Data Structure of Chord

For each peer
- successor link on the ring
- predecessor link on the ring
- for all $i \in \{0, \ldots, m-1\}$
  - $\text{Finger}[i] := \text{the peer following the value } r_V(b+2^i)$

For small $i$ the finger entries are the same
- store only different entries

Chord
- needs $O(\log n)$ hops for lookup
- needs $O(\log^2 n)$ messages for inserting and erasing of peers
Routing-Techniques for CHORD: DHash

Frank Dabek, Jinyang Li, Emil Sit, James Robertson, M. Frans Kaashoek, Robert Morris (MIT) „Designing a DHT for low latency and high throughput“, 2003

Idea
- Take CHORD

Improve Routing using
- Data layout
- Recursion (instead of Iteration)
- Next Neighbor-Election
- Replication versus Coding of Data
- Error correcting optimized lookup

Modify transport protocol
Data Layout

- Distribute Data?
- Alternatives
  - Key location service
    - store only reference information
  - Distributed data storage
    - distribute files on peers
  - Distributed block-wise storage
    - either caching of data blocks
    - or block-wise storage of all data over the network
Recursive Versus Iterative Lookup

- **Iterative lookup**
  - Lookup peer performs search on his own

- **Recursive lookup**
  - Every peer forwards the lookup request
    - The target peer answers the lookup-initiator directly

- **DHash++ chooses recursive lookup**
  - speedup by factor of 2
Recursive Versus Iterative Lookup

- DHash++ chooses recursive lookup
  - speedup by factor of 2
Next Neighbor Selection

- **RTT**: Round Trip Time
  - time to send a message and receive the acknowledgment

- **Method of Gummadi, Gummadi, Grippe, Ratnasamy, Shenker, Stoica, 2003, „The impact of DHT routing geometry on resilience and proximity“**
  - Proximity Neighbor Selection (PNS)
    - Optimize routing table (finger set) with respect to (RTT)
    - method of choice for DHASH++
  - Proximity Route Selection (PRS)
    - Do not optimize routing table choose nearest neighbor from routing table
\[ d_0 = d \]
\[ d_1 = \frac{2}{3} d \]
\[ d_2 = \frac{2}{3} d_1 = \left( \frac{2}{3} \right) d \]
\[ d_3 = \left( \frac{2}{3} \right)^2 d \]
\[ \vdots \]
\[ d_k = \left( \frac{2}{3} \right)^k d \]

\[ x \leq \left( \log_{\frac{3}{2}} \right) \cdot \log_{\frac{3}{2}} n \]

\[ \left( \frac{2}{3} \right)^x \leq \frac{1}{n^c} \]
Next Neighbor Selection

- Gummadi, Gummadi, Grippe, Ratnasamy, Shenker, Stoica, 2003, „The impact of DHT routing geometry on resilience and proximity“
  - Proximity Neighbor Selection (PNS)
    - Optimize routing table (finger set) with respect to (RTT)
    - method of choice for DHASH++
  - Proximity Route Selection (PRS)
    - Do not optimize routing table; choose nearest neighbor from routing table
- Simulation of PNS, PRS, and both
  - PNS as good as PNS+PRS
  - PNS outperforms PRS
Next Neighbor Selection

- DHash++ uses (only) PNS
  - Proximity Neighbor Selection
- It does not search the whole interval for the best candidate
  - DHash++ chooses the best of 16 random samples (PNS-Sample)
Next Neighbor Selection

- DHash++ uses (only) PNS
  - Proximity Neighbor Selection
- \( e (0.1, 0.5, 0.9) \)-percentile of such a PNS-Sampling

![Graph showing average lookup latency vs. number of PNS samples]
Cumulative Performance Win

- Following speedup
  - Light: Lookup
  - Dark: Fetch
  - Left: real test
  - Middle: simulation
  - Right: Benchmark latency matrix
Modified Transport Protocol
Discussion DHash++

- Combines a large quantity of techniques
  - for reducing the latency of routing
  - for improving the reliability of data access

- Topics
  - latency optimized routing tables
  - redundant data encoding
  - improved lookup
  - transport layer
  - integration of components

- All these components can be applied to other networks
  - some of them were used before in others
  - e.g. data encoding in Oceanstore

- DHash++ is an example of one of the most advanced peer-to-peer networks