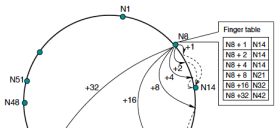


Applications of DHTs

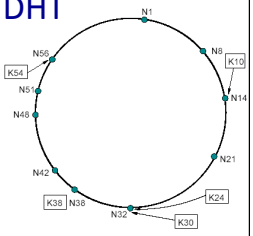
COS 461: Computer Networks

Recitation #6

http://www.cs.princeton.edu/courses/archive/spr13/cos461/

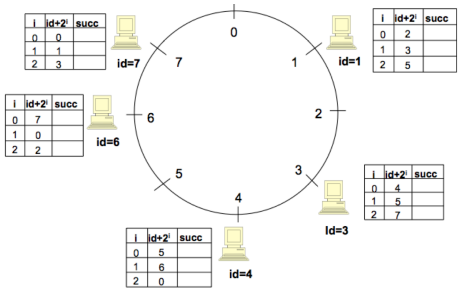
The Chord DHT

- ID space mod 2^{160}
  - *keyid = SHA1 (name)*
  - *nodeid = SHA1 (IP addr, i)*
    for i=1..v virtual IDs
    Each virtual ID maintains own routing table and keyspace



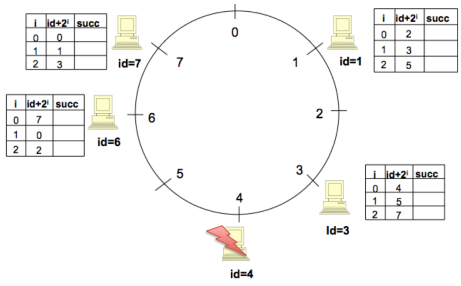
  - Now *vN* nodes, rather than N nodes. Each machine still owns $\sum_{vN}^{v} \frac{1}{vN} = \frac{1}{N}$ of total keyspace

- Key assigned to the next node found traversing the DHT ring in clockwise direction



Q1: Fill in the routing tables for node(s) above

Q2: What node(s) will receive query from node 1 for item with key 5? Assume iterative routing.



Suppose node 4 crashes, and system converges to a new state.

Q3: What routing entries changed?

Q4: What nodes will now receive query from node 7 for item 5?

5

Using DHTs for application-layer multicast routing

"SplitStream: High-Bandwidth Multicast in Cooperative Environments"

SOSP, 2003

6

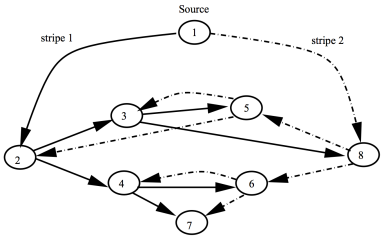
Problem with P2P multicast

- Goals for application
  - Capacity-aware bandwidth utilization
  - High tolerance of network churn
  - Load balance for all the end hosts
- P2P environment
  - Peers contribute resources
  - Different peers may have different limitations
- Challenge: tree-based multicast places high demand on few internal nodes

7

High-level design

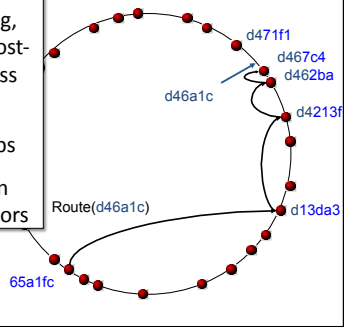
- Split data into stripes, each over its own tree
- Recover data from any m-out-of-n stripes
- Each node is internal to only one tree



8

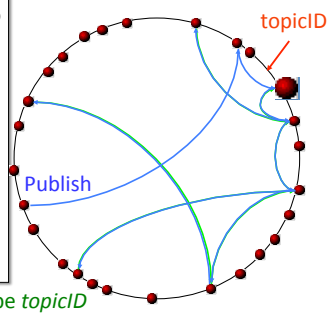
Background: DHT routing

- Built on Pastry DHT
- Visualize like Chord ring, but now correct for most-significant bit of address during each hop
- Log(n) state, log(n) hops
- Use proximity selection when choosing neighbors



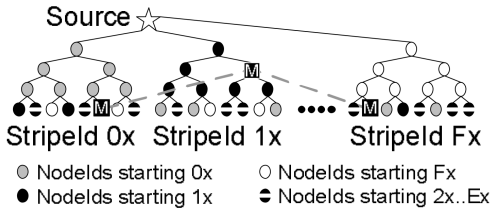
Scribe publish/subcribe system

- Topics map to nodes
- Subscribers register to receive topic updates
- Publishers push updates to topicID
- Message propagate back subscriber tree
- Insight: Convergence of lookup paths towards key's node



Design of SplitStream

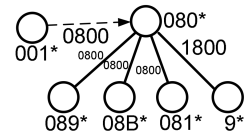
- Choosing groupIds for the Scribe multicast trees that differ in the most significant digit ensures the property that each node is internal to only one tree and a leaf in the rest



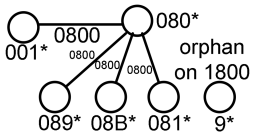
Design of SplitStream

- Divides data into stripes, each using one Scribe multicast tree
- Choosing groupIds for the trees that all differ in the most significant digit ensures property that each node is internal to only one tree
  - Inbound bandwidth: can achieve desired indegree while this property holds
  - Outbound bandwidth: Harder: we'll have to look at the node join algorithm to see how this works

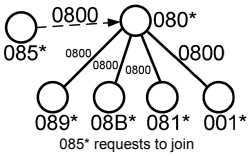
New Children Adoption



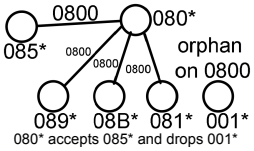
001* requests to join stripe 0800, but 080* has reached its outdegree limit of 4



080* accepts 001* and drops 9* because *it did not share the first digit with 080**



085* requests to join



080* accepts 085* and drops 001* because *it has a shorter prefix match with stripe 0800 than the other children*

13

Spare Capacity Group

- Orphaned nodes recursively try to reattach to their former siblings, but if that does not work..