EE 122: A Note On Joining Operation in Chord

Ion Stoica
October 20, 2002

Big Picture

- Assume circular identifier space is $0..2^m$
- To each node we associate a unique id in this identifier space
- Each node is responsible for all identifiers between itself and its predecessor on this circle

Identifier to Node Mapping Example

- Node 8 maps [5,8]
- Node 15 maps [9,15]
- Node 20 maps [16, 20]
- ...
- Node 4 maps [59, 4]

Routing

- Each node maintains its successor
- Route packet (ID, data) to the node responsible for ID using successor pointers
Joining Operation

- Each node $A$ periodically sends a \texttt{stabilize()} message to its successor $B$.
- Upon receiving a \texttt{stabilize()} message a node $B$ returns its predecessor $B'$ to $A$ by sending a \texttt{notify()} message.
- Upon receiving notify() from $B$, $A$ updates its successor to $B'$ if $B'$ is between $A$ and $B$; otherwise $A$ doesn’t do anything.

Joining Operation (cont’d)

- Node 44 sends a stabilize message to its successor, node 58.
- Node 58 reply with a notify message.
- Node 44 updates its successor to 50.

Joining Operation

- Node with id=50 joins the ring.
- Node 50 needs to know at least one node already in the system.
  - Assume known node is 15.

Joining Operation

- Node 50 asks node 15 to forward join message.
- When join(50) reaches the destination (i.e., node 58), node 58 (1) updates its predecessor to 50, and (2) returns a notify message to node 50.
- Node 50 updates its successor to 58.
Joining Operation (cont’d)

- Node 44 sends a stabilize message to its new successor, node 50
- Node 50 sets its predecessor to node 44

This completes the joining operation!

Achieving Robustness

- To improve robustness each node can maintain the k (> 1) immediate successors instead of only one successor
- In the notify() message, node A can send its k-1 successors to its predecessor B
- Upon receiving notify() message, B can update its successor list by concatenating the successor list received from A with A itself