

Freenet and Chord

Arvind Krishnamurthy
Fall 2003

Freenet

- Routing based lookup
 - Queries routed based on routing table
- Each node maintains a routing table with:
 - Key (to index the table)
 - Next hop node (where a file corresponding to the key might be available)
 - Pointer to local copy if one exists
- Searching for a file:
 - Find closest match and route
 - If failure, backtrack
 - "Steepest ascent hill climbing"
 - Ends up performing a DFS-like traversal
 - Routing is a heuristic

id	next_hop	file

Query side effects

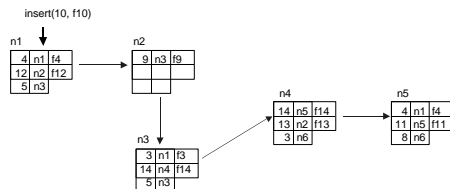
- Query determines a path to a copy of the file
- On the return path:
 - Each node caches a copy of the file
 - Each node remembers the source from which the file was obtained
- Local state on each node:
 - File cache: could be managed as LRU
 - Routing table cache: could also be managed as LRU

Lookup Analysis

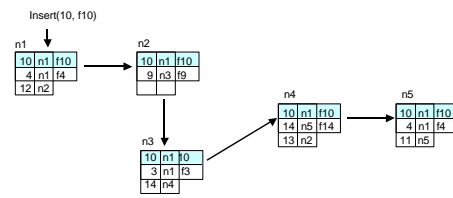
- Paper claims the following effects:
 - Nodes eventually specialize in locating sets of similar keys
 - If a node is listed in a routing table, it will get queries for related keys
 - Will gain more "experience" in answering those queries
 - Nodes become similarly specialized in storing files having related keys
 - Popular data will be transparently replicated and will exist closer to requestors
 - As nodes process requests, connectivity increases
 - Nodes can discover other nodes in the network
- Caveat: lexicographic closeness of filenames does not imply key-closeness

Insertion

- Insertion involves the following:
 - First attempt a "search" for the key
 - If you find a copy, "conflict" is detected and insert is abandoned
 - If no conflict, then insert the file by sending it along the query path



Insertion (contd.)



Analysis of insertion

- Newly inserted files are placed on nodes already possessing files with similar keys
 - Reinforces clustering
- New nodes can use inserts as a supplementary means of announcing their existence
- Attempts by attacker to supplant existing files with junk:
 - Initially insert performs a query
 - Query results in expanding the boundary of what is known
 - Eventually a conflict arises and insert cannot proceed
 - Surprising result: original file is more widely known!

Small World Property

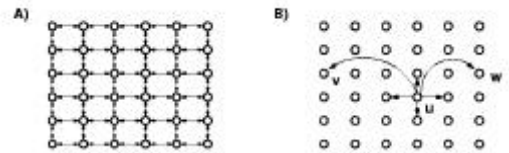
- Original experiments by Milgram:
 - Distance between two randomly selected persons is small
 - The path can be discovered in a distributed manner
- Experiment (1967):
 - 160 letters given to randomly chosen people in Omaha, Nebraska
 - Their target was a stockbroker in Boston
 - Can pass along the letter only through friends
 - 42 made it. Average length: 5.5 hops
- More generally, the setting is:
 - Clustered systems: most of the neighbors of an element are neighbors themselves
 - Still achieve a low diameter

Formal Analysis of Small World Property

- Diameter of uniformly random graphs is not too relevant
- Watts and Strogatz:
 - Rewired ring networks
 - Some short range connections
 - Some long range connections
 - Showed that it has low diameter
 - Number of systems have similar properties
 - Connections among neurons in certain species
 - Power grid in the western US
 - Hyperlink graph of the web
- Kleinberg's study:
 - Addresses the second question: can there be a distributed algorithm that discovers these low distance paths?

Kleinberg's Results

- Studied two-dimensional grids



- Assume that you have a budget for long links
 - Assume that the probability of a long link is some inverse-power of the number of lattice steps
- Distributed algorithm exists only when:
 - Probability is proportional to (lattice steps)⁻²

Using Small World Property in Freenet

- Question: is it possible to make a rigorous use of the small world property in Freenet?

Unstructured Networks

- Summary:
 - Connections between nodes are arbitrary
 - Files/keys are stored on arbitrary nodes
 - New routing table entries are created in a dynamic fashion
- Advantages of unstructured networks:
 - Algorithms tend to be simple
 - Can optimize for other properties: locality, quality of connections, etc.
- Disadvantage of unstructured networks:
 - Hard to make performance guarantees
 - Might result in query failures even though the object exists

Announcements

- Material for upcoming lectures:
 - Today: Chord
 - Wednesday: CAN
 - Friday: Pastry, Tapestry, Skip Graphs
- Assignment 2:
 - Design document due on Friday
 - Review meetings on Monday

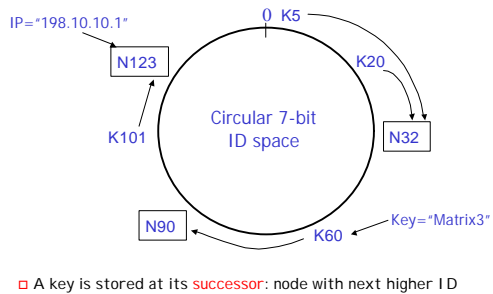
Structured Networks

- Commonly referred to as distributed hash tables
- Interesting systems: Chord, CAN, Tapestry, Pastry
 - Hypercube-like systems: Chord, Tapestry, Pastry
 - Multidimensional-mesh-like system: CAN
- Fundamental issues:
 - Try to keep the diameter of the network small
 - Try to minimize the neighborhood state of each node
 - Provide load-balance (in a probabilistic fashion)
 - Deal with dynamic node additions/deletions
 - Exploit locality of underlying network

Chord Overview

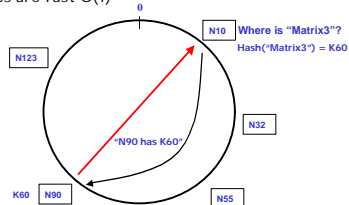
- Provides lookup service:
 - Lookup(key) \rightarrow IP address
 - Chord does not store the data
- m bit identifier space for both keys and nodes
- Key identifier = SHA-1(key)
 - Key="Matrix3" $\xrightarrow{\text{SHA-1}}$ ID=60
- Node identifier = SHA-1(IP address)
 - IP="198.10.10.1" $\xrightarrow{\text{SHA-1}}$ ID=123

Hashing Scheme



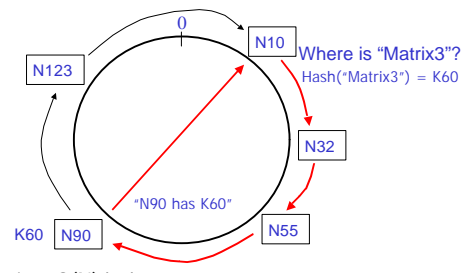
Simple Scheme

- Every node knows of every other node
 - that is, "N10" knows "N90" is "198.20.20.1"
 - requires global information
- Routing tables are large $O(N)$
- Lookups are fast $O(1)$



Other Extreme

- Every node knows its successor in the ring



- Intermediate solution: "finger tables"
- Every node knows m other nodes in the ring
 - That is, it knows the node that is maintaining $K + 2^i$
 - where K is mapped id of current node
- I increase distance exponentially

The diagram illustrates a ring network with four nodes: N80, N96, N112, and N16. Node N80 is at the bottom and is highlighted with a solid box. The other nodes are in dashed boxes. Arrows originate from N80 and point to the other nodes, representing the distances maintained in the finger tables. The distances are labeled as follows:

- From N80 to N80: $80 + 2^0$
- From N80 to N96: $80 + 2^1$
- From N80 to N112: $80 + 2^2$
- From N80 to N16: $80 + 2^4$

-

Faster Lookups

- Lookups take $O(\log N)$ hops
- Halve the distance to target with each hop

The diagram shows a circle representing a search space. Nodes are labeled N5, N10, N110, N20, N99, N32, N80, and N60. A path is shown from N32 to N110, with a label 'Lookup(K19)' next to N32. A green label 'K19' is also present.

-

Joining the ring

- Three step process:
 - Initialize all fingers of new node
 - Update fingers of existing nodes
 - Transfer keys from successor to new node
- Less aggressive mechanism (lazy finger update):
 - Initialize only the finger to successor node
 - Periodically verify immediate successor, predecessor
 - Periodically refresh finger table entries

- ## Joining the ring

Joining the ring: step 1

- Initialize the new node's finger table
 - Locate any node p in the ring
 - Ask node p to lookup fingers of new node N36
 - Return results to new node

1. Lookup(37,38,40,...,100,164)

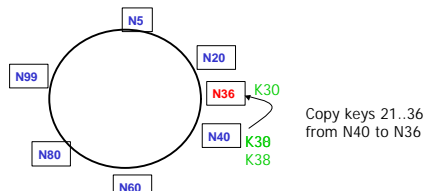
-
1. Lookup(37,38,40,...,100,164)

Joining the ring (contd.)

- Step 2: Transfer fingers of existing nodes
 - new node calls update function on existing nodes
 - existing nodes can recursively update fingers of other nodes
- Step 3: transfer keys from successor node to new node
 - only keys in the range are transferred

Copy keys 21...36
from N40 to N36

- ## Joining the ring (contd.)

[illegible]

-