DHT Design Goals

- An “overlay” network with:
  - flexible mapping of keys to physical nodes
  - small network diameter
  - small degree
  - local routing decisions

- A “storage” or “memory” mechanism with
  - best-effort persistence (soft state)

- We’ll look at two designs:
  - Chord
  - Pastry

Pastry

- Pastry nodeIDs and search keys are generated by a hash function that produces a value treated as:
  - A sequence of digits with base $2^b$ (typically, $b=4$, i.e., hexadecimal), and
  - Modulo $2^{128}$

- Given a message, $m$, and a key, $k$, Pastry routes the message to the live node with nodeID numerically closest to $k$. 
Hash IP addr to Node ID (b=4, modulo $2^{24}$)

- 152.2.137.47 (USA) → Hash → F33D88
- 203.199.213.5 (India) → Hash → 3B2C0C
- 166.111.4.37 (China) → Hash → 9B5988
- 169.229.60.105 (USA) → Hash → 7C6938
- 64.236.24.12 (USA) → Hash → E42E10
- 207.46.150.20 (USA) → Hash → C31950
- 198.175.96.33 (USA) → Hash → 74DB3C
- 144.82.100.65 (UK) → Hash → 04BCA8
- 138.96.146.2 (France) → Hash → D24808
- 129.126.11.23 (Germany) → Hash → 0ED1F2
- 193.232.113.43 (Russia) → Hash → 7438FC

Nodes in Pastry Example (b=4, modulo $2^{24}$)

- N-F33D88
- N-E42E10
- N-D24808
- N-C31950
- N-9B5988
- N-74DB3C
- N-7438FC
- N-0ED1F2
- N-04BCA8
- N-7C6938
- N-9B5988
- N-74DB3C
- N-7438FC
- N-0ED1F2
- N-04BCA8
- N-7C6938
- N-9B5988
- N-74DB3C
- N-7438FC
- N-0ED1F2
- N-04BCA8
- N-7C6938
- N-9B5988
- N-74DB3C
- N-7438FC
- N-0ED1F2
Key Locations in Pastry Example

Nodes should store Key/Value pairs for the numerically closest Key values.

For node with ID \( N \), let \( N_l \) be the next larger nodeID, and \( N_s \) the next smaller nodeID.

Node \( N \) stores keys in the range: \( \{N - (|N - N_s|/2), N + (|N - N_l|/2)\} \)

Pastry Routing State (“Prefix Routing”)

Routing Table:

\[ \log_2 b \] Rows, \( 2^{b-1} \) entries (with one null) per row

The \( 2^{b-1} \) entries at row \( n \) refer to remote nodes that:

- share the local node’s nodeID in the first \( n \) digits, but
- whose \( n+1 \) digit has one of the \( 2^{b-1} \) possible values other than the \( n+1 \) digit in the local nodeID.

If no such node is known, the entry is empty.

Leaf Set:

The leaf set, \( L \), is the set of nodes with the

- \( |L|/2 \) numerically closest larger nodeIDs,
- \( |L|/2 \) numerically closest smaller nodeIDs.

The typical value of \( |L| \) is \( 2^b \).
Pastry Routing/Lookup ("Prefix Routing")

- When a message for key $K$ arrives at a node:
  - If $K$ is in the range covered by the Leaf Set, it is forwarded to the entry whose nodeID is numerically closest.
  - Otherwise, the Routing Table is used to forward it to a nodeID that shares a common prefix with $K$ by at least one more digit than does this node's nodeID.
  - Otherwise, forward to a nodeID that shares a prefix with $K$ at least as long as this node's nodeID but is numerically closer to $K$ than this node's nodeID.

Pastry Routing Examples

**Arriving Key, $K$**

<table>
<thead>
<tr>
<th>Row</th>
<th>Arriving Key</th>
<th>NodeID</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>21032113</td>
<td>22301203</td>
</tr>
<tr>
<td>1</td>
<td>10311230</td>
<td>10323302</td>
</tr>
<tr>
<td>2</td>
<td>10233003</td>
<td>10233001</td>
</tr>
<tr>
<td>3</td>
<td>10233321</td>
<td>10233232</td>
</tr>
<tr>
<td>4</td>
<td>10231112</td>
<td>10222302</td>
</tr>
<tr>
<td>5</td>
<td>1023103</td>
<td>10233102</td>
</tr>
</tbody>
</table>

**NodeID 10233102**

<table>
<thead>
<tr>
<th>Row</th>
<th>Routing Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-0-2-3-102</td>
</tr>
<tr>
<td>1</td>
<td>1-1-3-102</td>
</tr>
<tr>
<td>2</td>
<td>10-1-3-102</td>
</tr>
<tr>
<td>3</td>
<td>102-2-2-3-2</td>
</tr>
<tr>
<td>4</td>
<td>1023-3-3-2</td>
</tr>
<tr>
<td>5</td>
<td>10233-3-3-2</td>
</tr>
<tr>
<td>6</td>
<td>102333-3-3-2</td>
</tr>
<tr>
<td>7</td>
<td>1023333-3-3</td>
</tr>
</tbody>
</table>

Example with $b=2$, $|L|=8$ (IP addresses not shown)
Pastry Routing Example

"some name" → Hash → Hash(6F2175) → RouteToClosest(6F2175) → N-D24808

Mean lookup is \( O(\log_2 b^n) \)

⇒ Scalable!

Pastry Node \( X \) Joins

- Node with nodeID = \( X \) knows about existing node with nodeID \( A \)
  - \( X \) contacts \( A \) and sends join message with key=\( X \)
  - Pastry nodes route message to some nodeID = \( Z \) that is numerically closest to \( X \)
  - All nodes contacted in routing the join message return their routing state to \( X \)

- Reallocating Key/Value mappings to nodes is left to the application (it is notified of changes)
Pastry Join Protocol

**Mean Join messages are \(~3*2^b*\log_2^b\ n\)**

=> Scalable!

Pastry Node X Joins (cont.)

- Z has the *nodeID* numerically closest to X so its Leaf Set becomes the base for X’s Leaf Set

- For Routing Table (RT) rows at X:
  - \(RT_X[0] = RT_A[0]\)
    - row zero independent of *nodeID* prefix in all nodes
  - \(RT_X[i] = RT[i]\)
    - The \(i^{th}\) row of X’s routing table can be taken from the \(i^{th}\) row of the table in the \(i^{th}\) node encountered while routing to Z.
    - This works because X’s *nodeID* shares a prefix at least as long as each successive *nodeID* along the path to Z.

- X transmits a copy of its new routing state to each *nodeID* found in its state.
**Node Failure/Leave**

- Failure of a node in some node’s Leaf Set (detected with periodic pings):
  - Contact node in Leaf Set with largest *nodeID* on the side (larger vs smaller) with the failed node and get its Leaf Set. It will contain an appropriate replacement.

- Failure of a node in some node’s Routing Table (detected on attempt to contact during routing):
  - Contact nodes in same (or higher, if necessary) row(s) as the failing entry and ask for one of their entries with the appropriate prefix.

---

**DHTs discussion**

- What systems can you build using DHTs?
- Is node diversity useful?
- How to reduce stretch?
  - Can it be done in Pastry?
- How to support range requests or partial matches between request and key?
- What real applications use DHTs today?
  - Why or why not?