Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications

Speaker: Cathrin Weiß

11/23/2004

Proseminar

Peer-to-Peer Information Systems

Overview

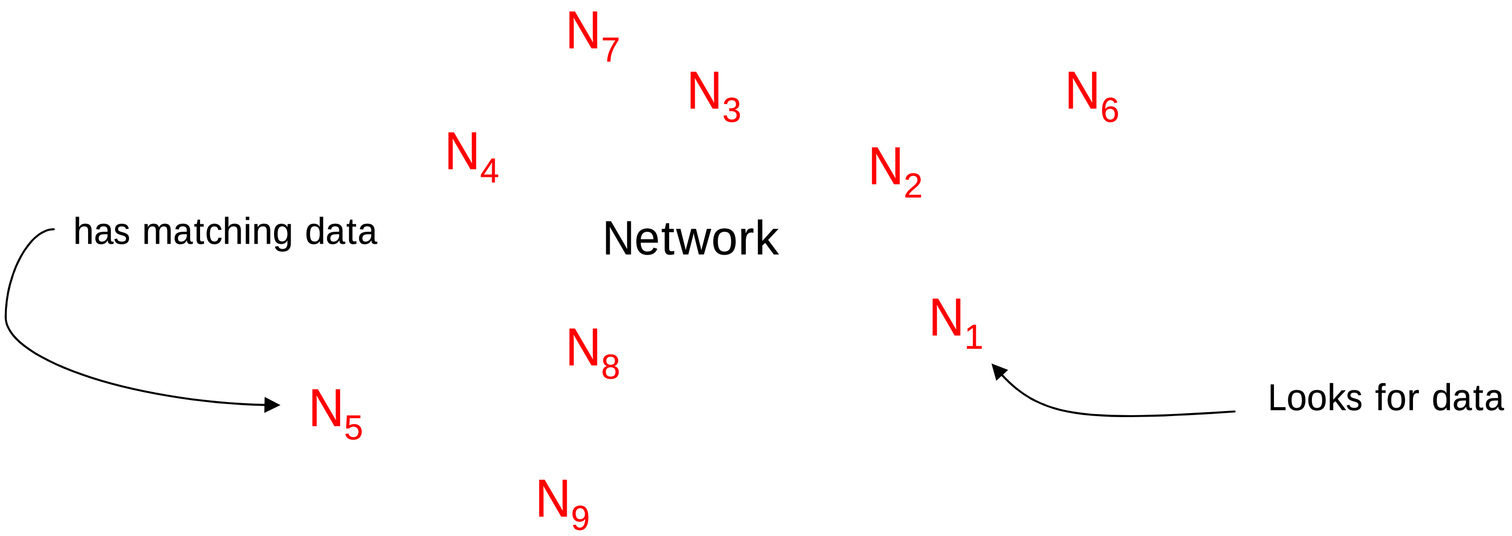
- I. Introduction
- II. The Chord Protocol and how it works
- III. Disadvantages and Improvements
- IV. Applications using Chord
- V. Summary

I. Introduction

- **Problem:**

  Decentralized network with several peers (clients)

  How to find specific peer that hosts desired data within this network?

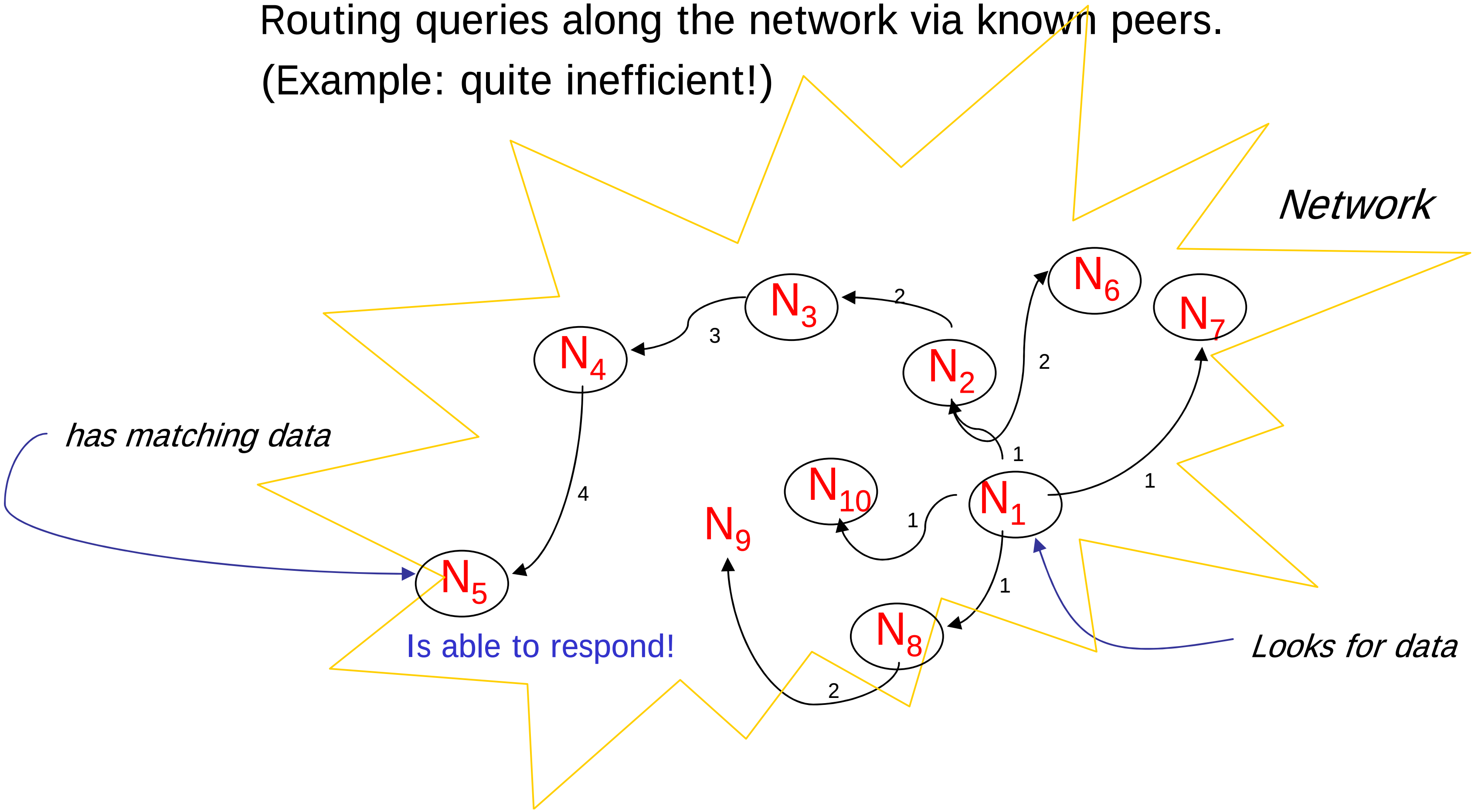


I. Introduction

- **Solution:**

  Routing queries along the network via known peers.

  (Example: quite inefficient!)



Overview

- I. Introduction
- II. The Chord Protocol and how it works
- III. Disadvantages and Improvements
- IV. Applications using Chord
- V. Summary

II. The Chord Protocol and how it works

What is Chord?

- Efficient
- Simple
- Provable Performant
- Provable Correct
- Scalable
- Supporting only one operation: mapping a given key onto a node

lookup protocol

II. The Chord Protocol and how it works

Consistent Hashing:

Using SHA1 (secure hash standard) Chord assigns each node and key a m-bit identifier

Node (-> hashing IP Address)

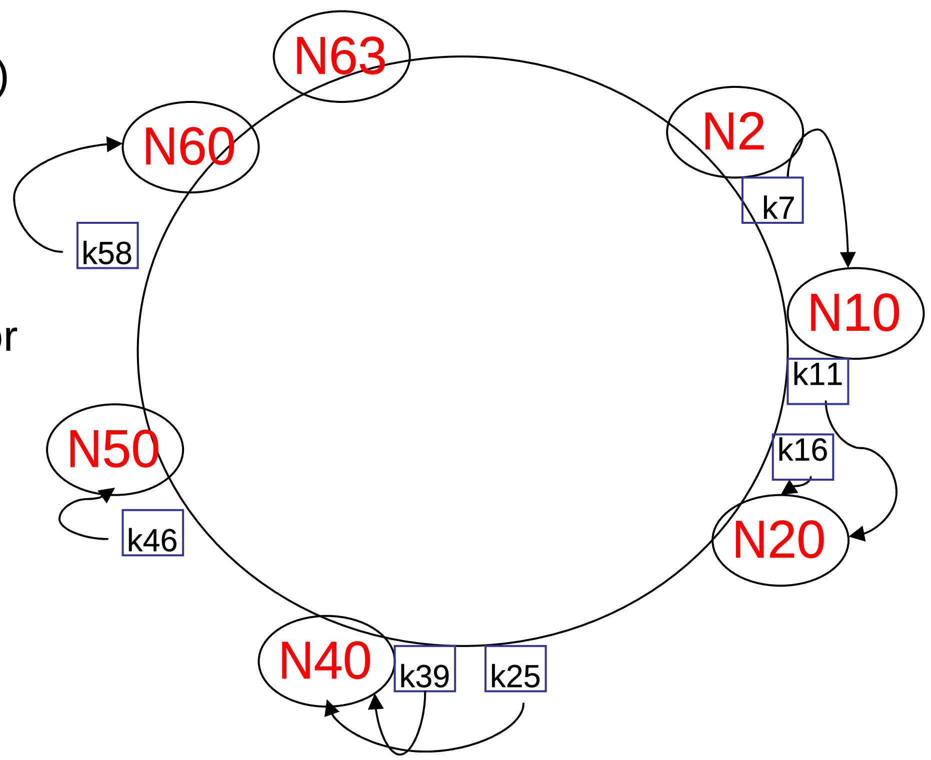
Keys(->hashing key itself)

Assign each key to its successor

ID Space: 0 to 2^m-1

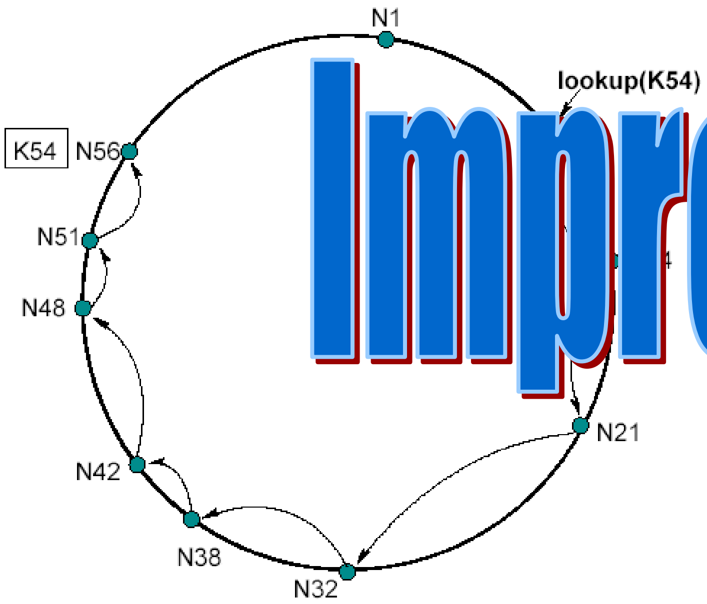
Here: *m = 6*

=> ID Range 0... 63



II. The Chord Protocol and how it works

A first simple (but slow!) lookup algorithm



Predecessor: pointer to the previous node on the id circle

Succe… or: …pointer … e succeeding node …

Improvable!!

… the …

- If *id* between *n* and it *successor* return *successor*
- Else forward query to *n*´s *successor* and so on

=>#messages linear in #nodes

- Protocol Improvements

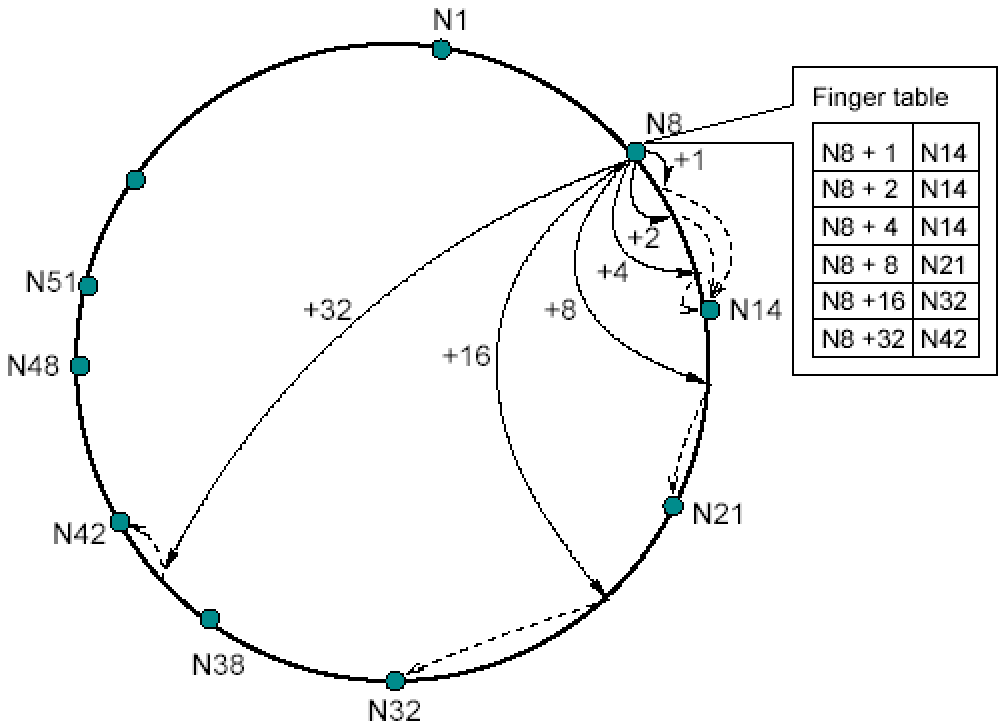
Introduce a routing table
m: number of bits in the key/node identifiers
n: ID of a node
i^{th} entry: the closest node that succeeds node *N* with ID *n* by at least $2^{i-1} \bmod 2^m$

⇒ Finger table could contain up to *m* entries

Each node has a successor list of its *r* successing nodes

II. The Chord Protocol and how it works

Finger Table for scalable node localisation: not enough information to find node directly but nodes in direct neighbourhood



- More efficient algorithm using finger tables

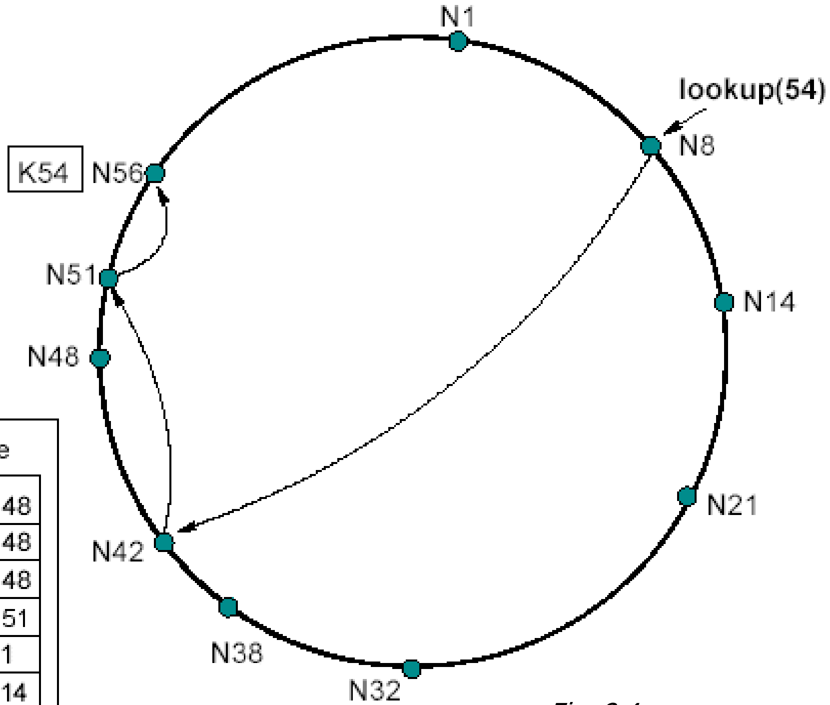
```
//ask node n to find the successor
//of id
n.find_successor(id)
 if (id in (n,successor])
  return successor
 else
  n' = closest_preceding_node(id);
  return n'.find_successor(id);
```

```
//search the local table for the
//highest predecessor of id
n.closest_preceding_node(id)
  for i=m downto 1 do
    if (finger[i] in (n,id))
      return finger[i];
  return n;
```

II. The Chord Protocol and how it works

Finger table

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



Finger table

N42+ 1	N48
N42+ 2	N48
N42+ 4	N48
N42+ 8	N51
N42+16	N1
N42+32	N14

Fig. 2.4

⇒#messages = log(#nodes) !!

II. The Chord Protocol and how it works

- Node joins and stabilization process

Lookups are expected to be correct (though the set of participating nodes can change!)

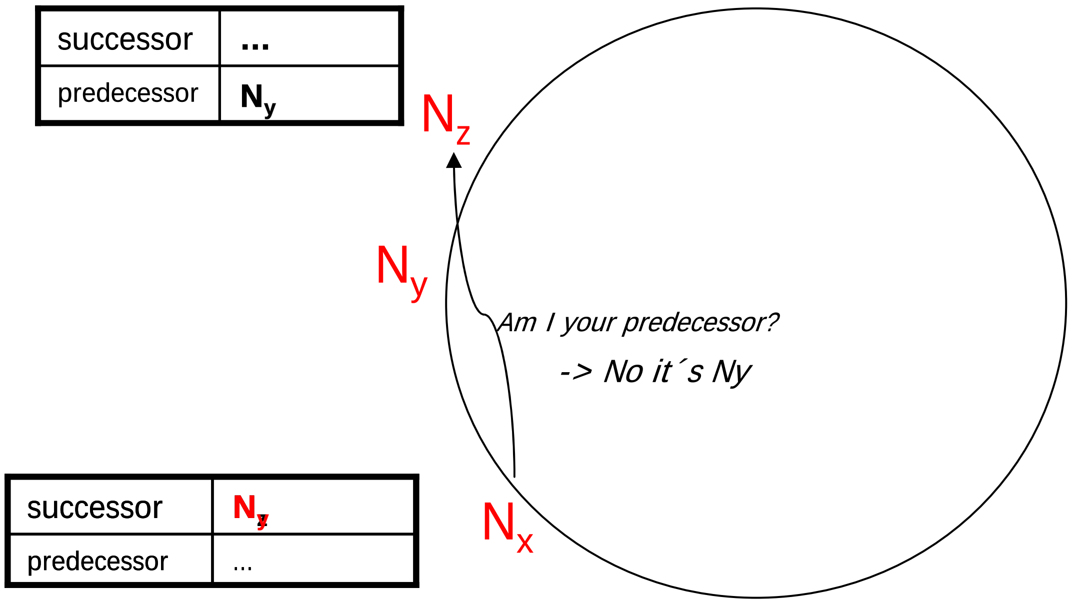
⇒ Important: Any node´s successor pointer is always up to date.

⇒ Stabilization protocol running periodically in the background and updating Chord´s finger tables and successor pointers

II. The Chord Protocol and how it works

- Stabilize()

  if *n* is not its *successor´* s *predecessor* it changes its *successor*



II. The Chord Protocol and how it works

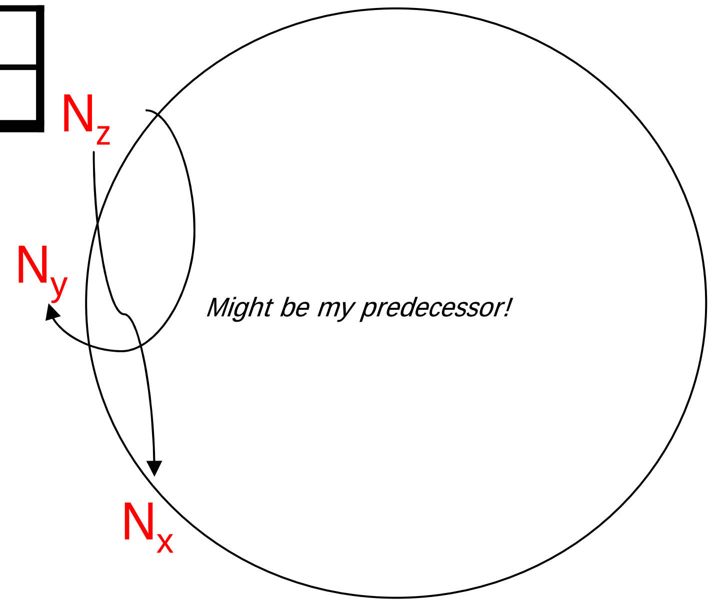
- Notify()

  N_z changes predecessor to N_x because predecessor = *nil*

  N_z changes predecessor to N_y because N_y between N_x and N_z.

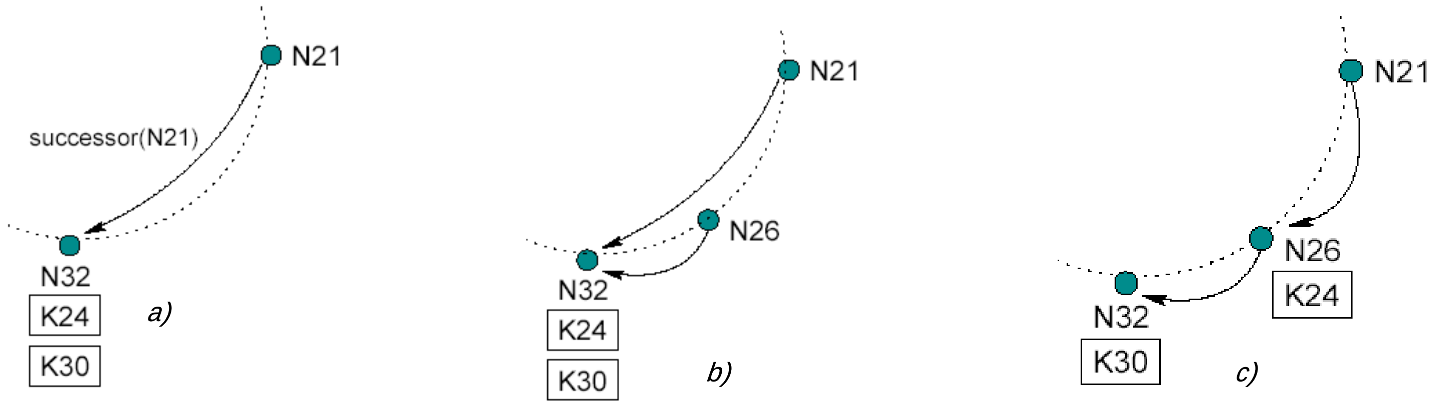
- Fix_fingers()
  updates finger tables

successor	...
predecessor	NIL N_x



II. The Chord Protocol and how it works

- Illustration (Joining node)



a) Initial state

b) node 26 enters the system & finds its *successor*

c) Stabilize procedure updates *successor* of *n21* to *n26*

II. The Chord Protocol and how it works

Node failures

- It is provable that even if half of the nodes in the network fail, lookups are executed correctly!

Overview

- I. Introduction
- II. The Chord Protocol and how it works
- III. Disadvantages and Improvements
- IV. Applications using Chord
- V. Summary

III. Disadvantages and Improvements

Why is Chord itself no search engine?

- Protocols are NO applications
- Upper layers have to coordinate purposes
- Chord only supports "exact match", cannot handle queries similar to one or more keys

Suitability for use in search engines

- Chord basically not suitable because specific information about desired data is needed to be able to compute the query key´s hashvalue

There exist several proposals for modifications to be able to find what is wanted without prior knowledge (meta data search extension)

III. Disadvantages and Improvements

Chord´s disadvantages

- asymmetric lookup => lookup from node n to node p could take a different number of hops than vice versa

=> In huge Chord Rings lookup to one of near preceding nodes routed clockwise over almost complete ring.

Approach for Improvement: **S-Chord** (->in-place notification of entry changes)

- Symmetric Structure:

  - routing entry symmetry – „if p points to n then n points to p“
  - routing cost symmetrie – lookup paths between to nodes very likely are equal though they are not the same! (=>no routing symmetry)
  - finger table symmetry

Overview

- **I. Introduction**
- **II. The Chord Protocol and how it works**
- **III. Disadvantages and Improvements**
- **IV. Applications using Chord**
- **V. Summary**

IV. Applications using Chord

DNS with Chord

- Host names hashed to keys, corresponding IP addresses are values
- Decentralized (no use for servers)
- Routing information and host changes can be updated dynamically

Cooperative mirroring

- Multiple providers of the same content
- Load is spread evenly over all hosts by mapping data blocks onto hosts

Overview

- I. Introduction
- II. The Chord Protocol and how it works
- III. Disadvantages and Improvements
- IV. Applications using Chord
- V. Summary

V. Summary

- Simple, scalable, provable correct and performant
- Any lookup requires O(log N) messages in a N node network
- Even in unstable systems lookups are correct
- Arbitrary implementation provided for developers
  => extendable in applications

Thank you for your attention!