Scribe notes from William Cook, handwritten pages attached

June 4, 2002

Naming Schemes

- DNS
- FreeNet
- Chord
- LDAP

- What are the application criteria for naming systems?

+ Performance
  - Fast lookup
  - Fast update
  - Scalability (clients, queries, names)
  - Updating
  - Wildcard search
  - Find nearest

+ Management
  - Delegation / no central authority

+ Reliability
  - Accuracy
  - Availability

+ Security
  - Authentication of updates
  - Access control for lookups

+ Ubiquity
  - Portability

+ Privacy
  - Plausible deniability

+ Context sensitive naming
  - Flexibility
  - Routing name to multiple address
  - Routing multiple names to same address

    Name1, Name2 } load balanced servers

+ Simplicity
  - Keep low level simple

+ API
  - Lookup
  - Content-based (ala Internet Keywords)
  - Reverse lookup
  - Triggers for changes

DNS vs Database

DNS very scalable in number of users, but not in updates

DNS doesn't provide searching sufficiently

DNS has only weak context sensitivity

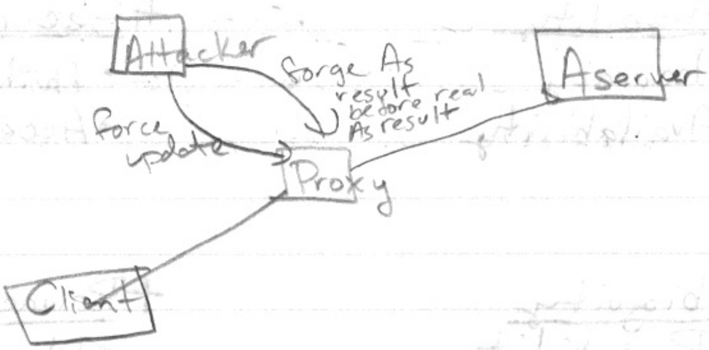
DNS doesn't support enough generality

DNS has name parcelling politics issues.

↳ searching w/ name means names must be aggregated together (all Coca-Cola names, etc.)

DNS has long insertion/update delay

DNS allows authenticated updates for authoritative server, but not for proxies.

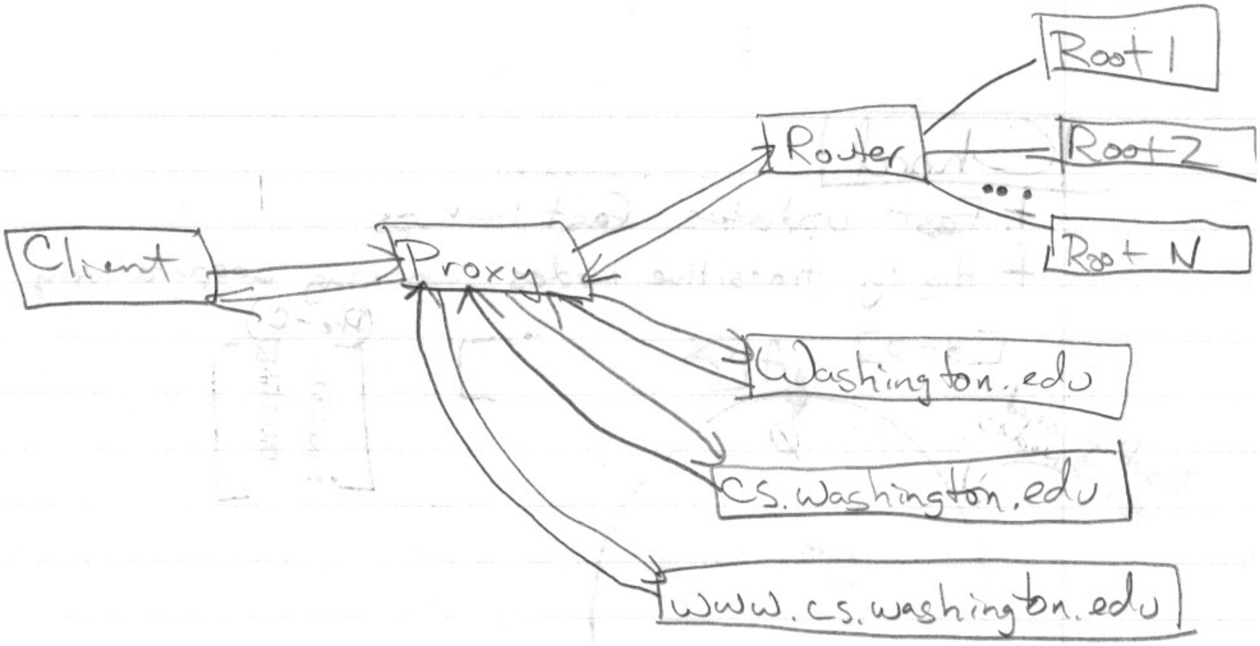


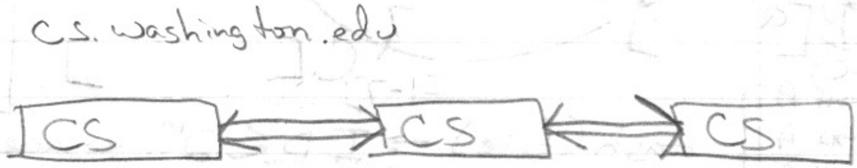
DNS does not have strict proxy coherency; instead, values time out after some long time, and are updated from the authoritative server.

↳ TTL ⇐⇒ Invalidation timeout

DNS can have multiple root servers, or have the roots hidden behind a single router, or partition names onto multiple servers. All are typically done.

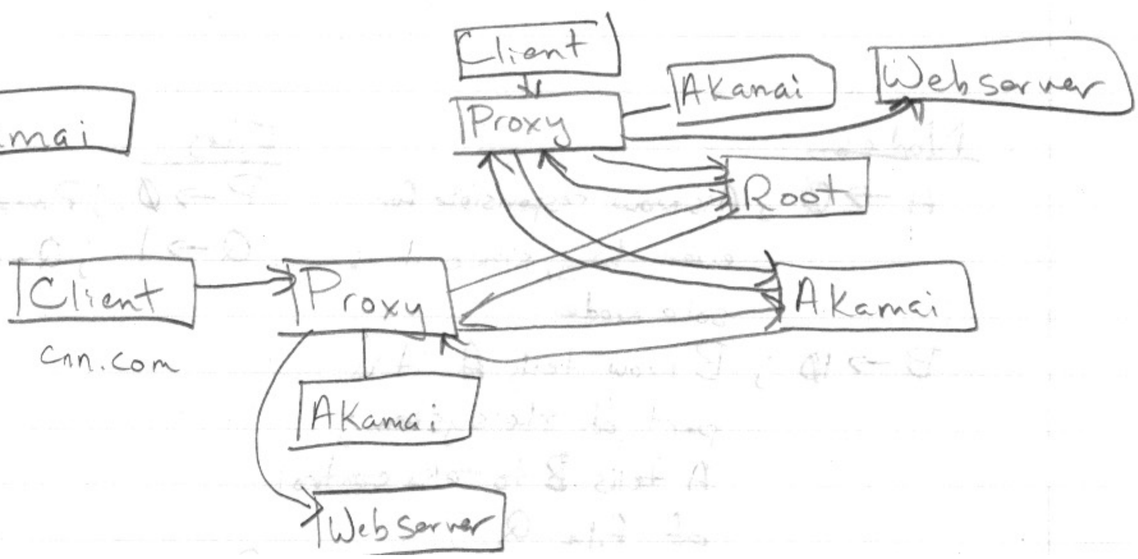
DNS allows easy updates, but they may take a while to propogate.





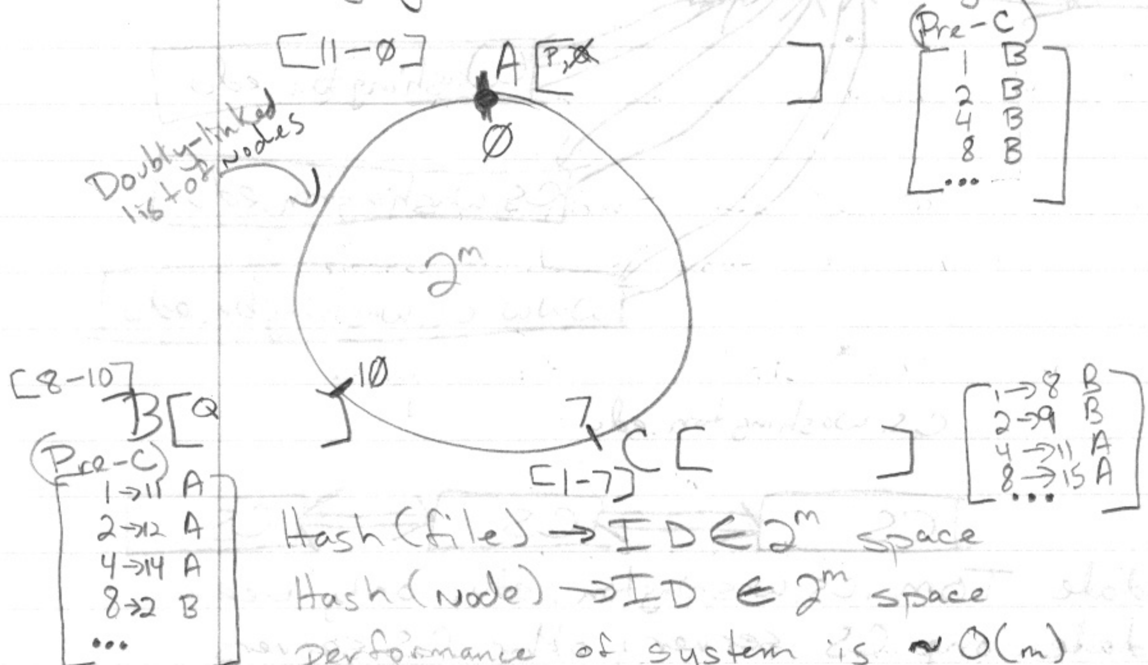
+ to update Tom.cs.washington.edu, only need to update one CS server; other CS servers will be updated

Akamai



Chord

+ Fast updates, fast lookups
+ Highly transitive nodes working cooperatively



Hash(file) → ID $\in 2^m$ space

Hash(node) → ID $\in 2^m$ space

performance of system is $\sim O(m)$

Nodes		Files	
A → Ø	; A is now responsible for everything, since it is sole node	P → Ø	; P mapped to A
B → 10	; B now tells A it is part of the system; A tells B to take control of file Q	Q → 1	; Q mapped to A
C → 7 ;		R → 4	; mapped to B until C jo

To add R starting at B, the 8→2 entry in its table is the closest to 4 possible, so since 8→2 maps to B, the implication is that B should own it. If R had mapped to 15, the 4→14 entry would have indicated A as owner.

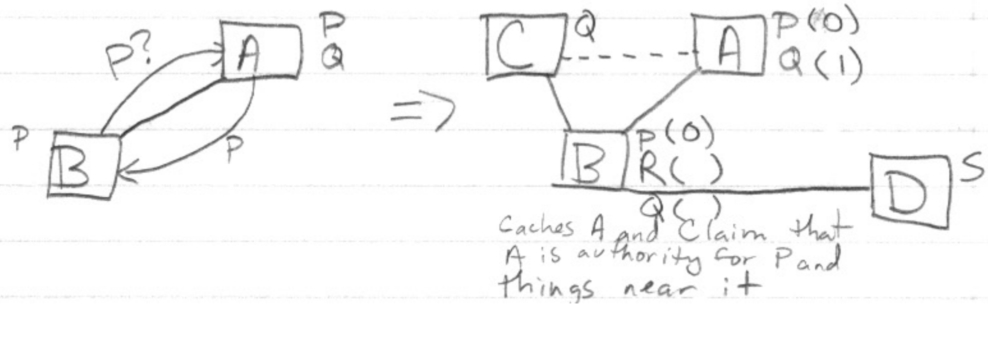
C joins to have $A \rightleftarrows C \rightleftarrows B \rightleftarrows A$; C only needs to steal entries from B; Mathematically, the randomness of the inefficiency of the distribution is bounded by $O(n)$ where n is # of nodes. C now builds his finger table & update any other node who might have him on their finger table (i.e. 7-1, 7-2, 7-4, 7-8, ...)

Plaxton trees extend this Chord setup to create rings that are topology-aware, such as doing a geographic lookup

If a node is willingly going away, it can transfer its load to the next node in the chain. However, for unwilling departures, redundancy must be introduced into the tables.

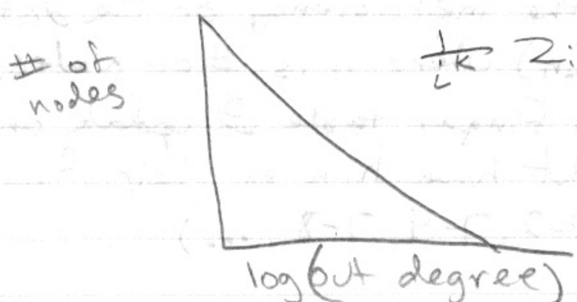
Freenet

Connect all nodes into random topology and search through that.



Searches in Freenet are DFS with a hop limit
Result is that searches for content may fail
simply because the content is too far away.

When a server responds for a search such as
P(Ø), it becomes "authoritative" for many near it.
Thus, "expert" nodes form in the center become
more authoritative and more interconnected.



Popular items become more popular.