**DIIT 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^n (typically, n=4, i.e., hexadecimal), and
  - Modulo 2^n
- Given a message, m, and a key, k, Pastry routes the message to the fixe node with nodeID numerically closest to k.
Key Locations in Pastry Example

Nodes should store Key/Value pairs for the numerically closest key values.
For node with ID \( N \), let \( N_f \) be the most frequent node ID, and \( N_s \) the next smallest node ID.

Node \( N \) stores keys in the range:
\[ \{ N - N_f(1), N - (N_f/N) \} \]

Pastry Routing Table

("Prefix Routing")

Routing Table:

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

The \( 2^n \) entries in row \( n \) refer to remote nodes that:
- share the local node's node ID in the first \( n \) digits, but
- where \( n+1 \) digit has one of the \( 2^n-1 \) possible values other than the \( n+1 \) digit in the local node ID.

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

Leaf Set:
The leaf set, \( L \), is the set of nodes with the:
- \( L \) is numerically closest \( \text{larger node ID} \),
- \( L \) is numerically closest \( \text{smaller node ID} \),
The typical value of \( L \) is \( 2^n \).

Example with \( b=2 \), \( d=8 \), \( l=8 \) (IP addresses not shown)
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

<table>
<thead>
<tr>
<th>Arriving Key, $K$</th>
<th>Leaf Set</th>
<th>NodeID 10233102</th>
</tr>
</thead>
<tbody>
<tr>
<td>21022113</td>
<td>22011260</td>
<td></td>
</tr>
<tr>
<td>10011230</td>
<td>10023202</td>
<td></td>
</tr>
<tr>
<td>10023202</td>
<td>10023801</td>
<td></td>
</tr>
<tr>
<td>10223321</td>
<td>10223322</td>
<td></td>
</tr>
<tr>
<td>10023812</td>
<td>10222312</td>
<td></td>
</tr>
<tr>
<td>10213102</td>
<td>10233102</td>
<td></td>
</tr>
</tbody>
</table>

Example with $d=2$, $d=4$ (IP addresses not shown)
Pastry Routing Example

Mean lookup is $O(\log_2 n)$
- Scalable!

Pastry Node X Joins

- Node with nodeID $= X$ knows about existing node with nodeID $= A$
  - Y 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 connected in routing the join message return their routing state to $X$

- Relocating Key/Value mappings to nodes is left to the application (it is notified of changes).
- 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 Y:
  - $RT_{i,j} = RT_{j,i}$
  - row zero independent of nodeID prefix in all nodes
  - $RT_{i,j} = RT_{j,i}$
  - The $P$ row of X's routing table can be taken from the $P$ row of the table in the $P$ 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 node in same (or higher, if necessary) row(s) as the failing entry and ask for one of their entries with the appropriate prefix.

DIIT's discussion

- What systems can you build using DIITs?
- Is node diversity useful?
- How to reduce stretch?
- How to support range requests or partial matches between request and key?
- What real applications use DIITs today?
  - Why or why not?
- Pros and cons of an unstructured p2p system?