Coordinate-based Routing: Refining NodeIds in Structured Peer-to-Peer Systems

Fabian Hartmann and Bernhard Heep

P2PNet’09 (in conjunction with ICUMT’09)
St. Petersburg, Russia
Motivation

Structured P2P overlays offering key-based routing (KBR)

- Various service possible
  - e.g. decentralized directory services
- Broad range of available protocols
  - Chord, Pastry, Bamboo, Kademlia, Broose, Koorde, CAN, …
  - usually O(log N) hops per message
- Problem: high routing latencies
  \[ \text{#hops} \times d_{\text{avg}} \] (recursive routing)

- Decreasing routing latencies by exploiting network coordinate systems
Agenda

Goals
- Optimization of KBR paths
- Speed-up of DHT get()-operations

Related Work

Problems and solutions
- Non-uniform nodeId distribution

Implementation
- Overlay Framework OverSim

Evaluation

Summary and Outlook
Paths in prefix-based KBR Overlays

- **siblings**: Close nodes in ID space
- **neighbors**: Physically close nodes in underlay
- Nodelds uniform distributed

→ Usually: Siblings ≠ Neighbors
Goals

- Minimizing KBR latencies in prefix-based peer-to-peer overlays
  - Enabling relationship between underlay and overlay:
    siblings ↔ neighbors
  - More efficient routing
  - Faster results in DHTs

**Idea**: Mapping of underlay on corresponding nodeIds prefix (like a city’s area code)

1. Defining node positions using network coordinate systems (NCS)
2. Mapping of network coordinates onto nodeIds:
   - Topology-based NodeId Assignment
Related Work

- **Topologically-sensitive** construction of CAN [1]
  - Latency measurements to dedicated nodes (landmarks)
  - $m$ landmarks $\rightarrow m!$ RTT-orderings
  - 1st dimension gets divided into $m$ parts,
    2nd dimension gets divided into $m-1$ parts, ...
  - lower latencies but non-uniform nodeId distribution

- **Canary** [2]
  - CAN using Vivaldi-based nodeIds
  - very low latencies but non-uniform nodeId distribution

---


Problems to solve…

P1: How can nodes be aware of their underlay position?

P2: What mapping (position → prefix) should be used?
   - Need for a well-defined mapping, known by all nodes

P3: How can load balancing be achieved?
   - Uniform distributed hashes vs. non-uniform distributed node positions / coordinates

P4: How can replicas reallocated?
   - Replicas usually on siblings in DHTs
     → CBR: Hotspots in geographic areas
     → All replicas could be lost if subnet fails
Usage of **GNP / NPS**

- Nodes measure latency to n+1 **landmark nodes**
  - Nodes are placed in a synthetic n-dimensional Euclidean space
- Internet latencies: Triangle equality not valid
  - Coordinates are error-prone

**Why no decentralized NCS? (Vivaldi, ...)**

- Partitioning of the underlay topology in prefix areas using a global picture
- **Fixed base** for global picture of coordinate distribution needed
  - Coordinate space is spanned by landmark nodes using their coordinates as basis
P2: Find a function

\[ f : \mathbb{R}^d \rightarrow P \quad \text{with} \quad P = \{p_i \in \mathbb{N}_0 \mid p_i < 2^i, 1 \leq i \leq \text{max}\} \]

→ Partitioning: Bisection of coordinate space for each dimension

P3: Simple cutting into halves leads to non-uniform node number in each area

→ Usage of global picture:
  Bisection according to distribution of nodes
Coordinate-based Routing (CBR)

- Divide the underlay into $2^d$ main areas
- Subdivide areas into prefix areas
  → all according to coordinate distribution
- Overlay hops leading target-oriented to the destination key
- CBR combined with Proximity Neighbor Selection (PNS)
Gaining Global Knowledge

- Need for a **Data Harvesting Phase**
  - Latency measurements (landmarks / nodes)
  - n-dimensional picture with fixed basis (GNP)
  - Here: 2-dim. Skitter data (usually: 5d-7d)

- Partitioning according to CBR rules
Gathering of replicas in one region should be avoided!

**Solution:** Multiple hashes $K_n$ of value $V$ \( K_n \in C, \text{ with } C = \{K | K = H^i(V), 0 < i \leq m\} \)

Search keys are spread over whole network

**Proximity-aware choice** of DHT replicas possible
Step-by-step: Preparation of CBR

(1) Landmark Initialization
(2) Data Harvesting Phase
(3) Creation of Global Knowledge
(4) Utilization
Integration of CBR into overlay framework OverSim [3]

- Extended Simple Underlay: Violation of triangle Inequality [4]
- Central module for providing CBR information
  - Global Knowledge: mapping \rightarrow prefixes
- GNP/NPS coordinate system integrated into OverSim’s NeighborCache module
- New DHT module on Tier 1: Reallocation of replicas
  \rightarrow common DHT-API: \texttt{put()}, \texttt{get()}


Simulation: Set-up and Parameters

- Evaluated protocols with CBR: *Pastry, Bamboo*
  - 2500 nodes (4500 without churn), 20 random seeds
  - 2h measurement time after network initialization
  - Churn: *weibull-distributed lifetime model* [5][6]
- Varied parameters:
  - `bitsPerDigit` {1, 2, 4}
  - `Churn` {no churn, moderate churn}
  - `Network coordinate system` {GNP, NPS (maxLayer = 3)}
  - `CBR stopAtDigit` {noCBR, 1, 2, 3, 4}
  - `DHT replicas` {1, 2, 3, 6}


Evaluation: CBR without Churn

Pastry: Latency decrease up to 13%
Bamboo: up to 20%
NPS leads to higher latencies due to deviation of coordinates from global knowledge
Pastry: Latency decrease up to 37%
Bamboo: up to 16%
Like in no churn scenarios: NPS comes with higher latencies
Evaluation: CBR-based Replication

- Significant speed-up of get()-operations: Up to 61% decrease with 6 spread replicas
- Effect is observable with up to 2 manipulated digits
Problems and TODOs

- Landmark infrastructure must be provided
- Coordinates must be accurate
  → else latencies increase
- Mapping → Global Knowledge
  - Must be distributed
    (e.g. during bootstrapping procedure)
  - Node distribution must not change
Summary:
- CBR significantly decreases KBR latencies
- CBR-based DHT replication strategy decreases latencies of `get()`-operations up to 60%

Future Work:
- Evaluation in real networks and testbeds like PlanetLab and G-Lab
- Usage of decentralized NCS
- Mobility?
Thank you!

Any Questions?