nDrive Sync

Design Draft

The problem

- Synchronize one or more clients with our storage server
  - Clients can be: Desktops, Tablets
- Storage server is also mounted as a network folder on ‘nDesktop’
- Storage needs to be exposed via FTP

Desktops

Tablets

Sync Server

FTP Server

Terminal Server

SMB

File Server

Backup and Geo-distribution

what have we tried

- ftp
- rSync
- git
- a mixed approach (not tried yet)

the ideal sync solution

new

edits

moving

rename

copying

conflicts

versioning

resume transfer

no polling

Disc conserving

CPU conserving

ftp

new

edits

moving

rename

copying

conflicts

versioning

resume transfer

no polling

Disc conserving

CPU conserving

rSync

new

edits

moving

rename

copying

conflicts

versioning

resume transfer

no polling

Disc conserving

CPU conserving

git

new

edits

moving

rename

copying

conflicts

versioning

resume transfer

no polling

Disc conserving

CPU conserving

what we don't need is...

Git

- multiple copies of the file
- compute actual file delta on each commit
- no support for empty directories

rSync

- heavy computation on server (if server is sender)
- reliance on timestamp for identifying sync direction

what we really need is...

- Goodness of all worlds
- intelligence of git
- lightness of ftp
- efficiency of rsync
- a way to avoid polling

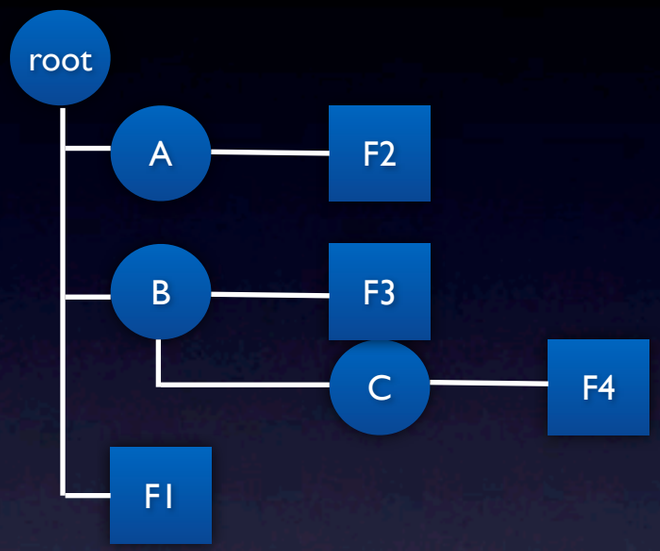
a mixed approach

- revision numbers and checksums like git
  - helps us identify which files changed
  - allows us to optimize for copy/move etc
  - removes reliance on time-stamps
  - can detect conflicts
- differential sync as in rSync
- alter rSync to offload compute to client
- notification protocol for connected clients

the basics

- each artifact (directory/file) has an entry in a table
- the table is basically a tree of checksum lists of the artifacts
- checksum list for each artifact is a list of historical checksums for that artifact

example...



- Hash:
  - checksum of a file determined by it's contents
  - checksum of a directory determined by checksums of it's files
- CHash is determined by hashing Hash and the artifact's own name
- the CHash and Hash fields are in fact pointers to a linear linked list containing the hashes in reverse chronological order. So, the most recent hash would be at the head and nodes moving forward in the list would represent historical hashes of the artifact

Id	Name	P	CHash	Hash
1	root		43214432	345345l5
2	A	1	23412523	23423145
3	B	1	45456223	72342335
4	F1	1	14546235	32432135
5	F2	2	25432423	56854623
6	F3	3	63763467	13432982
7	C	3	85867542	92431723
8	F4	7	76985642	68643674

sync init

- client establishes connection and requests for server table
- client does a tree comparison and identifies mismatches according to CHash
- each mismatch may require sync
- for each mismatch, to determine sync direction
  - try to look up server's CHash in the CHash list of the client artifact
  - if the CHash is found in the list, the artifact must have changed at the server
  - if not found, send client's CHash to server and look it up in server's CHash list
  - if the CHash is found in the list, the artifact must have changed at the client
  - if CHash-es of client and server are not found in other's list, it is a conflict i.e. change at both ends

some optimizations

- if the CHashes are different, but the Hash is same, it must be a rename
- if a folder's CHashes/Hashes match, it is not required to compare it's children

sync operation

- after determining the sync direction
  - run modified rSync
    - request server for it's block checksums
    - carry out rolling checksum calculation at client
    - request/send data after each mismatching block is identified

connected client

- through the open tcp connection, server notifies the client if there is any change on the server's file system.
- sync is carried out for the artifact mentioned in the notification
- if there is a change on the client, similar procedure is followed for the changed object

further optimizations

- with file-system hooks (DirectoryWatcher)
- at each file create event, we can check the table to see if the new file is a copy of existing one