Day 09: Tapestry
Continued
Node IDs: 011, 211, 322, 234

```
<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>211</td>
<td>300</td>
<td></td>
<td></td>
</tr>
<tr>
<td>211</td>
<td>222</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>211</td>
<td>211</td>
<td>211</td>
<td>211</td>
</tr>
</tbody>
</table>
```

Lookup (keyID, step):
row = step
col = keyID at (step)
nextNode = Table [col, row]
step++
nextNode Lookup (keyID, step)

```
<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>011</td>
<td>231</td>
<td></td>
<td></td>
</tr>
<tr>
<td>300</td>
<td>200</td>
<td></td>
<td></td>
</tr>
<tr>
<td>300</td>
<td>333</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

#Cols = base

\[ N = \log_{\text{base}} \# \text{rows} \]

Keyspace
Tapestry Network

all nodes participating in the protocol
Surrogate Routing

When the ID isn't a server ID, you will end up...

Diagram showing a circle with arrows labeled $S_x$ and $S_y$ pointing towards a grid.
New Node Joining a network

What are the goals for this node?
New Node Joining a network

What are the goals for this node?

1. Build Route Table
2. Get Objects
3. Build Back-pointers
4. fill the hole in the routing table of other nodes (need to know nodes)
5. Optimize table
New Node Joining a network

What are the goals for this node?

Tapestry Node

route table  Back pointers  object store

Tapestry Network

1. Build Route Table
2. Get Objects
3. Build Back-pointers
4. Fill the hole in the routing table of other nodes (need to know nodes)
5. Optimize table
New Node Joining a network

How to build route table?

Root Node

Tapestry Network

Clone the root node's routing table

1. Easy to code
2. Quick (performance)
3. Not optimized (route table)
4. $O(\log_{2} N)$

Find all the nodes in the tapestry network to build routing table

1. Best table ever!! (arggg)
2. Time/cpu overhead
Why does copy proof work?
New Node Joining a network

How to build route table?

Tapestry Network

Root Node

Clone Root Node

Suboptimal table which will node include "closest" neighbors

find all nodes & build new table
takes lots of time & since network is always changing may never end
Reliable Multicast

Goals: build optimal table by contacting a "few nodes" + inform "need-to-know" nodes of the new node

\[ p = \text{shared prefix}(\text{new node}, \text{root}) \]

Back pointer traversal

Note: backpointer
Check notes here to see if new node is closer than node x
leaving the network

choice (graceful exit)

1. let back pointers know of your exit & give candidate replacement
2. give objects to their new root servers

unexpectedly (ungraceful exit)

node leaving has no does nothing

Nodes in the network must
1. detect the missing node & change table
2. wait for client to republish the objects
Today

Tapestry Continued

1. “need to know” nodes
2. Overlays
3. Surrogate Routing
4. New node joining network
5. New node leaving network