Peer-to-Peer Storage Systems

- Cooperative File System (CFS)
- PAST
- Freenet
- FreeHaven

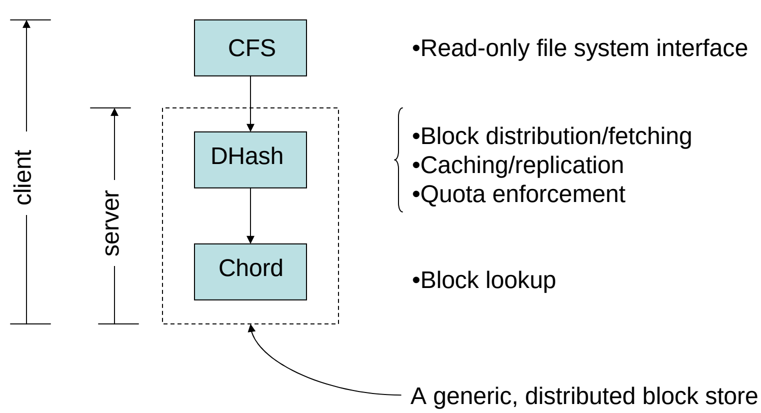
Peer-to-Peer Systems

- *Definition*: "Peer-to-peer systems can be characterized as distributed systems in which all nodes have identical capabilities and responsibilities, and all communication is symmetric." *–Rowstron-*
- *Popular Examples:*
  - Napster
  - Gnutella
- *Goals (from Dabek, et. al.)*
  - Symmetric and decentralized
  - Operate with unmanaged voluntary participants
  - Fast location of data
  - Tolerate frequent joining/leaving by servers
  - Balanced load

CFS: Properties

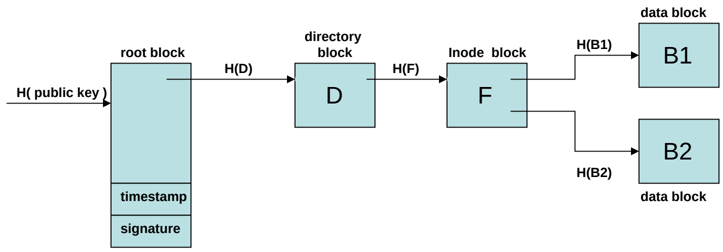
- Decentralized control (use ordinary Internet hosts)
- Scalability (overhead at most logarithmic in the number of servers)
- Availability (placement of replicas on unrelated servers)
- Load balance (block distribution and caching)
- Persistence (renewable lifetimes)
- Quotas (source-limited insertions)
- Efficiency (comparable to FTP access)

CFS: Architecture

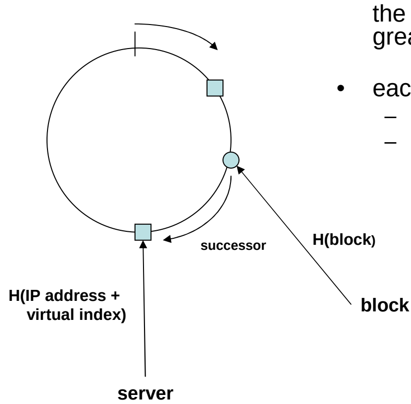


CFS: Content-hash indexing

- Each block (except for the root block of a file system) is identified by an index obtained from a hash (e.g., SHA-1) of its contents
- A root block is signed by the author; the index of the root block is a hash of the user's public key

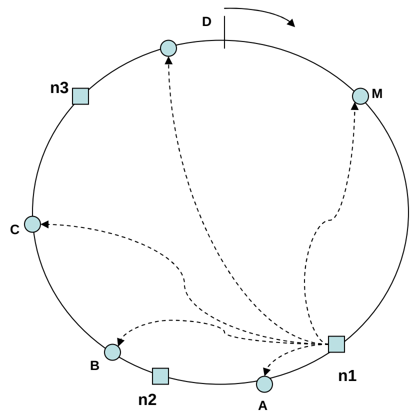


Chord: Mapping



- server *s* stores all values indexed by key *k* for which *s* is the successor of *k* (*successor(k)* is the node whose identifier is the smallest one greater than *k*)
- each Chord server maintains two lists:
  - a *finger table* for searching
  - *r* immediate successors and their latency information

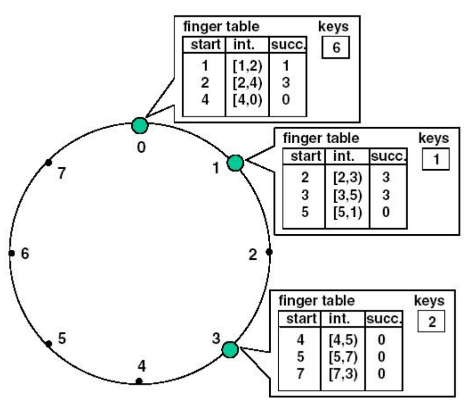
Chord: Searching (1)



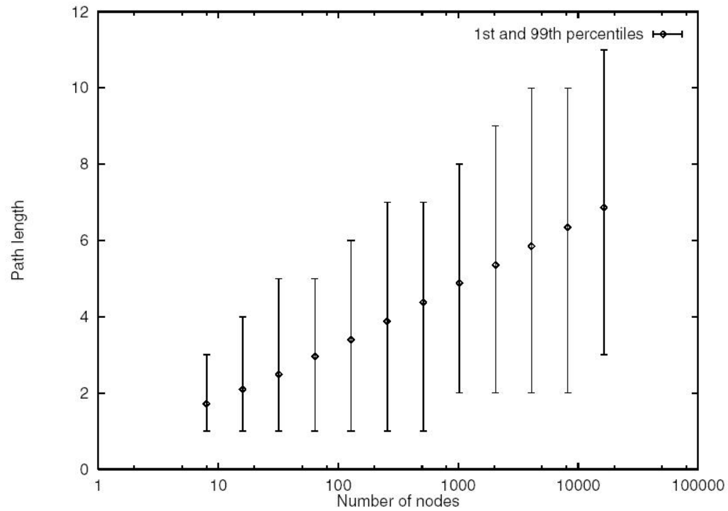
finger table for n1

start	interval	node
A = n1 + 2^0	[A,B)	n2
B = n1 + 2^1	[B,C)	n3
C = n1 + 2^2	[C,D)	n3
…		
M = n1 + 2^{m-1}		

Chord: Searching (2)



Chord: performance



Chord: Adding Servers (1)

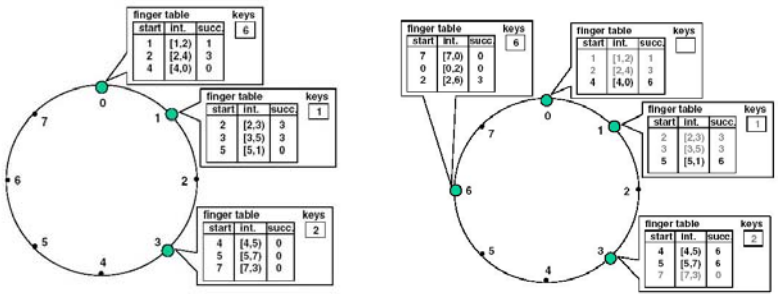
Two Invariants maintained:

- Successor information is correct
- Successor(k) is responsible for key k

Steps:

1. By out-of-band means, locate an existing server, n
2. Update tables
   - Update successor/predecessor links
   - Creates finger tables for new server
   - Update other server's finger tables
3. Redistribute responsibility for keys to n from its successor
   - Call higher (DHash) layer

Chord: Adding Servers (2)



Adding a new node at 6 assuming that node 6 knows, by out-of-band means, of node 3

DHash: Interface

- *put_h(block)* – stores *block* using content-hashing
- *put_s(block, pubkey)* – stores *block* as a root block; key is hash of *pubkey*
- *get(key)* – finds/returns block associated with *key*

DHash: replication

- Places replicas on k servers following successor
- Note: each Chord server maintains a list of r immediate successors. By keeping r >= k, it is easy for DHash to determine replica locations
- Existence of replicas eases reallocation when node leaves the system
- By fetching the successor list from Chord, the DHash layer can select the most efficient node from which to access a replica of a desired block

DHash: caching, load balancing, quotas

- Caching is effective because searches from different clients converge toward the end of the search
- Virtual servers hosted on one machine allow for more capable machines to store a larger portion of the identifier space
- Each server enforces a fixed, per-IP address quota on publishing nodes

DHash: replication and caching