CS-438 Decentralized Systems Engineering

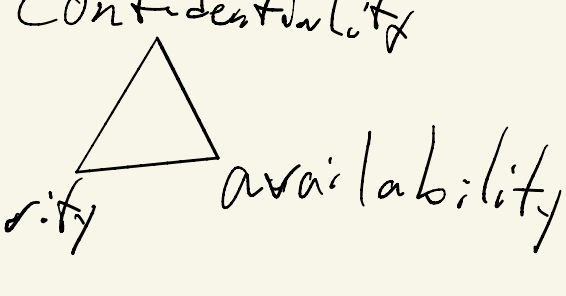
Week 7

Distributed Storage

Challenges:

- How to organize?
  - Logically? "flat", files, dirs, database, graph, ...
  - Physically? where stored?
- Availability, resilience to data loss (eg replication)
- Scalability, load-balancing: how to divide up work?
- How to share data / files
- How to maintain consistency? (synchronization)
- Confidentiality: access control, logging, accountability

CIA triad:



Centralized

file server ← client

file access API: open, close, read, write, seek, ...

- NFS, SMB, 9P
  File API-based
- Cloud-based either or simplified:
  - append-only log ("blockchain")
  - read-only (SFSRO, HTTP, BitTorrent)
  - block stores

advantages: simplicity

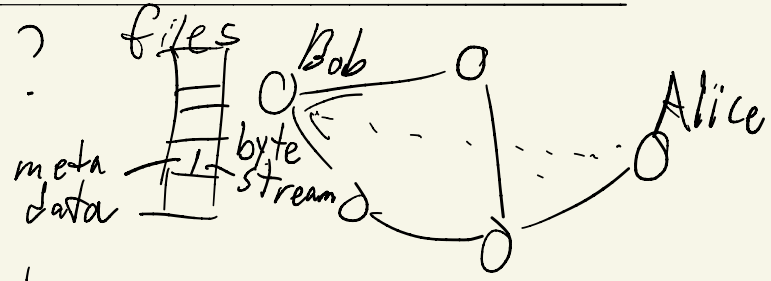
disadvantages: not that simple, server bottleneck, single point of failure

Decentralized

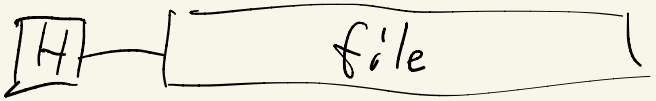
- P2P file sharing (Napster, Gnutella, BitTorrent, IPFS)
- DHT-based sharing (IPFS), block storage (CFS)
- Blockchain: Bitcoin, Ethereum, etc. (strongly consistent)

Distributed sharing of read-only files, directories

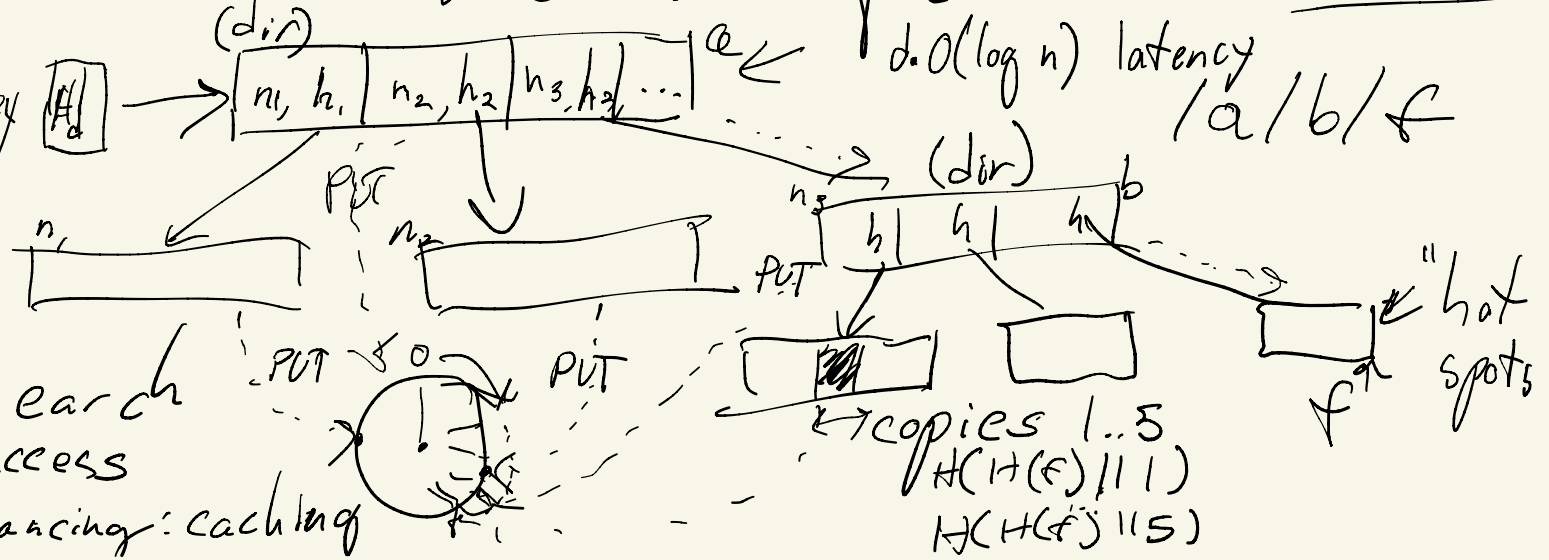
- P2P systems - Napster / Gnutella?
  - download file-at-a-time
  - rich directory structure: ZIP, tar, ...
  - problem: random access?



- File/block-level distribution, access
  - Cryptographic hashes as unique content IDs



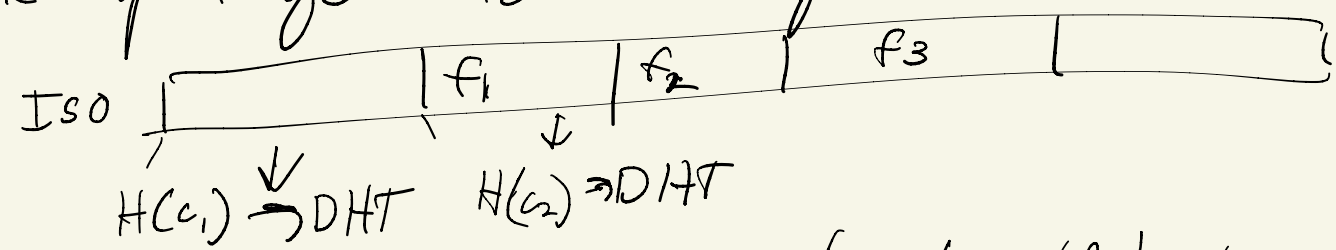
  - Merkle trees: hashes as unique IDs of metadata



  - DHTs search
    - random access
    - load balancing: caching

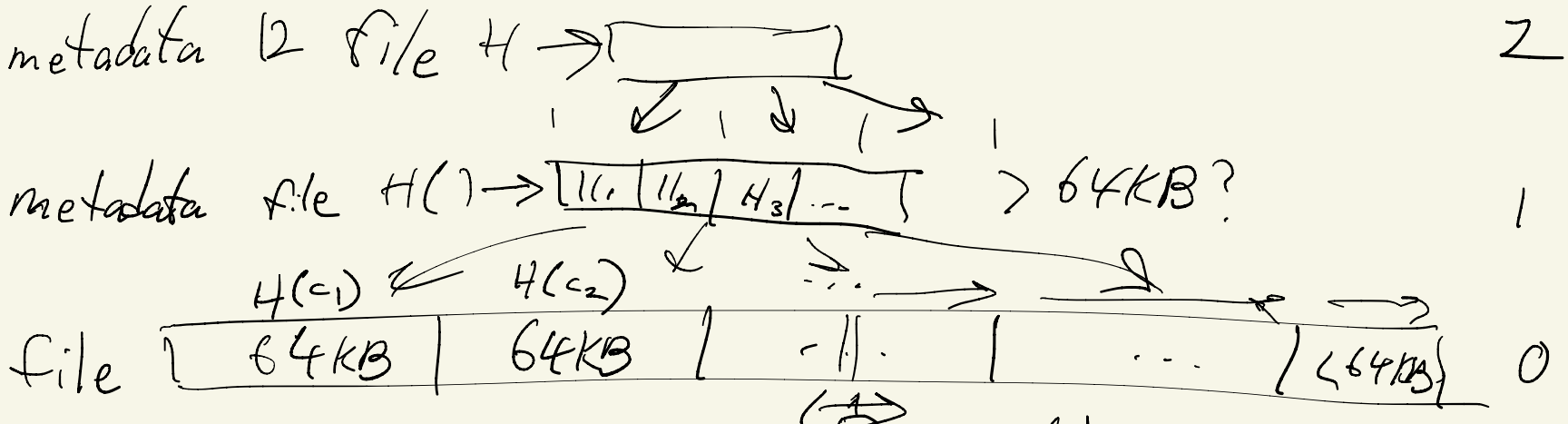
Block-level distribution

- Random access to parts of (large) files?
  - break up large file according to its internal structure



have to know structure

  - break into fixed-size chunks / blocks



  - break into variable-size chunks (oblivious to structure)?
    rolling checksums (rsync)