SwiftFS: A CRDT Filesystem

Annette Bieniusa

Joint work with
Marc Shapiro, Marek Zawirski (INRIA & LIP6)
Nuno Preguiça, Sérgio Duarte (UNL)
Stéphane Martin, Pascal Urso (LORIA)

Distributed file system

- „Classical“ problem in Distributed systems
- Covers lots of interesting aspects
  - Scalability
  - Usability
  - Diversity of elements
  - Latency
  - ...

Distributed file system

- „Classical“ problem in Distributed systems
- Covers lots of interesting aspects
  - Scalability
  - Usability
  - Diversity of elements
  - Latency
  - ...

Is it **possible** to build a file system on top of Swiftcloud only using CRDTs?

Files....

- Sequence CRDT for text files
  - Logoot and Treedoc variant
  - Allows fine-granualar merge/update
- Register CRDT for (paged) data blob
  - Files without mergeable content

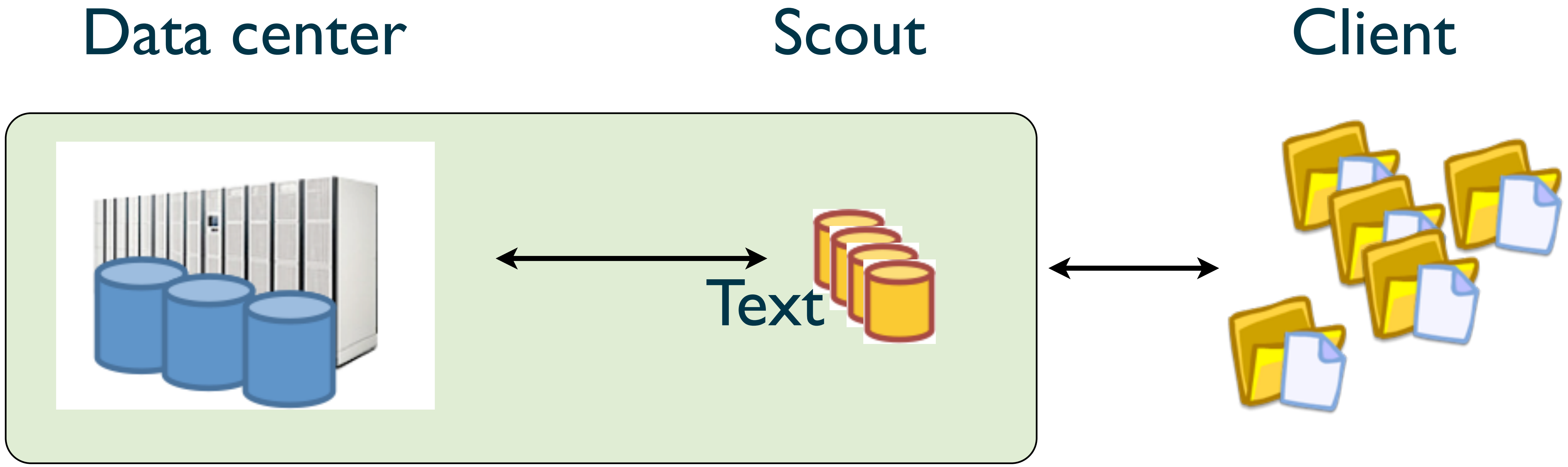
... and Directories

- Based on *recursive* CRDT
- Represented as Map: (name, type) $\rightarrow$ object
- Recursive merge only for objects of the same type, following Unix conventions

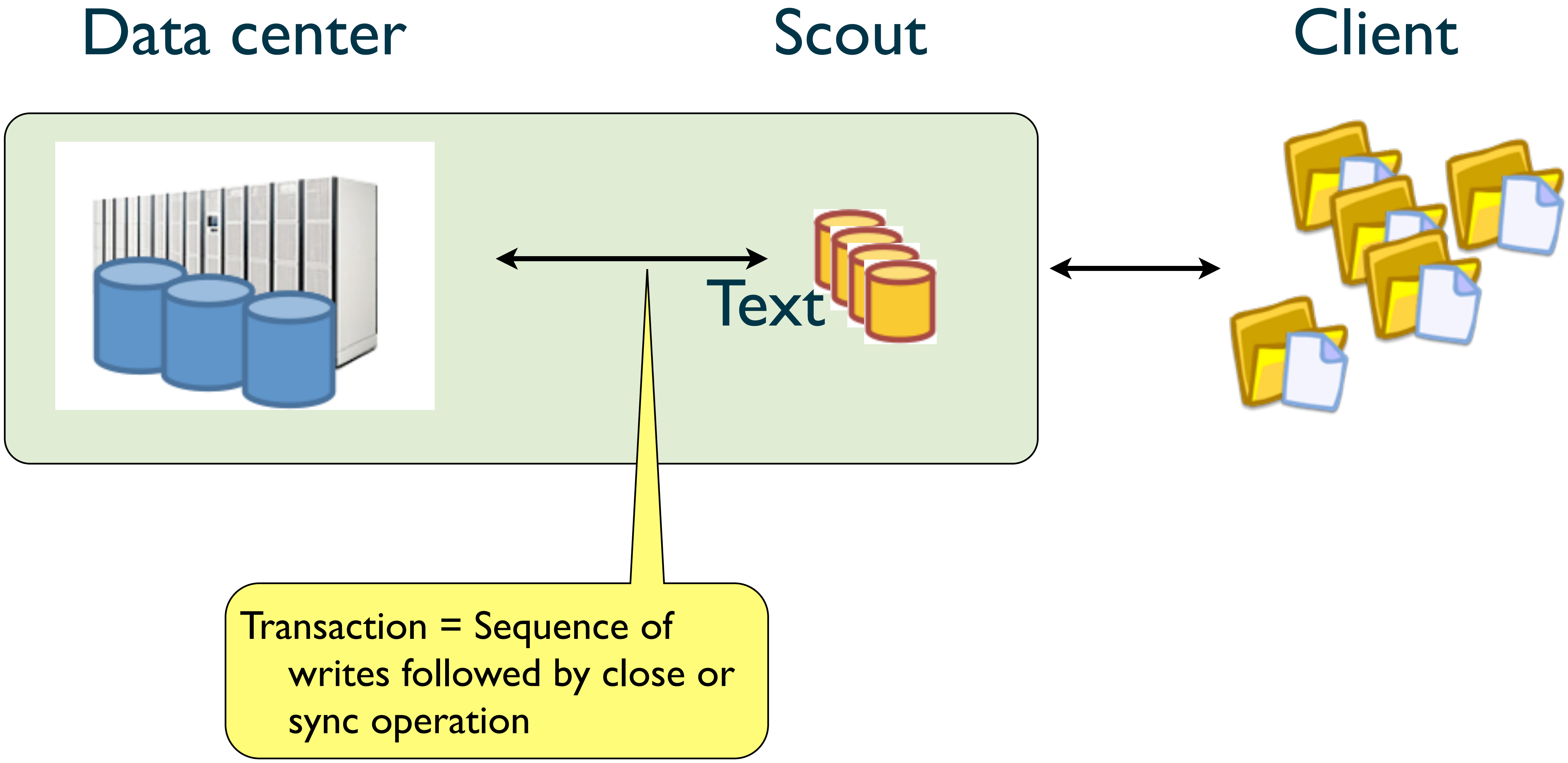
Operations

- *create_entry(name, type, value)*
  - Concurrent: merge subdirs recursively
- *get_entry(name,type) : value*
  - Obtain file content/directory listing
- *modify_file(name, type, value)*
  - Concurrent: merge file content
- *remove_entry(name, type, value)*
  - Concurrent: deletion dominates
  - Changes can be retrieved from history
  - Subdirectories are removed recursively

SwiftFS



SwiftFS

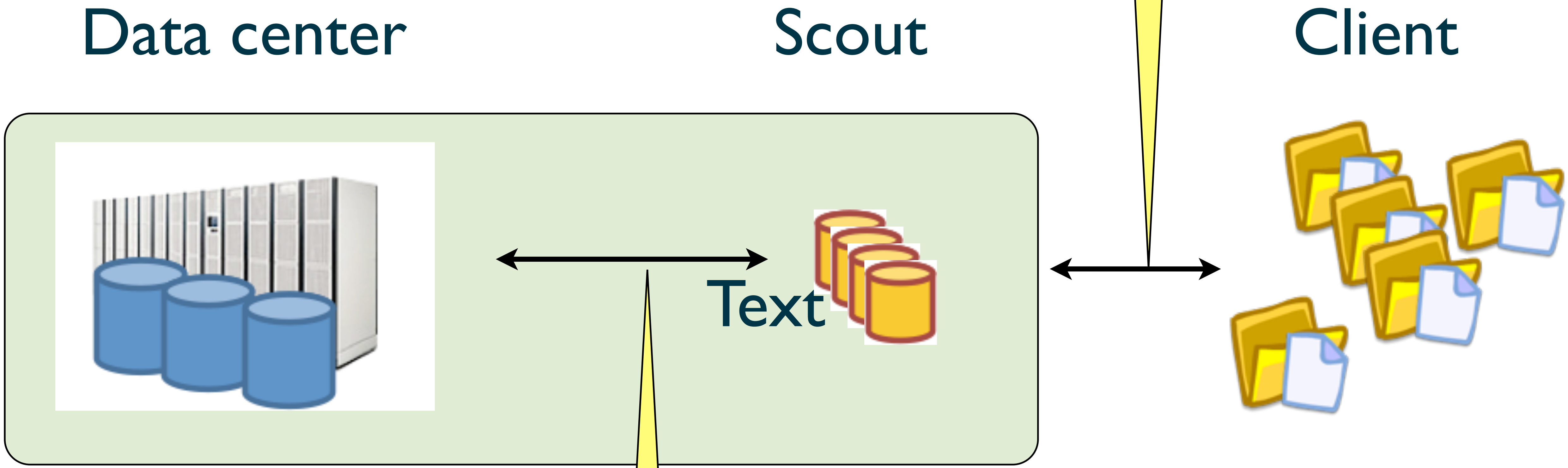


Transaction = Sequence of writes followed by close or sync operation

SwiftFS

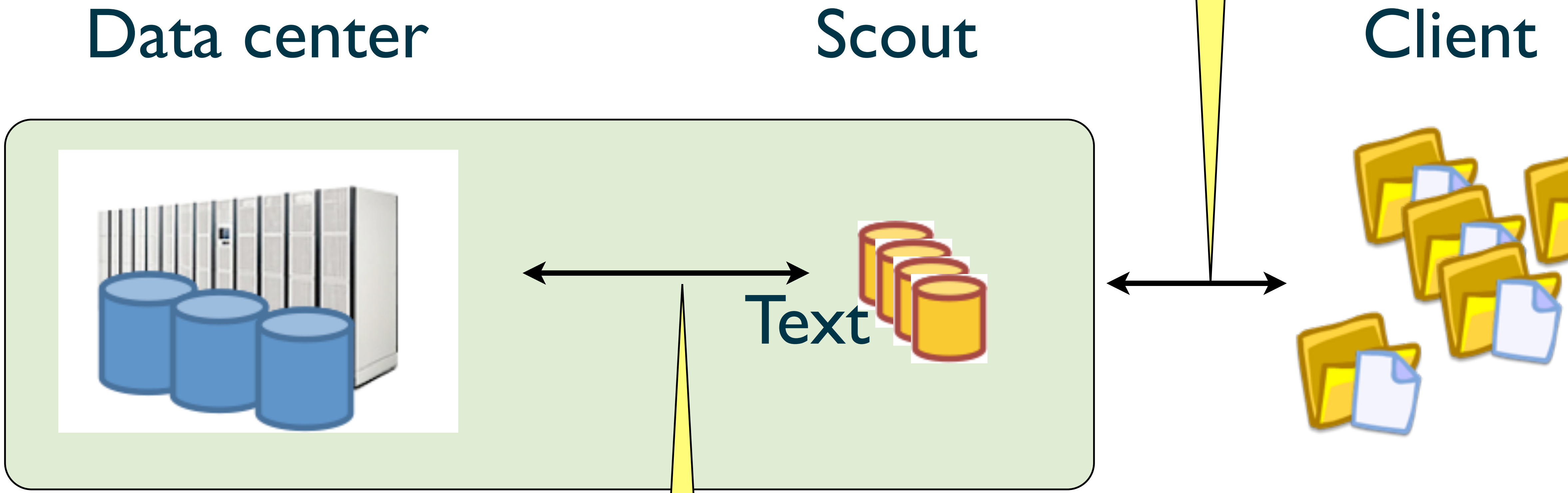
Currently only simple RPC for Client/Scout communication

Also possible to use NFS, Dropbox, ...



Transaction = Sequence of writes followed by close or sync operation

SwiftFS



Currently only simple RPC for Client/Scout communication

Also possible to use NFS, Dropbox, ...

Transaction = Sequence of writes followed by close or sync operation

File contents and attributes are cached locally at client;

Cache is emptied every minute or when invalidated

SiftFS

- Implementation of FUSE interface
  - Allows to use it as any other mounted file system
  - Bindings for Windows, Unix, MacOs,...
- Future work: (Extended) Attributes and Permissions

Evaluation

Andrew Benchmark (without compilation part)

- 550KB of data, 75 files in total (small text files)
- folder depth: 3

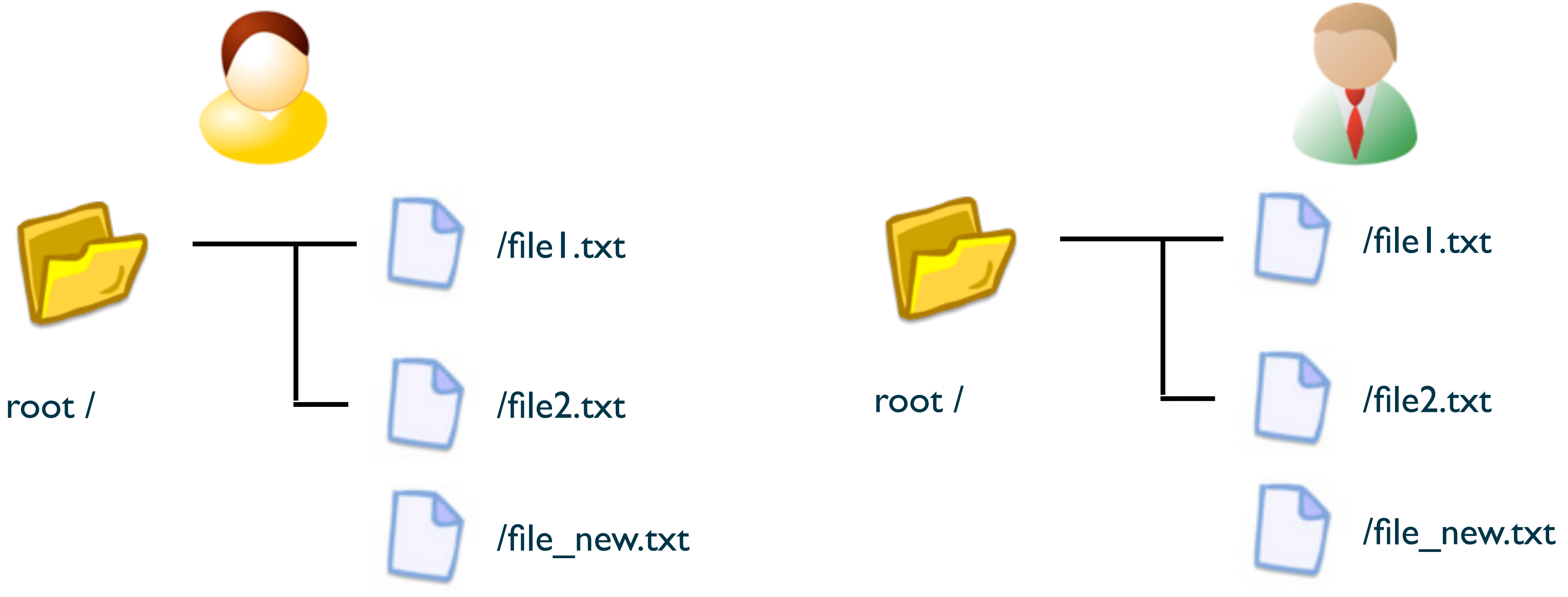
	MakeDir	Copy	ScanDir	ReadAll
1 scout x 1 client	3s	30s	2s	16s
5 scouts x 10 clients	6s	37s	8s	34s
Client@Scout	4s			
Local FS	3s			

Latency: Client -> Scout: 33ms; Scout -> DC: 10ms, for Client@Scout -> DC: 28ms
Benchmark executed in loop, each on different subdirectory
Files as Register CRDT
Txn mode: Repeatable read, async commit

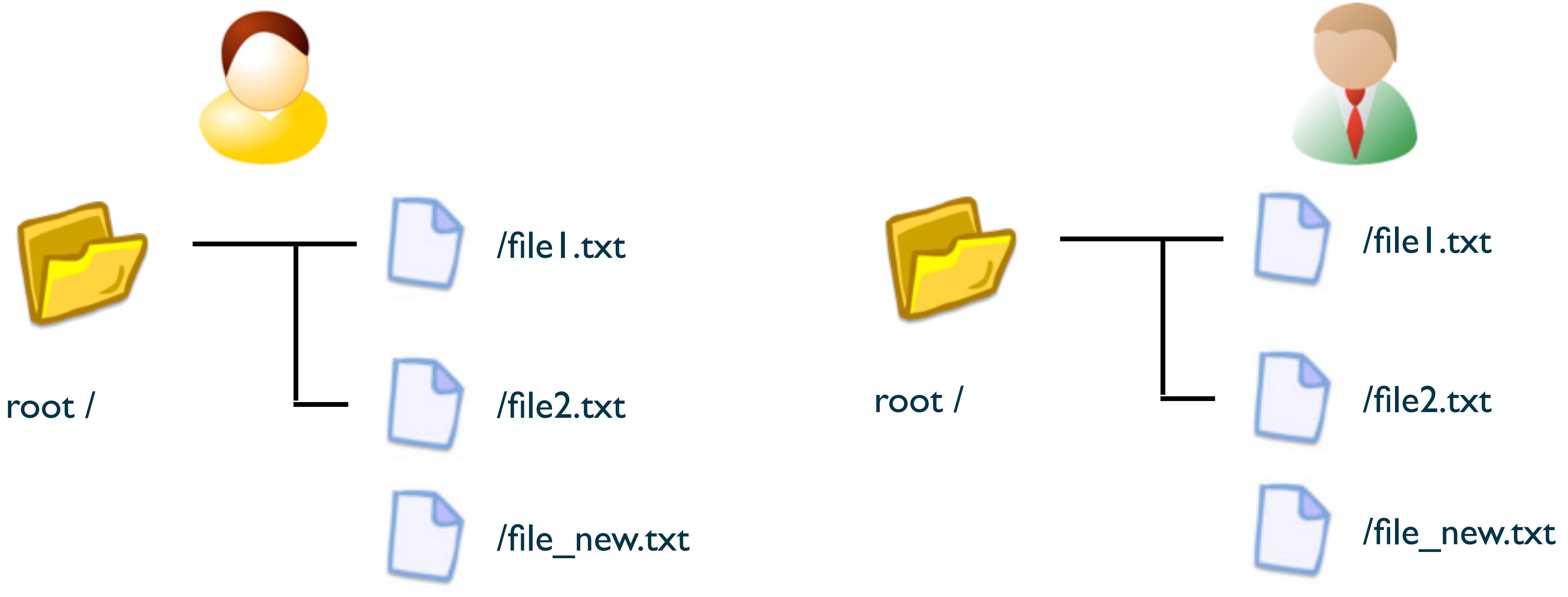
Outlook: SwiftGIT

- Possible to extend SwiftFS to Git-like version management system
- Transactional updates to directory and files
  - Possible to only checkout single parts of shared directory structure
- Versioning allows „travelling back to the past“
- More flexibility for content
- Customizable merge

Problem: Concurrent Creation



Problem: Concurrent Creation



How to identify this shared object?

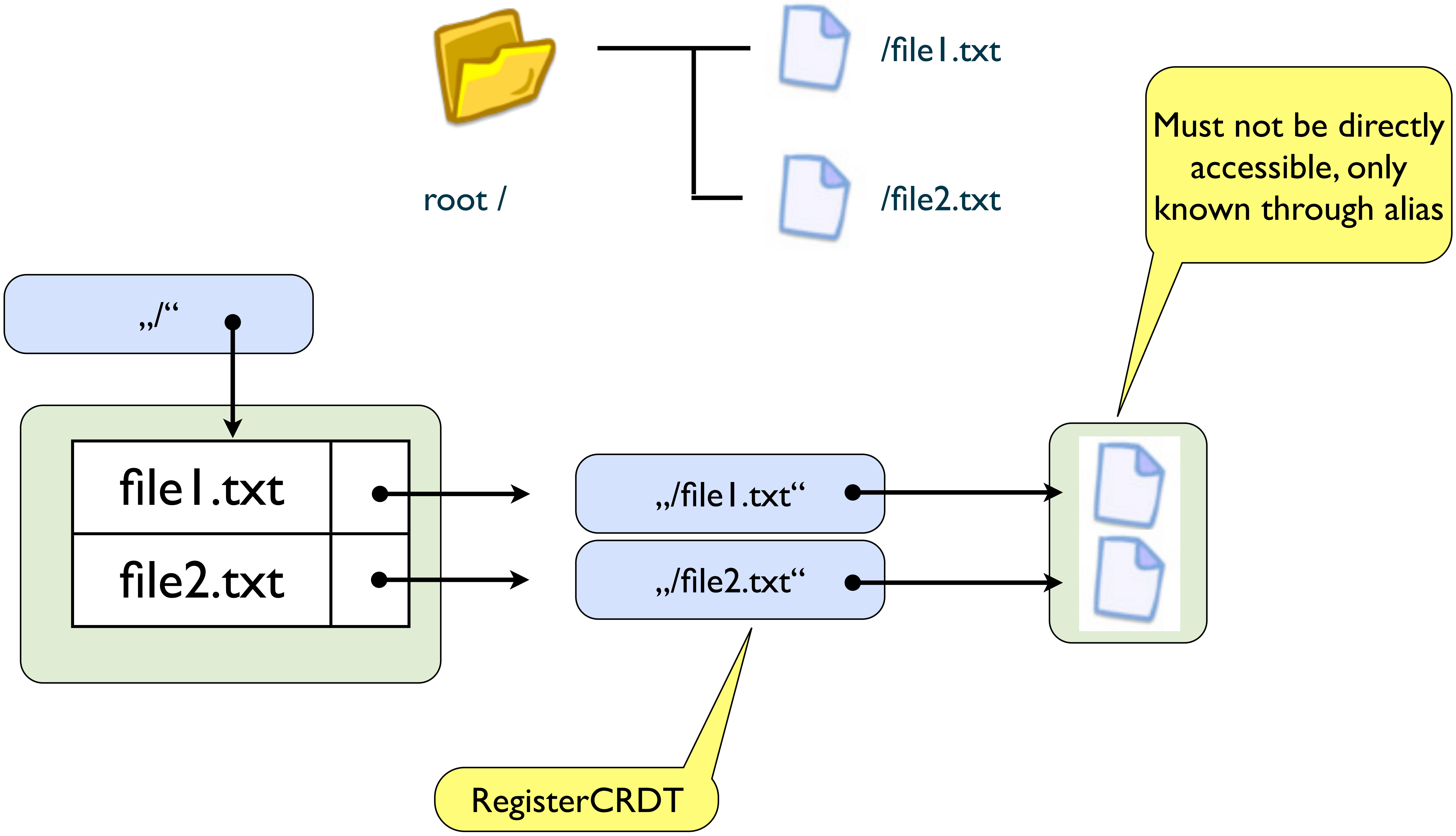
Options

- Centralized Name Server
  - not CRDT-style ...
- „Flat Nesting“
  - Make the file part of the Filesystem CRDT
  - Inflates the CRDT payload
- Global naming scheme
  - Identify entries by (unique) path name
  - Renaming?

Some more advanced option

- „Inode technique“
  - Used in many Unix filesystems
  - Add indirection from name to file content (+ meta data)
  - Simplifies moving, renaming, links
  - Requires indirection of updates (will be tricky in Swiftcloud...)

Wishlist: Inode CRDTs



Summary

- CRDT directory
- Different file types
- Support operations that do not require strong synchronization
  - Next step: move
- Simple global naming scheme for identifying CRDTs
  - Limitation of current Swiftcloud implementation