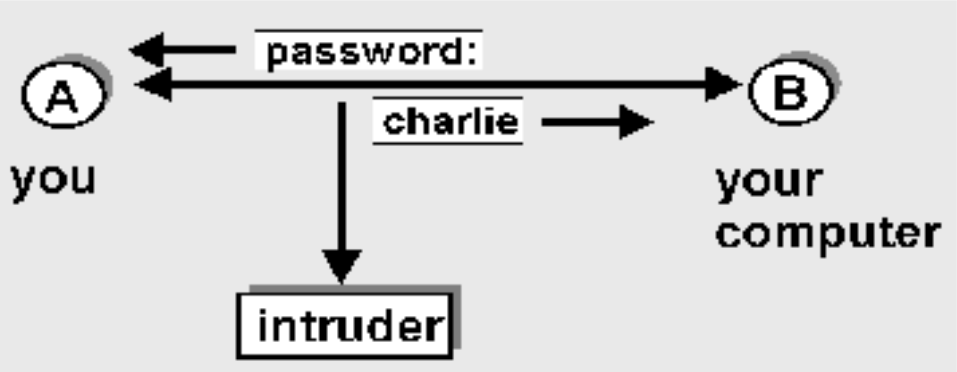
Important issues:

- *cryptography:* secrecy of info being transmitted
- *authentication:* proving who you are and having correspondent prove his/her/its identity

Security in Computer Networks

User resources:

- login passwords often transmitted unencrypted in TCP packets between applications (e.g., telnet, ftp)
- passwords provide little protection



Security Issues

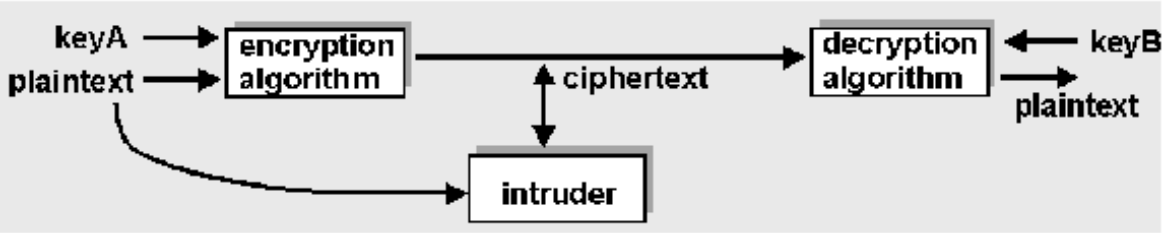
Network resources:

- often completely unprotected from intruder eavesdropping, injection of false messages
- mail spoofs, router updates, ICMP messages, network management messages

Bottom line:

- intruder attaching his/her machine (access to OS code, root privileges) onto network can override many system-provided security measures
- users must take a more active role

Encryption



plaintext: unencrypted message

ciphertext: encrypted form of message

Intruder may

- intercept ciphertext transmission
- intercept plaintext/ciphertext pairs
- obtain encryption decryption algorithms

A simple encryption algorithm

Substitution cipher:

```
abcdefghijklmnopqrstuvwxyz
poiuytrewqasdfghjklmnbvczx
```

- replace each plaintext character in message with matching ciphertext character:

plaintext: `Charlotte, my dear`

ciphertext: `iepksgmmy, dz uypk`

Encryption Algo (contd)

- key is pairing between plaintext characters and ciphertext characters
- **symmetric key:** sender and receiver use same key
- 26! (approx 10^26) different possible keys: unlikely to be broken by random trials
- substitution cipher subject to decryption using observed frequency of letters
  - 'e' most common letter, 'the' most common word