Peer-to-Peer Networks
06 Tapestry

Christian Schindelhauer
Technical Faculty
Computer-Networks and Telematics
University of Freiburg
Objects and Peers are identified by
- Objekt-IDs (Globally Unique Identifiers GUIDs) and
- Peer-IDs

IDs
- are computed by hash functions
  • like CAN or Chord
- are strings on basis B
  • B=16 (hexadecimal system)
Every peer A maintains for each prefix x of the Peer-ID:
- if a link to another peer sharing this Prefix x
- i.e., peer with ID B=xy has a neighbor A, if xy′=A for some y, y′

Links sorted according levels:
- the level denotes the length of the common prefix
- Level L = |x|+1
For each prefix $x$ and all letters $j$ of the peer with ID A
- establish a link to a node with prefix $x_j$ within the neighborhood set $N_{x,j}^A$.

Peer with Node-ID A has $b \cdot |A|$ neighborhood sets.

The neighborhood set of contains all nodes with prefix $s_j$
- Nodes of this set are denoted by $(x,j)$.
Example of Neighborhood Sets

Neighborhood set of node 4221

<table>
<thead>
<tr>
<th>j=0</th>
<th>j=1</th>
<th>j=2</th>
<th>j=3</th>
<th>j=4</th>
<th>j=5</th>
<th>j=6</th>
<th>j=7</th>
</tr>
</thead>
<tbody>
<tr>
<td>4220</td>
<td>4221</td>
<td>4222</td>
<td>4223</td>
<td>4224</td>
<td>4225</td>
<td>4226</td>
<td>4227</td>
</tr>
<tr>
<td></td>
<td>40??</td>
<td>41??</td>
<td>42??</td>
<td>43??</td>
<td>44??</td>
<td>45??</td>
<td>46??</td>
</tr>
<tr>
<td></td>
<td>0??</td>
<td>1??</td>
<td>2??</td>
<td>3??</td>
<td>4??</td>
<td>5??</td>
<td>6??</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Levels:
- Level 1
- Level 2
- Level 3
- Level 4
Links

- For each neighborhood set at most $k$ Links are maintained

$$k \geq 1: \left| N^A_{x,j} \right| \leq k$$

- Note:
  - some neighborhood sets are empty
Consistency

- If $N^A_{x,j} = \emptyset$ für any $A$
  - then there are no $(x,j)$ peers in the network
  - this is called a hole in the routing table of level $|x|+1$ with letter $j$

Network is always connected

- Routing can be done by following the letters of the ID $b_1b_2\ldots b_n$
  
  \[
  \begin{align*}
  N^A_{\phi,b_1} & \quad \text{1st hop to node } A_1 \\
  N^A_{b_1,b_2} & \quad \text{2nd hop to node } A_2 \\
  N^A_{b_1b_2,b_3} & \quad \text{3rd hop to node } A_3 \\
  \ldots 
  \end{align*}
  \]
Locality

- **Metric**
  - e.g. given by the latency between nodes

- **Primary node of a neighborhood set** $N^A_{x,j}$
  - The closest node (according to the metric) in the neighborhood set of A is called the primary node

- **Secondary node**
  - the second closest node in the neighborhood set

- **Routing table**
  - has primary and secondary node of the neighborhood table
Object with ID Y should be stored by a so-called Root Node with this ID.

If this ID does not exist then a deterministic choice computes the next best choice sharing the greatest common prefix.
Surrogate Routing

- compute a surrogate (replacement root node)
- If (x,j) is a hole, then choose (x,j+1),(x,j+2),… until a node is found
- Continue search in the next higher level
Example: Surrogate Routing

- Lookup of 4666 by peer 2716
  - Level 1, j=4
  - Level 2, j=6 does not exist, next link j=8
  - Level 3, j=6
  - Peer 4860 has no level 4 neighbors => end of search
Publishing Objects

- Peers offering an object (storage servers) - send message to the root node
- All nodes along the search path store object pointers to the storage server
Choose the root node of \( Y \)
- Send a message to this node using primary nodes
- Abort search if an object link has been found
  - then send message to the storage server
Fault Tolerance

- Copies of object IDs
  - use different hash functions for multiple root nodes for objects
  - failed searches can be repeated with different root nodes

- Soft State Pointer
  - links of objects are erased after a designated time
  - storage servers have to republish
    - prevents dead links
    - new peers receive fresh information
Surrogate Routing

- **Theorem**
  - Routing in Tapestry needs $O(\log n)$ hops with high probability
Adding Peers

- Perform lookup in the network for the own ID
  - every message is acknowledged
  - send message to all neighbors with fitting prefix,
    - Acknowledged Multicast Algorithm
- Copy neighborhood tables of surrogate peer
- Contact peers with holes in the routing tables
  - so they can add the entry
  - for this perform multicast algorithm for finding such peers
Leaving of Peers

- Peer A notices that peer B has left
- Erase B from routing table
  - Problem holes in the network can occur
- Solution: Acknowledged Multicast Algorithm
- Republish all object with next hop to root peer B
Pastry versus Tapestry

- Both use the same routing principle
  - Plaxton, Rajamaran und Richa
  - Generalization of routing on the hyper-cube

- Tapestry
  - is not completely self-organizing
  - takes care of the consistency of routing table
  - is analytically understood and has provable performance

- Pastry
  - Heuristic methods to take care of leaving peers
  - More practical (less messages)
  - Leaf-sets provide also robustness
Peer-to-Peer Networks

06 Tapestry

Christian Schindelhauer
Technical Faculty
Computer-Networks and Telematics
University of Freiburg