Peer-to-Peer Systems

DHT examples, part 3
(Symphony, Viceroy, Distance Halving, Koorde)

Michael Welzl  michael.welzl@uibk.ac.at

DPS NSG Team  http://dps.uibk.ac.at/nsg
Institute of Computer Science
University of Innsbruck, Austria

Symphony

• Key idea: Distributed Hashing in a Small World
  - Start with Chord, but discard the strong requirements on the routing table (finger table); rely on small world (random links) to reach the destination

• Construction
  - Map the nodes and keys to the ring
  - Link every node with its successor and predecessor
  - Add $k$ random links with probability proportional to $1/(d \cdot \log N)$, where $d$ is the distance on the ring
  - Lookup time $O(\log^2 N)$
  - If $k = \log N$ lookup time $O(\log N)$
  - Easy to insert and remove nodes
    (perform periodical refreshes for the links)
      • $O(\log^2 N)$ expected messages
Symphony /2

Nodes arranged in a unit circle (perimeter = 1)
Arrival ⇒ Node chooses position along circle uniformly at random
Each node has 1 short link (next node on circle) and k long links

Adaptation of Small World Idea: [Kleinberg00]
Long links chosen from a probability distribution function:
\[ p(x) = \frac{1}{x \log n} \] where \( n \) = #nodes.

Simple greedy routing:
"Forward along that link that minimizes the absolute distance to the destination."
Average lookup latency = \( O((\log_2 n) / k) \) hops

Fault Tolerance:
No backups for long links! Only short links are fortified for fault tolerance.

Symphony /3

• Key problem: network size estimation
  - Based on family of harmonic functions (as PDF), hence the name

\[ x = \text{Length of arc} \]
\[ \frac{1}{x} = \text{Estimate of n} \]
\[ p(x) = \frac{1}{(x \log n)} \]

• Symphony optimizations:
  - Bi-directional Routing
    • Exploit both outgoing and incoming links!
    • Route to the neighbor that minimizes absolute distance to destination
    • Reduces avg latency by 25-30%
  - 1-Lookahead
    • List of neighbor’s neighbors; reduces avg. latency by 40%
Viceroy

- is a butterfly

- Butterfly = well known network topology with some desirable properties
  - Small degree (4) and small (proven to be close to optimal) diameter
  - Logarithmic path length between any two nodes
  - Simple routing, no bottlenecks, high resilience

Thanks to google :-)

Routing in a butterfly network

- That’s a bit complicated and inefficient
- Hence, in Viceroy, nodes are also connected within each level
Viceroy network

- Arrange nodes and keys on a ring
  - like in Chord

- Assign to each node a level value
  - chosen uniformly from the set \{1, \ldots, \log n\}
  - estimate $n$ by taking the inverse of the distance of the node with its successor
  - easy to update

Viceroy network /2

- Create a ring of nodes within the same level
  - In addition to a "general" ring of all nodes

- Each node $x$ at level $i$ has two downward links to level $i+1$
  - a left link to the first node of level $i+1$ after position $x$ on the ring
  - a right link to the first node of level $i+1$ after pos. $x + (\frac{1}{2})^i$
**Downward links**

- Each node $x$ at level $i$ has an upward link to the next node on the ring at level $i-1$.

**Upward links**

- Each node $x$ at level $i$ has an upward link to the next node on the ring at level $i-1$. 
**Viceroy: Joining**

1. Insert peer at random position of the general ring
2. Estimate $\log n$ by looking at the distance between a node and its successor
3. Randomly pick level $i$ (uniformly distributed between 1 and $\log n$)
4. Find position in ViceRoy network via lookup starting at the ring neighbor
5. Insert peer into the ViceRoy network level by level
   - Inserting peer in ring $i$ of the network
   - Finding the...
     - Successor of $(i, x)$
     - Successor of $(i+1, x)$
     - Successor of $(i+1, x+2^i)$
     - Predecessor of $(i-1, x)$
     - Predecessor of $(i-1, x-2^i)$
   - ...starting at the edges connected to the neighbor in ring $i$

- **Complexity**
  - Lookup time ($O(\log n)$) +
  - Finding the successor / predecessor ($O(\log n)$)

**Viceroy: Searching**

- Peer $(i, x)$ gets search request for $(j, y)$
    
    IF $i=j$ und $|x-y| \leq (\log n)^2/n$ THEN
    Forward search request to neighbor of ring $i$
    ELSE
    IF $y$ is to the right of $x+2^i$ THEN
      Forward request to successor of $(i+1, x+2^i)$
    ELSE
      Forward request to $Z = \text{successor of } (i+1, x)$
      IF successor $Z$ is to the right of $x$ THEN
        Search a node $(i+1, p)$ with $p < x$ on the ring $(i+1)$, starting at $Z$
      FI
      FI
    FI
    FI

- With a high probability, this takes time (and messages) of $O(\log n)$
Viceroy conclusion

- First Peer-to-Peer network with constant in- and outdegree
  - Outdegree 8, Indegree should be constant
    - additional "multiple choice" mechanism was added to insertion procedure to truly make it constant
      (not included on previous slide about insertion for simplicity)
- ...but:
  - Multiple ring structure quite complex
  - Multiple choice method causes $O(\log 2n)$ insertion complexity
  - As we will see, there are easier networks with similar properties

Distance Halving

- Published by Moni Naor and Udi Wieder in 2003
  - Moni Naor is also a coauthor of the Viceroy paper :-)
- Based on continuous graphs
  - Infinite graphs with continuous node and edge set
- In Distance Halving:
  - Nodes: $x \in [0,1)$
  - Edges:
    - Left-edges: $(x,x/2)$
    - Right-edges: $(x,1/2+x/2)$
    - and edges back:
      - $(x/2,x)$
      - $(1/2+x/2,x)$
  - Note that distance halves with every step $\Rightarrow$ hence the name :-)

Insertion:

- $(x,x/2)$
- $(x,1/2+x/2)$

Discretization

• Consider fixed number of discrete intervals in the continuous space
  - formed by successive halving

• Insert an edge between intervals A and B if there are \( x \in A \) and \( y \in B \) such that \((x,y)\) is an edge of the continuous graph

• Possible (simple) implementation:
  peers pick a random position in \([0,1)\)
  - They are responsible for data from their position to their successor

• Neighboring intervals are also bidirectionally connected (ring)

Multiple choice principle

• Goal, as in Viceroy: constant degree
  - Emerges if ratio between largest and smallest interval is constant
  - With a high probability, largest interval = \(2/n\), smallest interval = \(1/(2n)\)
  \[ \Rightarrow \text{constant degree} \]
  - \samethingsd\ and logarithmic diameter

• Degree of 4 can be achieved via multiple choice principle for joining (goal: evenly spread nodes across range):
  - Send \(c \log n\) queries to randomly chosen intervals
  - Select largest interval and halve it
  - Update ring edges
  - Update left- and right-edges

• Time and number of messages for inserting peers: \(O(\log^3 n)\)
Routing

- Distance is halved with each step
  - $O(\log n)$ hops and messages

- Example algorithm:
  Left-Routing (src, dst)
  - IF dst is in neighbor interval
    - Forward query to dst
  - ELSE
    - newSrc = left-edge(src);
    - newDst = left-edge(dst);
    - Send message from src to newSrc;
    - Left-Routing(newSrc, newDst);
    - Send message from newDst to dst;

- Note: this only uses left-edges
  - Could also be done with right-edges only

Routing and conclusion

- Left- and right-edges can be combined using an arbitrary strategy (alternate, random, ..)
  - Congestion (number of packets transmitted by each peer) is $O(\log n)$ in the worst case (when every peer sends a request)
  - Proof based on similar proof for hypercube; same result can also be shown for Viceroy

- Conclusion: simple and efficient structure
  - degree $O(1)$, diameter $O(\log n)$, lookup $O(\log n)$, join $O(\log^2 n)$, load balancing

- Principle of discretizing continuous graphs also used in other DHTs
  - But this is the first time the problem was explicitly formulated like this
Enhancing Chord: degree or diameter?

- Chord: degree $O(\log n)$, diameter $O(\log n)$
  - Making these smaller is desirable

- Question 1: can we get a smaller diameter with degree $g=O(\log n)$?
  - Distance 1: at most $g$ nodes
  - Distance 2: at most $g^2$ nodes
  - $\Rightarrow$ thus, distance $d$: $g^d$ nodes

- Hence: $(\log n)^d = n$

- It follows that: $d = \frac{\log n}{\log \log n}$

- Therefore, only minor improvement of diameter possible

Koorde


- Goal: maintain Chord’s $O(\log n)$ diameter make in- and outdegree $= 2$
  - This can be done with a binary tree, a butterfly net, a DeBruijn graph...

- Foundation: operations on binary string $S$ of length $m$
  - Shuffle:
    - $\text{shuffle}(s_1, s_2, s_3, \ldots, s_m) = (s_2, s_3, \ldots, s_m, s_1)$
  - Exchange:
    - $\text{exchange}(s_1, s_2, s_3, \ldots, s_m) = (s_1, s_2, s_3, \ldots, s_m)$
  - Shuffle-Exchange:
    - $\text{SE}(S) = \text{exchange}(\text{shuffle}(S))$
      - $(s_2, s_3, \ldots, s_m, s_1)$
Shuffle and exchange

• Any string A can be turned into any string B by applying Shuffle and Shuffle-exchange operations m times

Example:
From 0 1 1 1 0 1 1 to 1 0 0 1 1 1 1 via SE SE SE S SE S S operations

The DeBruijn Graph

• A DeBruijn graph consists of n=2^m nodes
  - represented as m-digit binary numbers

• Every node has two outgoing edges
  - 1. edge points from u to shuffle(u)
  - 2. edge points from u to SE(u)

• DeBruijn-Graph has degree 2 and m=\log_2 n diameter

• Koorde = Ring + DeBruijn-Graph
  - Ring of 2^m nodes + DeBruijn edges
Koorde = Ring + DeBruijn Graph

- **Edges**
  - shuffle(s₁, s₂, ..., sₘ) = (s₂, ..., sₘ, s₁)
  - shuffle(x) = (x div 2ᵐ)+ (2x) mod 2ᵐ
  - SE(s) = (s₁, s₂, ..., sₘ, ¬ s₁)
  - SE(x) = 1-(x div 2ᵐ)+ (2x) mod 2ᵐ
  - (x div 2ᵐ) can be either 0 or 1
  - successors of x are 2x mod 2ᵐ and 2x+1 mod 2ᵐ

- **Exactly 2ᵐ nodes unlikely in a P2P network**
  - Choose large m (typically 128 or 160) ⇒ more DeBruijn nodes than peers

- **Unoccupied DeBruijn nodes become “virtual nodes”**
  - Every peer manages all DeBruijn nodes between itself and successor
  - Only necessary for incoming edges

Koorde properties

- **Per definition, four edges per node**

- **With high probability**
  - at most O(log n) incoming edges per node
  - Reason:
    - distance to next peer is at most c (log n)/2ᵐ (with high probability)
    - this is the max. interval from which peers can point to a peer (and its virtual nodes)
    - Within this interval, there are at most O(log n) peers with high probability
  - Diameter = O(log n)
  - Routing requires O(log n) messages

- **But low coherence of Koorde graph**
K-degree DeBruijn Graph

- Consider alphabet over k letters, e.g. k = 3
- Each k-DeBruijn-node x has successors
  - \((kx \mod m), (kx + 1 \mod m), (kx + 2 \mod m), \ldots, (kx + k - 1 \mod m)\)
- Diameter becomes \((\log m) / (\log k)\)
- Coherence grows with k

References / acknowledgments

- Slides from:
  - Jussi Kangasharju
  - Christian Schindelhauer
  - Klaus Wehrle