Disconnected Operation in the Coda File System ('91)
Kistler, Satyanarayanan

What are the paper's goals?

- An early examination of disconnected replication, and implications

Major contributions:

- given whole-file caching, disconnected operation is easy
    - allow access to cached copy in period of disconnection ("emulation")
    - implies callbacks will break, won't learn of new copies
    - might be reading stale data (r/w conflict)
        - tricky: leads to cascading conflicts – read from stale file, write to different file – different file is "tainted"
        - causal connection
    - might diverge (w/w conflict)

- disconnected operation has all the same problems as cache coherency – the major difference is what you do if a server is unavailable
    - Ivy
        - Block everything to maintain coherence
    - Coda?
        - Let things continue, clean up mess later

- notion of hoarding
    - FS automatically determines what you want given your usage pattern
    - not clear this worked then
    - clear it can likely work now – replicate all files, pretty much
- log writes during disconnection
    - keep track of side-effects to play back
    - log writes? why not just dirty bit on modified files?
        - so coda could only send deltas, not full file, on reconciliation
- reintegration by replaying logs at servers
    - single transaction: any conflict and it fails, punts to human!
    - conflict checking via version numbers
    - possible to do better
        - update files that don't conflict
        - need to worry about larger-level consistency issues
        - causal consistency and tracking as one way
        - hard problem! Leads to notion of transactions

    - conflict resolution
        - entirely up to human in coda
        - need better way
            - ask human which version to keep?   (apple isync)
            - figure out what semantically makes sense for application (bayou: calendar, mail)
            - record-level merge?   (cvs)

Evaluation

- about 3MB/hour of dirty data produced during disconnected operation
    - jives with other FS studies – a few megabytes to tens of megabytes per day
    - getting bigger with big read-only files (media, powerpoint, etc.)
- sequential write sharing is rare – 0.5% modifications are by different writer than previous.
    - what are implications of sequential write sharing? potential conflict if either writer is disconnected
- concurrent write sharing is typically non-existent or super-duper-rare
    - implications? conflict even if connected!
    - has this changed?

Questions

- Comparison between coda-style hoarding/reintegration and DVCS?

- Is there a better way to handle conflicts, maybe automatically?
    - Bayou says yes.
        - think PIM (calendar. address book etc.) sync'ing
        - think IMAP
        - typically rely on app-specific semantics, and even still, have to punt to user from time to time

- is disconnected operation relevant anymore?

- client-server integration vs. p2p integration?
    - bayou