<table>
<thead>
<tr>
<th><strong>CS 6/75995</strong></th>
<th><strong>Kent State University</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Foundation of Peer-to-Peer</strong></td>
<td><strong>Dept. of Computer Science</strong></td>
</tr>
<tr>
<td><strong>Applications &amp; Systems</strong></td>
<td><strong><a href="http://www.cs.kent.edu/~javed/class-P2P08">www.cs.kent.edu/~javed/class-P2P08</a></strong></td>
</tr>
<tr>
<td></td>
<td><strong>PASTRY</strong></td>
</tr>
</tbody>
</table>
[mechanics]

- Update overview
- 1 class start+routing+node failure

Pastry [update.. Old]

- Overview
  - Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems, Antony Rowstron and Peter Druschel, 2001
- Topology
  - Consistent Hashing
  - Key Space
- Routing
  - Leaf Set
  - Numerically Closest Set
  - Physically Closest Set
- Node Arrival
  - Bootstrapping
  - Finding a Zone
  - Joining the Routing (Route Table Updates)
- Node Departure
  - Identification of Takeover Node
  - Recovery Algorithm
- Performance Analysis
- Evaluation
  - Stability
  - Robustness
  - Load balancing
Pastry

Topology

Pastry

• An overlay network that provides a self-organizing routing and location service (like Chord).

• Seeks to minimize the “distance” (scalar proximity metric like routing hops) messages travel.

• Expected number of routing steps is $O(\log N)$; $N=$No. of Pastry nodes in the network.
Pastry Topology

- Nodes are organized in a circular ID space, using consistent DHT hashing.
- $NodeId$ randomly assigned from $\{0, ..., 2^{128}.1\}$
- A pastry node can route to the numerically closest node to a given key in less than $\log_{2^b} N$ steps. ($b, |L|$ are configuration parameters)
- Despite concurrent node failures, delivery is guaranteed unless more than $|L|/2$ nodes with adjacent NodeIds fail simultaneously
- Each node join triggers $O(\log_{2^b} N)$ messages

Pastry: Object distribution

- Consistent hashing
  - 128 bit circular id space
  - $nodeIds$ (uniform random)
  - $objIds/keys$ (uniform random)
- Invariant: node with numerically closest nodeld maintains object
Pastry: Object insertion/lookup

- **Problem**: complete routing table not feasible

Pastry Routing
Node ID

- NodeIds are in base $2^b$

$$n$$

<table>
<thead>
<tr>
<th>0001</th>
<th>0000</th>
<th>0010</th>
<th>0011</th>
<th>0011</th>
<th>0001</th>
<th>0000</th>
<th>0010</th>
</tr>
</thead>
</table>

NodId#10233102

Three Concept of Proximity

- Set of nodes with $|L|/2$ smaller and $|L|/2$ larger numerically closest NodeIds
- Prefix-based routing entries
- $|M|$ "physically" closest nodes
**Routing Table Dimensions**

<table>
<thead>
<tr>
<th></th>
<th>Nodeld 10233102</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Leaf set</strong></td>
<td></td>
</tr>
<tr>
<td>SMALLER</td>
<td></td>
</tr>
<tr>
<td>10233093</td>
<td>10233921</td>
</tr>
<tr>
<td>10233001</td>
<td>10233000</td>
</tr>
<tr>
<td>LARGER</td>
<td></td>
</tr>
<tr>
<td>10233120</td>
<td>10233122</td>
</tr>
<tr>
<td>10233200</td>
<td>10233232</td>
</tr>
</tbody>
</table>

L nodes in leaf set (typical L = $2^b$)

- $2^b$ columns
- $2^b$ rows

Routing table

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1-1-301233</td>
<td>1-2-230203</td>
</tr>
<tr>
<td>10-0-31203</td>
<td>10-1-32102</td>
</tr>
<tr>
<td>102-0-0230</td>
<td>102-1-30203</td>
</tr>
<tr>
<td>1023-0-322</td>
<td>1023-1-2102</td>
</tr>
<tr>
<td>10233-0-01</td>
<td>10233-2-32</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

M neighbors (typical M = $2 \times 2^b$)

- How to select b?
  - NodeIds are in base $2^b$
  - One row for each prefix of local NodeId ($\log_2 N$ populated on average)
  - One for each possible digit in the NodeId representation $2^b - 1$ columns

b defines the tradeoff:

$(\log_2 N) \times (2^b - 1)$ entries Vs. $\log_2 N$ routing hops
A Hypothetical Pastry node with ID 10233102

Values: b = 2, and l = 8. All numbers are in base 4.

The top row of the routing table is row zero.

The entries are common prefix with 10233102 - next digit - rest of nodeId.
**Pastry: Leaf Sets**

- In leaf set each node maintains IP addresses of the nodes with the IL|/2 numerically closest larger IL|/2 |L|/2 smaller numerically closest nodeIds.
- Routing efficiency/robustness
- Fault detection (keep-alive)
- Application-specific local coordination

---

**Neighborhood Set**

- The neighborhood set M contains nodeIDs and IP addresses of |M| nodes those are physically closest (or as per some other proximity metric) to the local node.
- Its use will be discussed in “proximity routing” discussion.
Find (d46a1c)

- Route Table of A 65a1fc
- Route Table of B d13da3
- Route Table of C d4213f
- 65a1fc finds B (d13da3)
- d13da3 finds C (d4213f)
- d4213f finds D (d462ba)

Pastry: Routing

Properties
- $\log_{16} N$ steps
- $O(\log N)$ state
Pastry Routing Algorithm

1. if \((L_{i-1}[L_i/2] \leq D \leq L_{i+1}[L_i/2])\) {
   2. // \(D\) is within range of our leaf set
   3. forward to \(L_i\), s.t., \(|D - L_i|\) is minimal:
   4. } else {
      5. // use the routing table
      6. Let \(l = \text{shl}(D, A)\);
      7. if \((R_i^d \neq \text{null})\) {
         8. forward to \(R_i^d\);
      9. }
      10. else {
          11. // rare case
          12. forward to \(T \in L \cup R \cup M\), s.t.
             \(\text{shl}(T, D) \geq l\),
             \(|T - D| < |A - D|\)
          13. }
      14. }
      15. }

Pastry: Routing Procedure

if (destination is within range of our leaf set)
forward to numerically closest member
else
let \(l = \) length of shared prefix
let \(d = \) value of \(l\)-th digit in \(D\)’s address
if \((R_i^d \text{ exists})\)
   forward to \(R_i^d\)
else
   forward to a known node that
   (a) shares at least as long a prefix
   (b) is numerically closer than this node
Routing Performance: Intuition

- (1) – Single hop, termination
- (2) – No. of nodes which prefix-match the key upto current length reduces by $2^b$
- (3) – Low probability, adds one hop

Pastry
Self-Organization
**Pastry: Node Addition**

- New node: d46a1c
- Route(d46a1c)

**Self-organization: Node Arrival**

- Arriving Node X knows “nearby” node A.
- X asks A to route a “join” message with key = NodeId(X).
- Message is routed and finds Z, whose NodeId is numerically closest to NodeId(X)
- All nodes along the path A, B, ..., Z send state tables to X
- X initializes its state using this information.
- X sends its state to “concerned” nodes
State Initialization (1)

- X borrows A’s Neighborhood Set
  - A is geographically closer to X so it is OK to borrow the set.

State Initialization (2)

- Z’ ID is numerically closest to X’s Therefore:
- X’s leaf set is derived from Z’s leaf set
State Initialization (3)

- $X_0$ set to $A_0$
- $X_1$ set to $B_1$, $X_2$ set to $C_2$, ...
- Finally, $X$ transmits its leafset, neighborhood set and routing table to each of the nodes in these sets.

- The total message cost is $O(\log_b N)$. The constant is $3 \times 2^b$.
- To handle concurrent arrival, extensive timestamps are used.

Self-organization: Node Failure (1)

- Detected when a live node tries to contact a failed node

- Updating Leaf set – get leaf set from largest index on the side of the failed node.

- This set partially overlaps the present nodes leaf set $L$ and extra nodes not in $L$.

- It thus selects the appropriate one. Verifies that it is alive and adds.
Self-organization: Node Failure (2)

- Updating routing table - To repair \( R^d_1 \), ask any \( R^i_1 \) \( i \neq d \) in the same row for its \( R^d_1 \)

- If the unlikely case it’s empty (no live node), with the right prefix then it contacts any \( R^i_{i+1} \) \( i \neq d \). thereby casting a wider net.

- This process is highly unlikely to fail.

Self-organization: Node Failure (3)

- Updating neighborhood set
- This is not used in routing generally.
- – Ask any alive set-members for their neighbors
Locality

- Application provides the “distance” function
- Invariant: “All routing table entries refer to a node that is near the present node, according to the proximity metric, among all live nodes with an appropriate prefix”
- Invariant maintained on self-organization

Handling Malicious Nodes

- Routing is deterministic
- Randomize choice between multiple suitable candidates – with a bias towards the best one
Routing Performance

- The expected number of routing steps is $\log_2 N$ steps, assuming accurate routing tables and no recent node failures. Consider the three cases in the routing procedure.

- If a message is forwarded using the routing table (lines 6–8), then the set of nodes whose ids have a longer prefix match with the key is reduced by a factor of $2^b$ in each step, which means the destination is reached in $\log_2 N$ steps.

- If the key is within range of the leaf set (lines 2–3), then the destination node is at most one hop away.

- The third case arises when the key is not covered by the leaf set (i.e., it is still more than one hop away from the destination), but there is no routing table entry. Assuming accurate routing tables and no recent node failures, this means that a node with the appropriate prefix does not exist (lines 11–14). The likelihood of this case, given the uniform distribution of nodeIds, depends on $|L|$.

- Analysis shows that with $|L| = 2^b$ and $|L| = 2x2^b$, the probability that this case arises during a given message transmission is less than .02 and 0.006, respectively. When it happens, no more than one additional routing step results with high probability.

- In the event of many simultaneous node failures, the number of routing steps required may be at worst linear in $N$, while the nodes are updating their state. This is a loose upper bound; in practice, routing performance degrades gradually with the number of recent node failures (shown experimentally). Eventual message delivery is guaranteed unless $|L|$ nodes with consecutive nodeIds fail simultaneously. The probability of such a failure can be made very low.
Pastry
Extensions: API & Applications

The Pastry API

• Operations exported by Pastry
  – nodeId =
    pastryInit(Credentials,Application)
  – route(msg,key)

• Operations exported by the application working above Pastry
  – deliver(msg,key)
  – forward(msg,key,nextId)
  – newLeafs(leafSet)