Peer-to-Peer Networks
04: Chord

Christian Schindelhauer
Technical Faculty
Computer-Networks and Telematics
University of Freiburg
Why Gnutella Does Not Really Scale

- **Gnutella**
  - graph structure is random
  - degree of nodes is small
  - small diameter
  - strong connectivity

- **Lookup is expensive**
  - for finding an item the whole network must be searched

- **Gnutella’s lookup does not scale**
  - reason: no structure within the index storage
Two Key Issues for Lookup

- Where is it?
- How to get there?

Napster:
  - Where? on the server
  - How to get there? directly

Gnutella
  - Where? don't know
  - How to get there? don't know

Better:

- Where is x?
  - at f(x)

- How to get there?
  - all peers know the route
Distributed Hash-Table (DHT)

- Hash table
  - does not work efficiently for inserting and deleting

- Distributed Hash-Table
  - peers are „hashed“ to a position in an continuos set (e.g. line)
  - index data is also „hashed“ to this set

- Mapping of index data to peers
  - peers are given their own areas depending on the position of the direct neighbors
  - all index data in this area is mapped to the corresponding peer

- Literature
Entering and Leaving a DHT

- **Distributed Hash Table**
  - peers are hashed to position
  - index files are hashed according to the search key
  - peers store index data in their areas

- **When a peer enters**
  - neighbored peers share their areas with the new peer

- **When a peer leaves**
  - the neighbors inherit the responsibilities for the index data
Features of DHT

- Advantages
  - Each index entries is assigned to a specific peer
  - Entering and leaving peers cause only local changes

- DHT is the dominant data structure in efficient P2P networks

- To do:
  - network structure
Chord

- Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek and Hari Balakrishnan (2001)
- Distributed Hash Table
  - range \{0,..,2^m-1\}
  - for sufficient large m
- Network
  - ring-wise connections
  - shortcuts with exponential increasing distance
Chord as DHT

- n number of peers
- V set of peers
- k number of data stored
- K set of stored data
- m: hash value length
  - \( m \geq 2 \log \max\{K,N\} \)
- Two hash functions mapping to \( \{0,..,2^{m-1}\} \)
  - \( r_V(b) \): maps peer to \( \{0,..,2^{m-1}\} \)
  - \( r_K(i) \): maps index according to key
    i to \( \{0,..,2^{m-1}\} \)
- Index i maps to peer
  \( b = f_V(i) \)
  - \( f_V(i) := \arg \min_{b \in V} \{(r_V(b)-r_K(i)) \mod 2^m\} \)
For each peer
- successor link on the ring
- predecessor link on the ring
- for all $i \in \{0,..,m-1\}$
  - $\text{Finger}[i] :=$ the peer following the value $rV(b+2^i)$

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

Lemma
- The number of different finger entries is $O(\log n)$ with high probability, i.e. $1-n^{-c}$. 
Theorem
- We observe in Chord for n peers and k data entries
  • Balance&Load: Every peer stores at most $O(k/n \log n)$ entries with high probability
  • Dynamics: If a peer enters the Chord then at most $O(k/n \log n)$ data entries need to be moved

Proof
- …
Properties of the DHT

- Lemma
  - For all peers $b$ the distance $|r_{\sqrt{V}}(b.\text{succ}) - r_{\sqrt{V}}(b)|$ is
    - in the expectation $2^m/n$,
    - $O((2^m/n) \log n)$ with high probability (w.h.p.)
    - at least $2^m/n^{c+1}$ für a constant $c>0$ with high probability
  - In an interval of length $w$ $2^m/n$ we find
    - $\Theta(w)$ peers, if $w=\Omega(\log n)$, w.h.p.
    - at most $O(w \log n)$ peers, if $w=O(\log n)$, w.h.p.

- Lemma
  - The number of nodes who have a pointer to a peer $b$ is $O(\log^2 n)$ w.h.p.
**Lookup in Chord**

- **Theorem**
  - The Lookup in Chord needs $O(\log n)$ steps w.h.p.

- **Lookup for element $s$**
  - **Termination($b,s$):**
    - if peer $b,b' = b.\text{succ}$ is found with $r_K(s) \in [r_V(b),r_V(b')]$
  - **Routing:**
    - Start with any peer $b$
      - while not Termination($b,s$) do
        - for $i=m$ downto 0 do
          - if $r_K(s) \in [r_V(b.\text{finger}[i]),r_V(\text{finger}[i+1])]$ then
            - $b \leftarrow b.\text{finger}[i]$
          fi
        od
Theorem
- The Lookup in Chord needs $O(\log n)$ steps w.h.p.

Proof:
- Every hops at least halves the distance to the target
- At the beginning the distance is at most
- The minimum distance between is $2^m/n^c$ w.h.p.
- Hence, the runtime is bounded by $c \log n$ w.h.p.
**Lemma**

- The out-degree in Chord is $O(\log n)$ w.h.p.
- The in-degree in Chord is $O(\log_2 n)$ w.h.p.

**Proof**

- The minimum distance between peers is $2^m/n^c$ w.h.p.
  - this implies that the out-degree is $O(\log n)$ w.h.p.
- The maximum distance between peers is $O(\log n \cdot 2^m/n)$ w.h.p.
  - the overall length of all line segments where peers can point to a peer following a maximum distance is $O(\log^2 n \cdot 2^m/n)$
  - in an area of size $w=O(\log^2 n)$ there are at most $O(\log^2 n)$ w.h.p.
Inserting Peer

- **Theorem**
  - For integrating a new peer into Chord only $O(\log^2 n)$ messages are necessary.
Adding a Peer

- First find the target area in $O(\log n)$ steps
- The outgoing pointers are adopted from the predecessor and successor
  - the pointers of at most $O(\log n)$ neighbored peers must be adapted
- The in-degree of the new peer is $O(\log^2 n)$ w.h.p.
  - Lookup time for each of them
  - There are $O(\log n)$ groups of neighbored peers
  - Hence, only $O(\log n)$ lookup steps with at most costs $O(\log n)$ must be used
  - Each update of has constant cost
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 blacks
    - 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
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
Cumulative Performance Win

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

![Graph showing cumulative probability vs latency for STP and TCP](image-url)
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
Peer-to-Peer Networks
04: Chord

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