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
  + Wildcard search
  + Delegation / NO
  + Fast update
  + Find nearest central authority
  + Scalability (clients, queries, names)
  + Updating

+ Reliability
  + Accuracy
  + Availability

+ Security
  + Authentication of update
  + Access control for lookup

+ Privacy
  + Plausible deniability

+ Ubiquity
  + Portability

+ Context sensitive naming

+ Simplicity
  + Flexibility
  + Keep low level simple
  + Routing name to multiple address
  + Routing multiple names to same address
    Names: 3 load balanced servers

+ 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 parceling 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.
  \( \text{TTL} \leftrightarrow \text{Invalidation timeout} \)
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 propagate.
Root 1
Root 2
Root N

Client
Proxy
Washington.edu
CS.washington.edu
www.cs.washington.edu

CS
CS
CS

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

Akamai

Client
Proxy
Akamai
Web server

Client
Proxy
Akamai
Web server
Chord
+ Fast updates, fast lookups
+ Highly transitive nodes working cooperatively

Hash (file) \rightarrow \text{ID} \in 2^m \text{ space}
Hash (node) \rightarrow \text{SID} \in 2^m \text{ space}
Performance of system is \( \Theta(n) \)

Nodes
\[
\text{A} \rightarrow \emptyset; \text{A is now responsible for everything, since it is sole node}
\]
\[
\text{B} \rightarrow \emptyset; \text{B now tells A it is part of the system}
\]
\[
\text{A} \text{ tells B to take control of file Q}
\]
\[
\text{C} \rightarrow \emptyset
\]

Files
\[
\text{P} \rightarrow \emptyset; \text{P mapped to A}
\]
\[
\text{Q} \rightarrow \emptyset; \text{Q mapped to A}
\]
\[
\text{R} \rightarrow \emptyset; \text{R mapped to B until C}
\]
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 1→14 entry would have indicated A as owner.

C joins to have A=C=B=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 to 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.

\[ \text{Freenet} \]

Connect all nodes into random topology and search through that.

\[ \text{Caches A and claim that B is authority for P and things near it} \]
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(∅), it becomes "authoritative" for many near it. Thus, "expert" nodes form in the center become more authoritative and more interconnected.

![Ziff curve](image)

Popular items become more popular.