Distributed Hash Table

- primjer Chord

Preuzeto i dorađeno iz

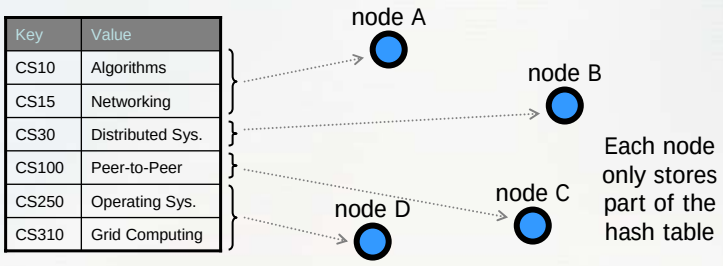
Self-Organization in Structured Overlay Networks (DKS) http://dks.sics.se

Seif Haridi, Ali Ghodsi, Luc Onana, Sameh El-Ansary, Per Brand, Supriya Krishnamurthy, Erik Aurell

1

What is a Distributed Hash Table (DHT)? 1/2

- **An infrastructure that enables the distribution of an ordinary *hash table* onto a set of cooperating nodes**



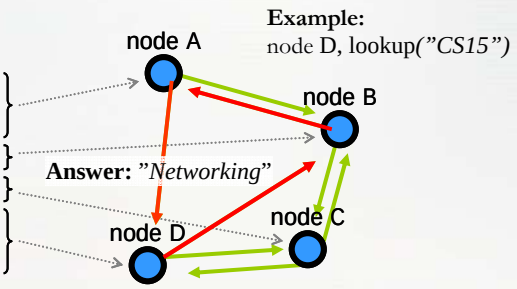
- **The DHT provides a basic *lookup service*, which allows *any* node to find the value associated with a given key**
- **Example:**
  lookup("*CS30*"), at any node should return: *"Distributed Sys."*

2

What is a Distributed Hash Table (DHT)? 2/2

- **To provide the lookup service, the nodes must be interconnected**

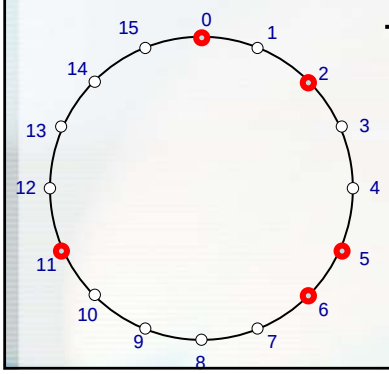
Key	Value
CS10	Algorithms
CS15	Networking
CS30	Distributed Sys.
CS100	Peer-to-Peer
CS250	Operating Sys.
CS310	Grid Computing



- **Each node maintains a *routing table* with pointers to some other nodes such that lookup requests can be routed to the node storing the requested key/value-pair (a.k.a. item)**

How do we distribute the hash table?

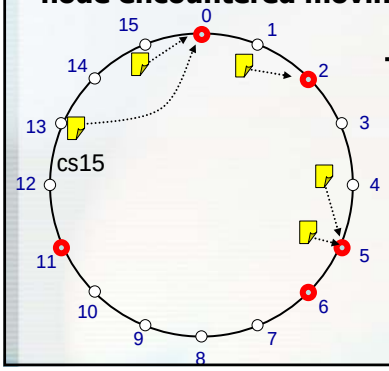
- **Use a logical name space called the *identifier space*, consisting of identifiers {**0,1,2,…, N-1**}**
- **Identifiers can be binary encoded, key length equals** m
- **The identifier space can be perceived as a logical ring modulo** N
- **Every node is assigned an identifier using a function** H_1**.**
- **Note there are not necessarily** N **nodes in a network!**



- Example: N=16 (m=4), nodes {**a,b,c,d,e**}
- Node **a** gets identifier 0 since H_1(**a**)=0, the other nodes **b, c, d, e,** get identifiers 2, 5, 6, 11 the same way (they are marked red in the figure)

How are items assigned to nodes?

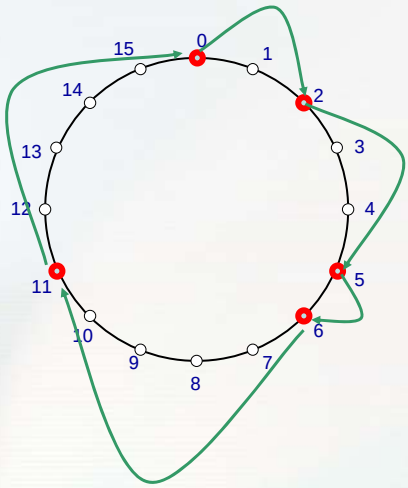
- **Items are mapped to the same identifier space using another function H_2, every node knows H_2**
- **Items are stored at nodes with the same identifier (if possible)**
- **As items may be mapped to identifiers of non-existing nodes, such items are stored at their *successor nodes*, i.e. the first node encountered moving in the clockwise direction**



- Item (”**cs15**”, ”**networking**”) is mapped to identifier 13 since H_2 (”**cs15**”) = 13, other items are similarly mapped to 15, 2, 4, 5
- As there is no node with id=13 in the network, the item is stored at the successor node (node 0 in this case)

How do we interconnect the nodes?

- **Each node maintains a *routing pointer* to the successor in the ring (single routing pointer)**



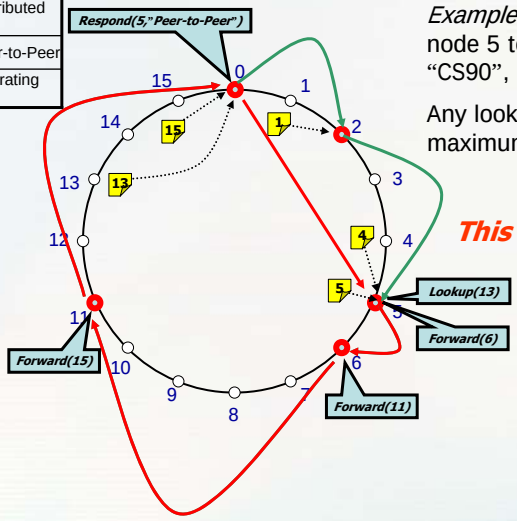
- **The successor of a node n is the first node met going in clockwise direction starting at $n+1$**

Successor of Node 0 → Node 2
Successor of Node 2 → Node 5
Successor of Node 5 → Node 6
Successor of Node 6 → Node 11
Successor of Node 11 → Node 0

Simple lookup example

Key	Value
1 (CS10)	Algorithms
4 (CS15)	Networking
5 (CS30)	Distributed S.
13 (CS90)	Peer-to-Peer
15 (CS95)	Operating Sys.

- **Lookups can be resolved by following the routing pointers sequentially**



Example: A lookup is made at node 5 to get the value of key "CS90", identifier H_2("CS90")=13

Any lookup can be resolved in maximum O(*N*) hops.

This solution is too slow!

Speeding up lookups (1)

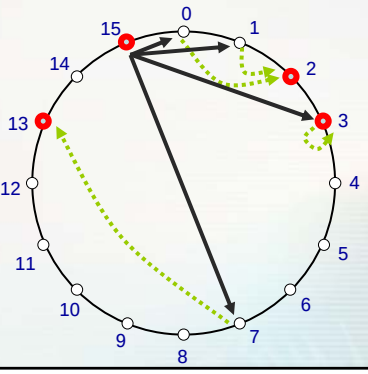
- Each node, n, not only points to its successor but to the successors of the following nodes

$n+2^1, n+2^2, n+2^3,\ldots, n+2^L$

Example: routing table for node n=15

Successor of $15+2^0=0$ → Node 2
Successor of $15+2^1=1$ → Node 2
Successor of $15+2^2=3$ → Node 3
Successor of $15+2^3=7$ → Node 13

Start of the intervals! As these nodes do not exist, node 15 points to their successors



Speeding up lookups (2)

- At each step during search if node cannot find the item, route to the node in your routing table with the identifier less then the item identifier
- At each step during routing, the distance between the currrent node and destination is halved (in the worst case), requires $O(\log_2 N)$ hops at worst

Example: search for item 10 starting at node 15
(item 10 is stored at 13)

Node 15 routes search to node 3 (node 13 is larger than 10!)

Node 3 routes the request to node 13