P2P network *concept presentation*

High level languages: Rust

A. Ellwanger T. Erdelt A. Griesbeck

January 8, 2019

Ludwig Maximilian University of Munich

Chord Algorithm

- Introduced in 2001 by MIT[1]
- Algorithm for a peer-to-peer distributed hash table (DHT): Key/value pairs get stored distributed in the network by different nodes
- *Identifier*: A consistent hash function assigns each node and each key an m-bit identifier using SHA 1 ($m =$ number big enough to make collisions improbable)
- Both are uniformly distributed
- Both exist the same ID space
- A key k is assigned to the node whose identifier is equal to or greater than the key's
- Nodes arranged in **circle structure** by ascending $identifiers(nodes)$

---

[1] Stoica, I., Morris, R., Karger, D., Kaashoek, M. F., & Balakrishnan, H. (2001). Chord: A scalable peer-to-peer lookup service for internet applications http://doi.org/10.1145/964723.383071

Chord Algorithm - Assignment of keys to nodes

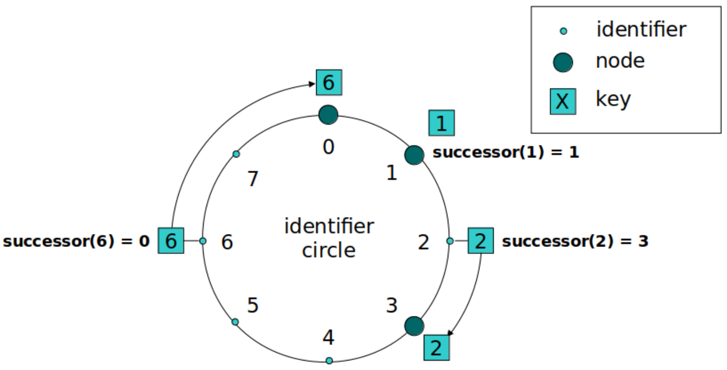


Figure: https://web.archive.org/web/20190108111028/https://people.eecs.berkeley.edu/~kubitron/courses/cs294-4-F03/slides/lec03-chord.ppt

Chord Algorithm - Node *n*

- *Successor*(n): Next node s in the circle structure ($identifier(s) > identifier(n)$)
- *Predecessor*(n): Previous node p in the circle ($identifier(p) < identifier(n)$)
- Finger table: stores x closest nodes
- Storage: Stores y key/value pairs

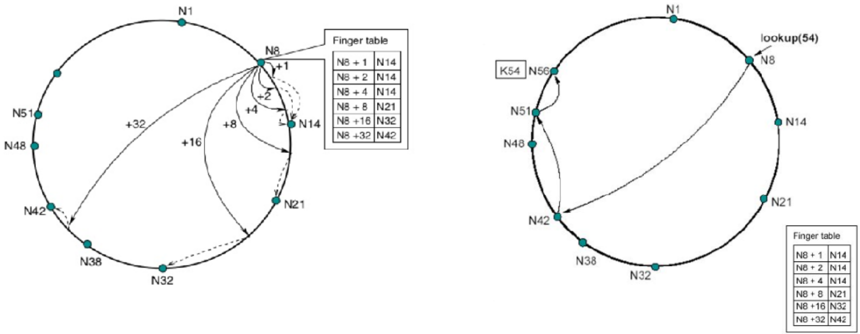
Chord Algorithm - how it works (1)

- Value look-up by key k
    - Query local storage for k
    - If key can't be found on current node, contact node which is closest to $successor(k)$
- Joining of new nodes:
    - Initialise new node n
    - Find $s = successor(n)$ based on identifier
    - Set $predecessor(n) = predecessor(s)$ and $predecessor(s) = n$

Chord Algorithm - how it works (2)

- Stabilisation
  - Finger tables, predecessors & successors of each node get updated periodically to react on node dropouts
- Redundancy
  - Has to be implemented manually e.g. by storing key/value pairs on multiple nodes

Lookup example



Finger table

N8 + 1	N14
N8 + 2	N14
N8 + 4	N14
N8 + 8	N21
N8 +16	N32
N8 +32	N42

Finger table

N8 + 1	N14
N8 + 2	N14
N8 + 4	N14
N8 + 8	N21
N8 +16	N32
N8 +32	N42

Figure:
https://web.archive.org/web/20190108111201/http://resources.mpi-inf.mpg.de/d5/teaching/ws03_04/p2p-data/11-18-paper1.ppt

Key libraries and crates

- std::net
  Networking primitives for TCP/UDP communication
- std::collections::HashMap
- sha1 - `https://crates.io/crates/sha1`
  Minimal implementation of SHA1
- tokio - `https://crates.io/crates/tokio`
  Event-driven, non-blocking I/O platform for writing asynchronous apps
- ...

Custom data structure

```
struct Node {
            predecessor: (i32, IpAddr),
            fingerTable: HashMap<i32, IpAddr>,
            storage: HashMap<str, str>,
}
```

Proof of concept application

- Not finally decided yet:
  1. Chat: Use Chord to find IP for username then establish connection directly
  2. Chat: Use modified Chord to route messages
  3. Collaborative mirroring of files
  4. Distributed file storage
- Feedback welcome!