More CODA replication

- Replica States may be transformed by
  - Updates
    - Storing files on closing, creation/deletions etc.
    - 2 Phase protocol: First each site in AVSG checks LSID and CVV of client. If they are equal to or dominate its own LSID and CVV, then it commits the change. Then, CVVs at servers are updated to reflect the clients view of who committed the change.
  - Forces
    - Server-Server Protocol. Basically replays the changes at the dominant site not done at submissive site already.
  - Repairs
    - 2 Phase operation to resolve inconsistencies, or to recover from crashes.
    - Automatic vs. User Intervention
  - Migrates
    - Create a covolume of inconsistent data.

Coda Cache Coherence

- Basic Cache operation involves caching file (and directories) on demand!
- Due to Optimistic replication, the Cache Manager (VENUS) also has to be aware of:
  - AVSG enlargement : missing members of VSG are contacted once every tau seconds. If AVSG is enlarged, callbacks are dropped and next reference to data causes fresh fetch and callback reestablishment.
  - AVSG shrinkage: AVSG members from which data is cached probed every tau seconds. If prefered server is lost then drop callbacks.
  - Problem – since callback only on prefered server, what if my prefered server is not in other clients AVSG
    - Solution: when probing, ask for volume CVV, and compare. Drop callbacks as needed.

Disconnected Operation

- Operate on cached data. As long as data is cached everything is fine
- Cache miss is bad, since there is no way to make it transparent to the user.
  - Avoid! Besides LRU replacement, allow user to tag certain files as “sticky”
- Reintegration: Upon reconnection, replay your operations – force for files, more complex for directories. If there are inconsistencies, co-locate the volume and try to fix.

Sundry stuff

- Please Read section 9.5.5
- Log Structured File Systems (9.6)
  - Basic problem – multiple seeks needed to do read/writes, and this costs time
  - Solution – cache file (directories) in memory. Do all updates in memory. Write once to disk the “log” of all changes, including data and metadata.

NOW/xFS

- Designed to work on workstation clusters (fast connections, trusted environment)
- Serverless architecture
- Uses software RAID, Coperative caching, and LogFS
- Manager Maps, imap (index node), Directories(name to index), stripe map
- Read: Try local cache, contact manager, try finding someone caching it, else read from disk.
- Write: Logging
- Cache Consistency: Token based.