Peer-to-Peer Networks
Pastry & Tapestry
4th Week

Albert-Ludwigs-Universität Freiburg
Department of Computer Science
Computer Networks and Telematics
Christian Schindelhauer
Summer 2008
Peer-to-Peer Networks

Pastry
Pastry

- Peter Druschel
  - Rice University, Houston, Texas
  - now head of Max-Planck-Institute for Computer Science, Saarbrücken/Kaiserslautern
- Antony Rowstron
  - Microsoft Research, Cambridge, GB
- Developed in Cambridge (Microsoft Research)
- Pastry
  - Scalable, decentralized object location and routing for large scale peer-to-peer-network
- PAST
  - A large-scale, persistent peer-to-peer storage utility
- Two names one P2P network
  - PAST is an application for Pastry enabling the full P2P data storage functionality
Pastry Overview

- Each peer has a 128-bit ID: nodeID
  - unique and uniformly distributed
  - e.g. use cryptographic function applied to IP-address
- Routing
  - Keys are matched to \( \{0,1\}^{128} \)
  - According to a metric messages are distributed to the neighbor next to the target
- Routing table has \( O(2^b(\log n)/b) + \ell \) entries
  - \( n \): number of peers

- \( \ell \): configuration parameter
- \( b \): word length
  - typical: \( b = 4 \) (base 16), \( \ell = 16 \)
  - message delivery is guaranteed as long as less than \( \ell/2 \) neighbored peers fail
- Inserting a peer and finding a key needs \( O((\log n)/b) \) messages
Routing Table

- **NodeID presented in base $2^b$**
  - e.g. NodeID: 65A0BA13
- **For each prefix $p$ and letter $x \in \{0, \ldots, 2^b-1\}$ add an peer of form $px^*$ to the routing table of NodeID,** e.g.
  - $b=4$, $2^b=16$
  - 15 entries for $0^*, 1^*, \ldots, F^*$
  - 15 entries for $60^*, 61^*, \ldots, 6F^*$
  - ...
  - if no peer of the form exists, then the entry remains empty
- **Choose next neighbor according to a distance metric**
  - metric results from the RTT (round trip time)
- **In addition choose $l$ neighbors**
  - $l/2$ with next higher ID
  - $l/2$ with next lower ID
Routing Table

- **Example b=2**
- **Routing Table**
  - For each prefix $p$ and letter $x \in \{0,\ldots,2^b-1\}$, add an peer of form $px^*$ to the routing table of NodeID
- **In addition choose $\ell$ neighbors**
  - $\ell/2$ with next higher ID
  - $\ell/2$ with next lower ID
- **Observation**
  - The leaf-set alone can be used to find a target
- **Theorem**
  - With high probability there are at most $O(2^b \frac{\log n}{b})$ entries in each routing table
Routing Table

- **Theorem**
  - With high probability there are at most $O(2^b (\log n)/b)$ entries in each routing table.

- **Proof**
  - The probability that a peer gets the same m-digit prefix is $2^{-bm}$.
  - The probability that a m-digit prefix is unused is $(1 - 2^{-bm})^n \leq e^{-n/2^{bm}}$.
  - For $m = c (\log n)/b$ we get $e^{-n/2^{bm}} \leq e^{-n/2^c \log n} \leq e^{-n/n^c} \leq e^{-n^{c-1}}$.
  - With (extremely) high probability there is no peer with the same prefix of length $(1 + \varepsilon)(\log n)/b$.
  - Hence we have $(1 + \varepsilon)(\log n)/b$ rows with $2^{b}-1$ entries each.
A Peer Enters

- New node $x$ sends message to the node $z$ with the longest common prefix $p$
- $x$ receives
  - routing table of $z$
  - leaf set of $z$
- $z$ updates leaf-set
- $x$ informs $z$-leaf set
- $x$ informs peers in routing table
  - with same prefix $p$ (if $\ell/2 < 2^b$)
- Number of messages for adding a peer
  - $\ell$ messages to the leaf-set
  - expected $(2^b - \ell/2)$ messages to nodes with common prefix
  - one message to $z$ with answer
When the Entry-Operation Errs

- Inheriting the next neighbor routing table does not allow work perfectly
- Example
  - If no peer with 1* exists then all other peers have to point to the new node
  - Inserting 11
  - 03 knows from its routing table
    - 22,33
    - 00,01,02
  - 02 knows from the leaf-set
    - 01,02,20,21
- 11 cannot add all necessary links to the routing tables
Missing Entries in the Routing Table

- Assume the entry \( R_{i,j} \) is missing at peer \( D \)
  - \( j \)-th row and \( i \)-th column of the routing table
- This is noticed if message of a peer with such a prefix is received
- This may also happen if a peer leaves the network
- Contact peers in the same row
  - if they know a peer this address is copied
- If this fails then perform routing to the missing link
Lookup

- Compute the target ID using the hash function
- If the address is within the $l$-leaf set
  - the message is sent directly
  - or it discovers that the target is missing
- Else use the address in the routing table to forward the message
- If this fails take best fit from all addresses
Lookup in Detail

- **L**: \( \ell \)-leafset
- **R**: routing table
- **M**: nodes in the vicinity of \( D \) (according to RTT)
- **D**: key
- **A**: nodeID of current peer
- **R_{ij}**: \( j \)-th row and \( i \)-th column of the routing table
- **L_i**: numbering of the leaf set
- **D_i**: \( i \)-th digit of key \( D \)
- **shl(A)**: length of the largest common prefix of \( A \) and \( D \) (shared header length)

(1) if \( (L_{\lfloor \ell \rfloor /2} \leq D \leq L_{\lceil \ell \rceil /2}) \) {
(2) \(/ \ D \) is within range of our leaf set
(3) forward to \( L_i \), s.th. \( | D - L_i | \) is minimal;
(4) } else {
(5) \(/ / \) use the routing table
(6) Let \( l = \text{shl}(D, A) \);
(7) if \( (R_{ij}^{D_i} \neq \text{null}) \) {
(8) forward to \( R_{ij}^{D_i} \);
(9) }
(10) else {
(11) \(/ / \) rare case
(12) forward to \( T \in L \cup R \cup M \), s.th.
(13) \( \text{shl}(T, D) \geq l \),
(14) \( | T - D | < | A - D | \)
(15) }
(16) }
Routing — Discussion

- If the Routing-Table is correct
  - routing needs \( O((\log n)/b) \) messages

- As long as the leaf-set is correct
  - routing needs \( O(n/l) \) messages
  - unrealistic worst case since even damaged routing tables allow dramatic speedup

- Routing does not use the real distances
  - \( M \) is used only if errors in the routing table occur
  - using locality improvements are possible

- Thus, Pastry uses heuristics for improving the lookup time
  - these are applied to the last, most expensive, hops
Localization of the $k$ Nearest Peers

- Leaf-set peers are not near, e.g.
  - New Zealand, California, India, ...
- TCP protocol measures latency
  - latencies (RTT) can define a metric
  - this forms the foundation for finding the nearest peers
- All methods of Pastry are based on heuristics
  - i.e. no rigorous (mathematical) proof of efficiency
- Assumption: metric is Euclidean
Locality in the Routing Table

- **Assumption**
  - When a peer is inserted the peers contacts a near peer
  - All peers have optimized routing tables

- **But:**
  - The first contact is not necessary near according to the node-ID

- **1st step**
  - Copy entries of the first row of the routing table of P
    - good approximation because of the triangle inequality (metric)

- **2nd step**
  - Contact fitting peer p of p with the same first letter
  - Again the entries are relatively close

- **Repeat these steps until all entries are updated**
Locality in the Routing Table

- **In the best case**
  - each entry in the routing table is optimal w.r.t. distance metric
  - this does not lead to the shortest path

- **There is hope for short lookup times**
  - with the length of the common prefix the latency metric grows exponentially
  - the last hops are the most expensive ones
  - here the leaf-set entries help
Localization of Near Nodes

- Node-ID metric and latency metric are not compatible
- If data is replicated on k peers then peers with similar Node-ID might be missed
- Here, a heuristic is used
- Experiments validate this approach
Experimental Results — Scalability

- Parameter $b=4$, $l=16$, $M=32$
- In this experiment the hop distance grows logarithmically with the number of nodes
- The analysis predicts $4 \log n$
- Fits well
Experimental Results
Distribution of Hops

- Parameter $b=4$, $l=16$, $M=32$, $n = 100,000$

- Result
  - deviation from the expected hop distance is extremely small

- Analysis predicts difference with extremely small probability
  - fits well
Experimental Results — Latency

- Parameter $b=4$, $l=16$, $M=3$
- Compared to the shortest path astonishingly small
  - seems to be constant

![Graph showing relative distance vs. number of nodes]
Critical View at the Experiments

- Experiments were performed in a well-behaving simulation environment
- With $b=4$, $L=16$ the number of links is quite large
  - The factor $2^b/b = 4$ influences the experiment
  - Example $n=100\,000$
    - $2^b/b \log n = 4 \log n > 60$ links in routing table
    - In addition we have 16 links in the leaf-set and 32 in $M$
- Compared to other protocols like Chord the degree is rather large
- Assumption of Euclidean metric is rather arbitrary
Experimentelle Untersuchungen
Knotenausfälle

- Parameter \( b=4, l=16, M=32, n = 5\,000 \)
- No fail: vor Ausfall
- No repair: 500 von 5000 Peers fallen aus
- Repair: Nach Reparatur der Routing-Tables
Peer-to-Peer Networks

Tapestry

Zhao, Kubiatowicz und Joseph (2001)
Tapestry

- 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)
Neighborhood of a Peer (1)

- 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
Neighborhood Set (2)

- For each prefix $x$ and all letters $j$ of the peer with ID $A$
  - establish a link to a node with prefix $xj$ within the neighborhood set $N_{x,j}^A$
- Peer with Node-ID $A$ has $b |A|$ neighborhood sets
- The neighborhood set of contains all nodes with prefix $sj$
  - Nodes of this set are denoted by $(x,j)$
Example of Neighborhood Sets

Neighborhood set of node 4221

<table>
<thead>
<tr>
<th>Level 4</th>
<th>Level 3</th>
<th>Level 2</th>
<th>Level 1</th>
</tr>
</thead>
</table>
| j=0     | 4220    | 420?    | 40??    | 0???
| j=1     | 4221    | 421?    | 41??    | 1???
| .       | 4222    | 422?    | 42??    | 2???
| .       | 4223    | 423?    | 43??    | 3???
| .       | 4224    | 424?    | 44??    | 4???
| .       | 4225    | 425?    | 45??    | 5???
| .       | 4226    | 426?    | 46??    | 6???
| j=7     | 4227    | 427?    | 47??    | 7???

Peer-to-Peer-Networks
Summer 2008

Dienstag, 20. Mai 2008
Links

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

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

Note:
- some neighborhood sets are empty
Properties of Neighborhood Sets

- **Consistency**
  - If $N_{x,j}^A = \emptyset$ 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 $b_1b_2...b_n$

\[
\begin{align*}
N_{\emptyset,b_1}^A & \text{ 1st hop to node } A_1 \\
N_{b_1,b_2}^A & \text{ 2nd hop to node } A_2 \\
N_{b_1b_2,b_3}^A & \text{ 3rd hop to node } A_3 \\
\vdots & \\
\end{align*}
\]
Locality

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

- **Primary node of a neighborhood set** \( N_{x,j}^A \)
  - 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
Root Node

- Object with ID Y should 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

- Surrogate Routing
  - compute a surrogate (replacement root node)
  - If \((x,j)\) is a hole, then choose \((x,j+1),(x,j+2),\ldots\) 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
Lookup

- 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

Diagram:

- Peer A
- Peer B (marked with an X)
- Root

A green arrow points from Peer A to Peer B's location.
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

End of 4th Week

Albert-Ludwigs-Universität Freiburg
Department of Computer Science
Computer Networks and Telematics
Christian Schindelhauer
Summer 2008