Module BSI: Big-Data Storage and Processing Infrastructures

Gabriel Antoniu, *KERDATA* & Davide Frey, *WIDE*
INRIA

Administrivia

- Gabriel Antoniu, DR INRIA, KERDATA Team

  gabriel.antoniu@inria.fr

- Davide Frey, CR INRIA, WIDE Team

  davide.frey@inria.fr

- Course Website: http://bsi-sif.irisa.fr/

Schedule

Date	Room	Instructor	Topic and Slides
14 September 2020 5pm – 7pm	B02B-E110	D. Frey	From P2P to the Edge Basic Technologies: DHTs
16 September 2020 5pm – 7pm	B02B-E110	G. Antoniu	Introduction to parallel and distributed infrastructures: supercomputers, clusters, grid, clouds
21 September 2020 5pm – 7pm	B02B-E110	D. Frey	Basic Technologies: Gossip Application: Multicast
23 September 2020 5pm – 7pm	B02B-E110	G. Antoniu	Introduction to Big Data: applications, challenges
28 Septembre 2020 5pm – 7pm	B02B-E110	D. Frey	Video Streaming
30 September 2020 5pm – 7pm	B02B-E110	G. Antoniu	Programming Models and Environments for Data Analytics: MapReduce, Hadoop
5 October 2020 5pm – 7pm	B02B-E110	D. Frey	Big-Data Applications: Key Value Stores Big Data Applications: Private Recommendation Systems
7 October 2020 5pm – 7pm	B02B-E110	G. Antoniu	Applications and limitations of Map-Reduce.
12 October 2020 5pm – 7pm	B02B-E208	D. Frey	Big-Data Applications: Decentralized Machine Learning Blockchain
14 October 2020 5pm – 7pm	B02B-E110	G. Antoniu	Post-Hadoop Data Processing Approaches: Spark, Flink and What Comes After
19 October 2020 5pm – 7pm	B02B-E110	Written Exam	
21 October 2020 5pm – 7pm	B02B-E110	Student Presentations	

Exam

- Select research paper

- Assignment/Written Exam:
  - Report/Questions on the paper
  - Questions on course

- Oral Exam: give a talk

Half the Course in one slide

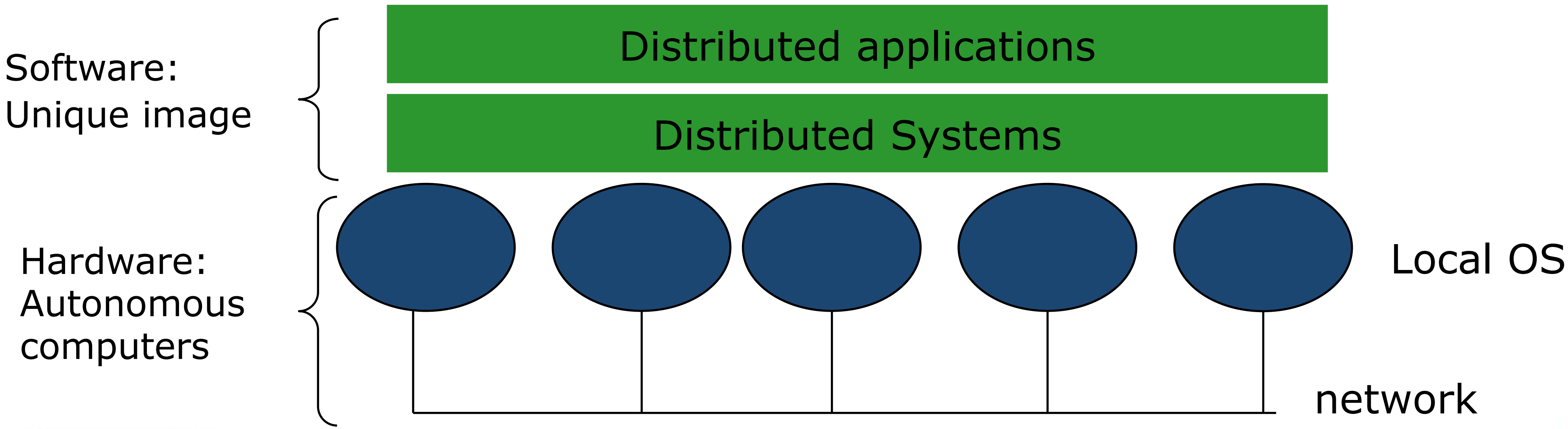
- Historical Perspective:
  - Peer-to-Peer -> Grid -> Cloud-> Edge
- Basic Technologies
  - Distributed Hash Tables
  - Gossip Protocols
- Non-Big-Data Applications
  - Multicast
  - Video Streaming
- Big-Data Applications
  - Key Value Stores
  - Recommendation Systems
  - Private Machine Learning
- Blockchain

IBD
From Peer-to-Peer to the Edge

informatics mathematics

Distributed System

- Definition [Tan95] « *A collection of independent computers that appears to its users as a single coherent system* »



Why should we decentralize/parallelize ?

- Economical reasons
- Performance
- Availability
- Resource aggregation
- Flexibility (load balancing)
- Privacy

Growing need of working collaboratively, sharing and aggregating distributed (geographically) distributed.

How to decentralize ?

Tools (some)

- Client-server Model
- Peer-to-peer Model
- Grid Computing
- Cloud

Goals

- Scalability
- Reliability
- Availability
- Security/Privacy

The Peer-To-Peer Model

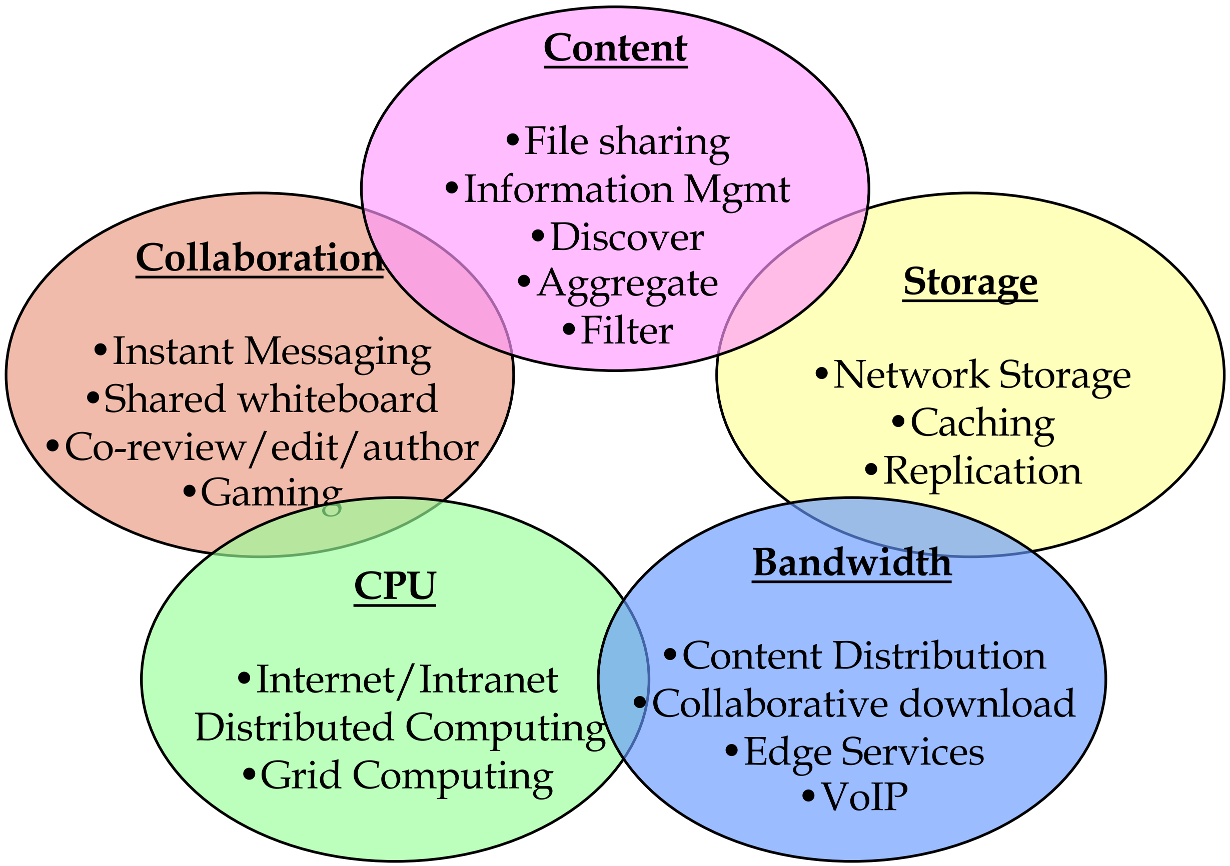
- Name one peer-to-peer technology

Peer-to-Peer Systems

The Peer-to-Peer Model

- End-nodes become active components!
  – previously they were just clients
- Nodes participate, interact, contribute to the services they use.
- Harness huge pools of resources accumulated in millions of end-nodes.

P2P Application Areas



Grid Computing

- Collection of computing resources on LAN or WAN
- Appears as large virtual compuiting system
- Focus on high-computational capacity
- May use some P2P technologies but can exploit "central" components

Cloud Computing

- Provide Service-level computing
- On-demand instances
- On-demand provisioning
- May exploit
  - Grids
  - Clusters
  - Virtualization

Edge Computing

- Ever used Dropbox to send a file to your deskmate?
- Augment Cloud with edge of the network
  - Bring back P2P ideas into the cloud model

Back to P2P

- Use resources at the edge of the Internet
  - Storage
  - CPU cycles
  - Bandwidth
  - Content
- Collectively produce services
  - Nodes share both benefits **and** duties
- Irregularities and dynamics become the norm

Essential for Large Scale Distributed System

CPU Resources

- Cool example in the news
  - even though not strictly P2P

https://phys.org/news/2019-09-sum-cubes-solvedusing-real-life.html

Main Advantages of P2P

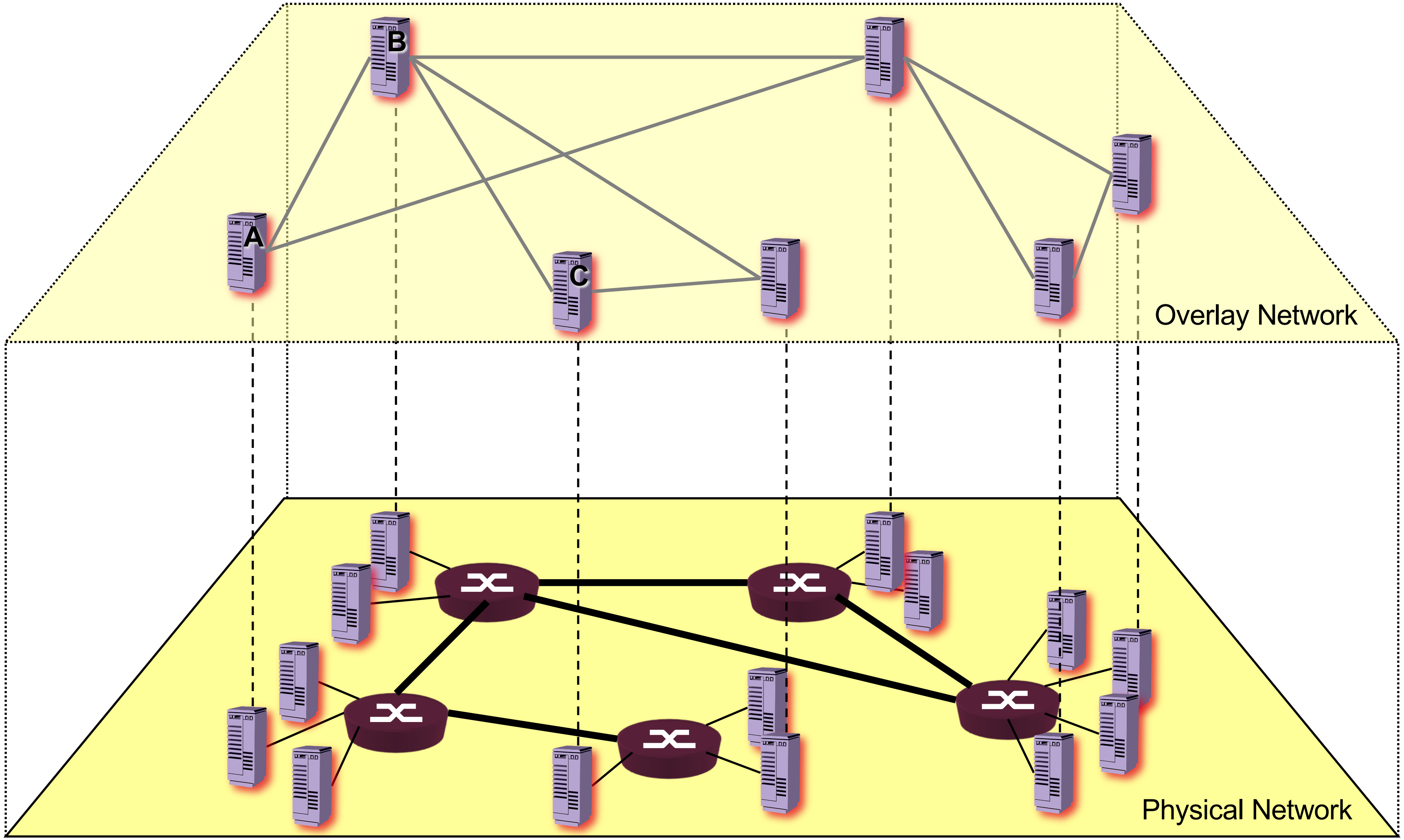
- Scalable
  - higher demand → higher contribution!
  - Increased (massive) aggregate capacity
  - Utilize otherwise wasted resources

- Fault Tolerant
  - No single point of control
  - Replication makes it possible to withstand failures
  - Inherently handle dynamic conditions

Main Challenges in P2P

- Fairness and Load Balancing
- Dynamics and Adaptability
- Fault-Tolerance: Continuous Maintenance
- Self-Organization

Key Concept: Overlay Network

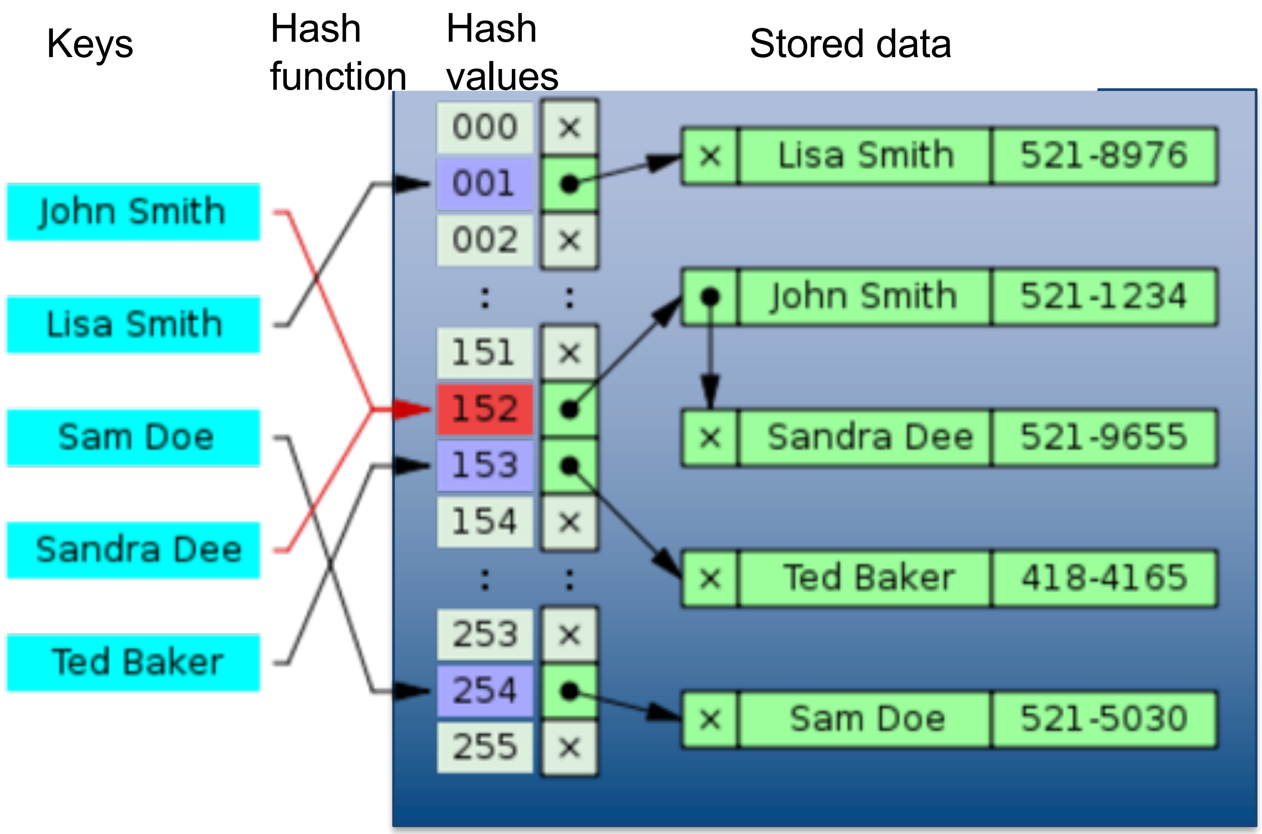


Overlay types

Unstructured P2P	Structured P2P
○ Any two nodes can establish a link ○ Topology evolves at random ○ Topology reflects desired properties of linked nodes	○ Topology strictly determined by node IDs

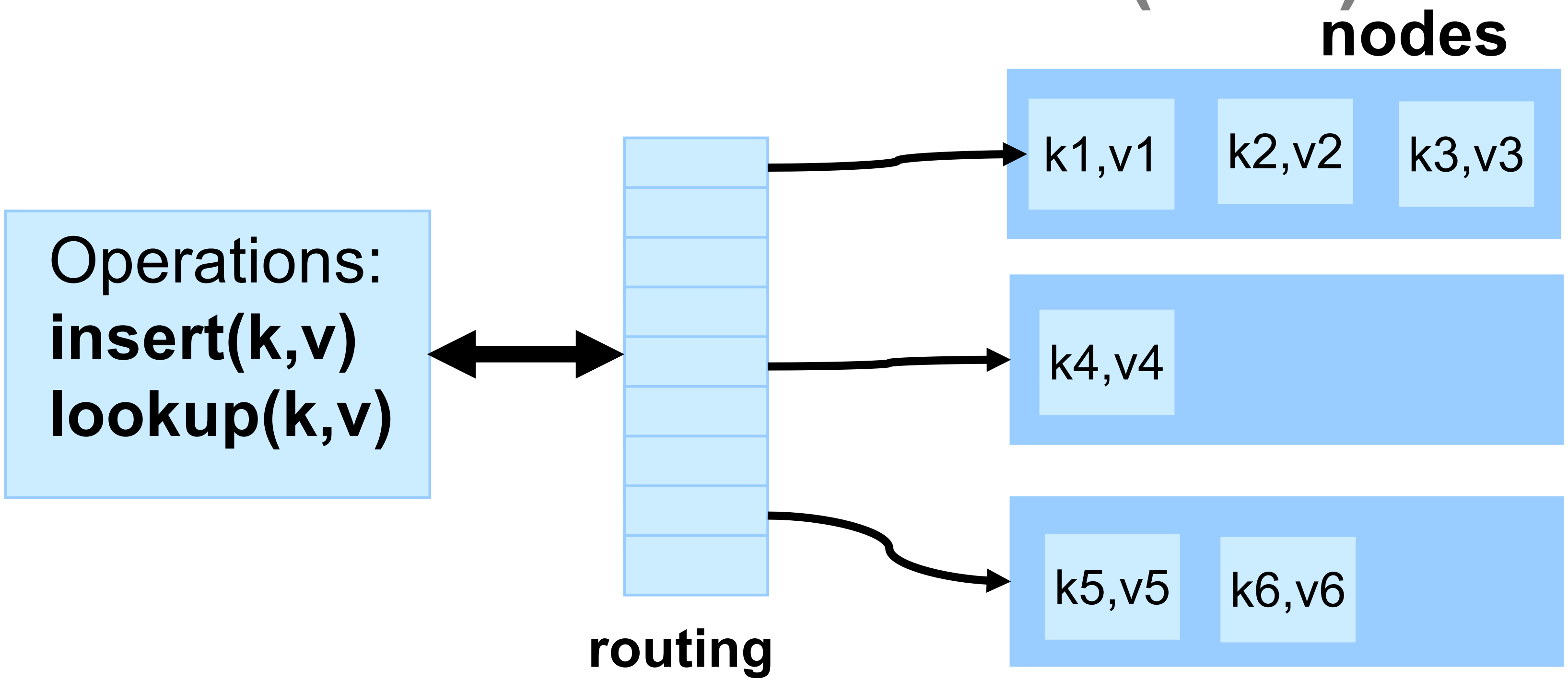
IBD
Distributed Hash Tables

Hash Table



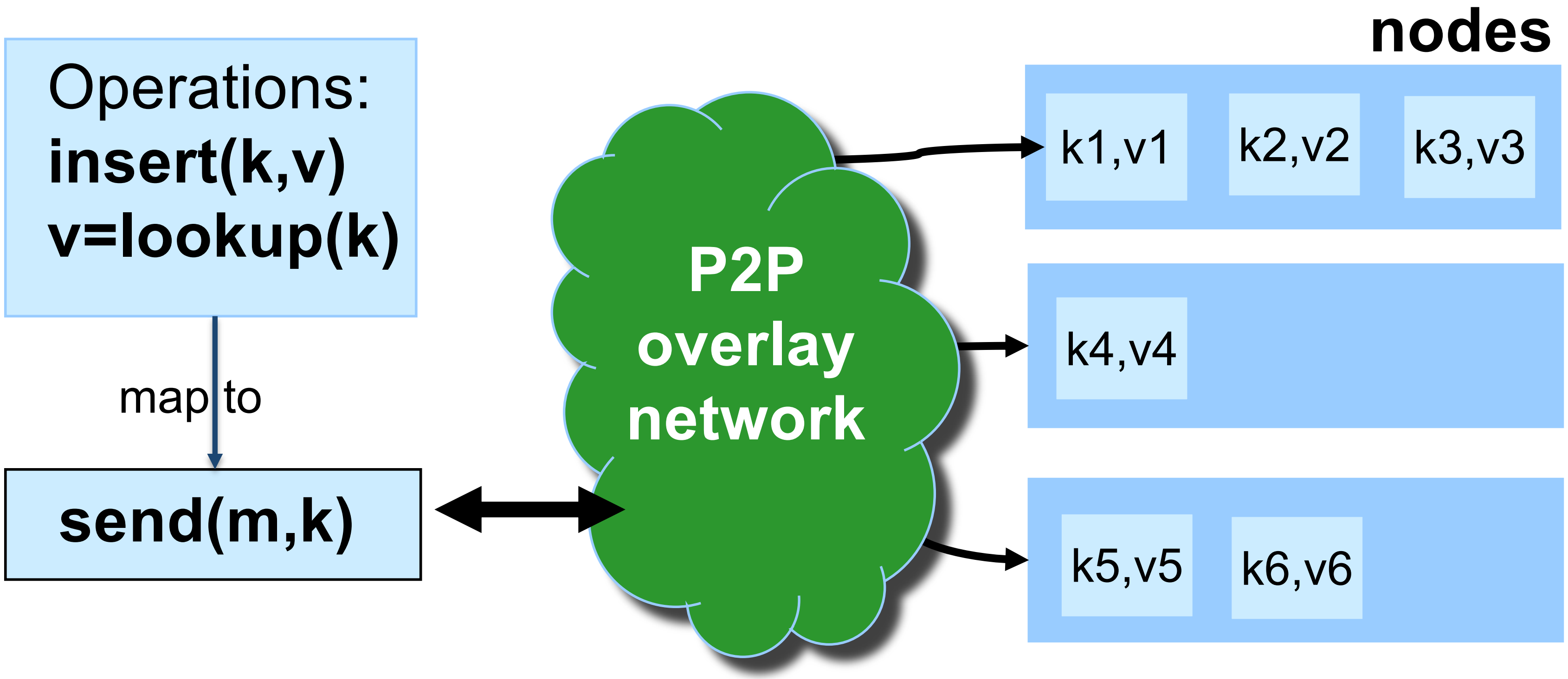
Efficient information lookup

Distributed Hash Table (DHT)



- Store <key,value> pairs
- Efficient access to a value given a key
- Must route hash keys to nodes.

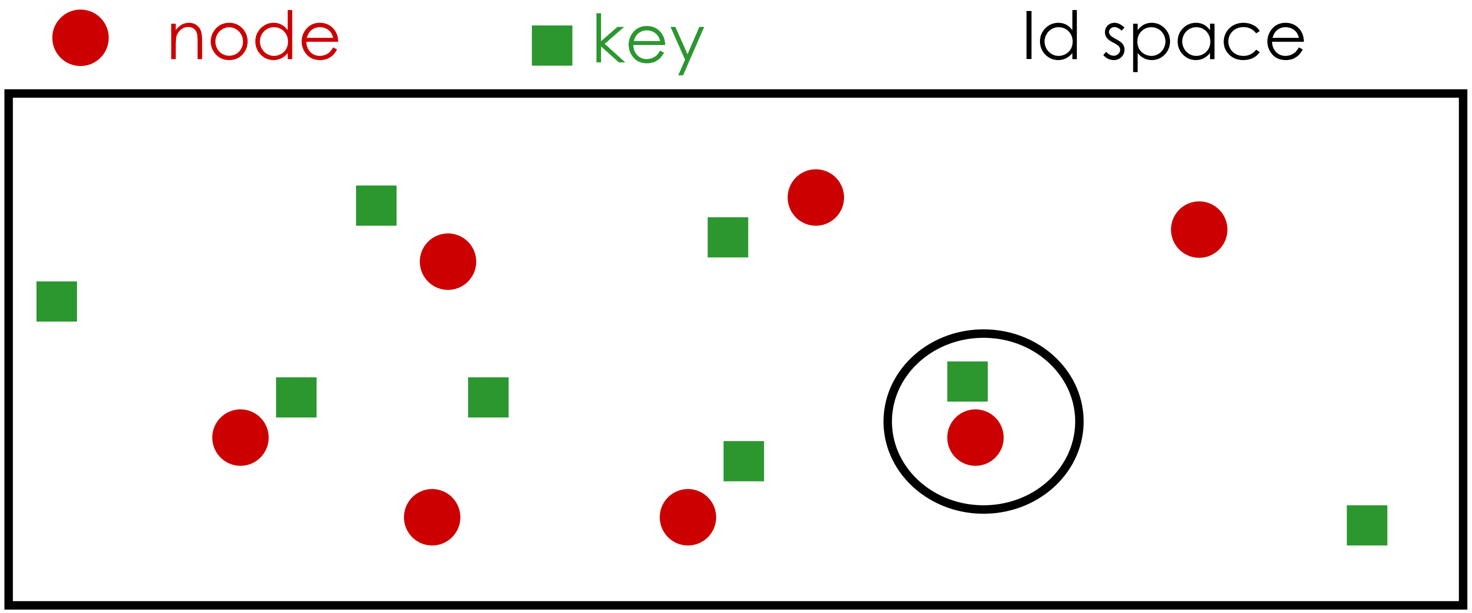
Distributed Hash Table



- Insert and Lookup send messages keys
- P2P Overlay defines mapping between keys and physical nodes
- Decentralized routing implements this mapping

DHT Examples

Pastry (MSR/RICE)



NodeId = 128 bits
Nodes and key place in a linear space (ring)
Mapping : a key is associated to the node with the numerically closest nodeId to the key

Pastry (MSR/Rice)

Naming space :

- Ring of 128 bit integers
- *nodeIds* chosen at random
- Identifiers are a set of digits in base 16

Key/node mapping

- key associated with the node with the numerically closest node id

Routing table:

- Matrix of 128/4 lines et 16 columns
- routeTable(i,j):

  nodeId matching the current node identifier up to level I
  with the next digit is j

Leaf set

- 8 or 16 closest numerical neighbors in the naming space

Proximity Metric

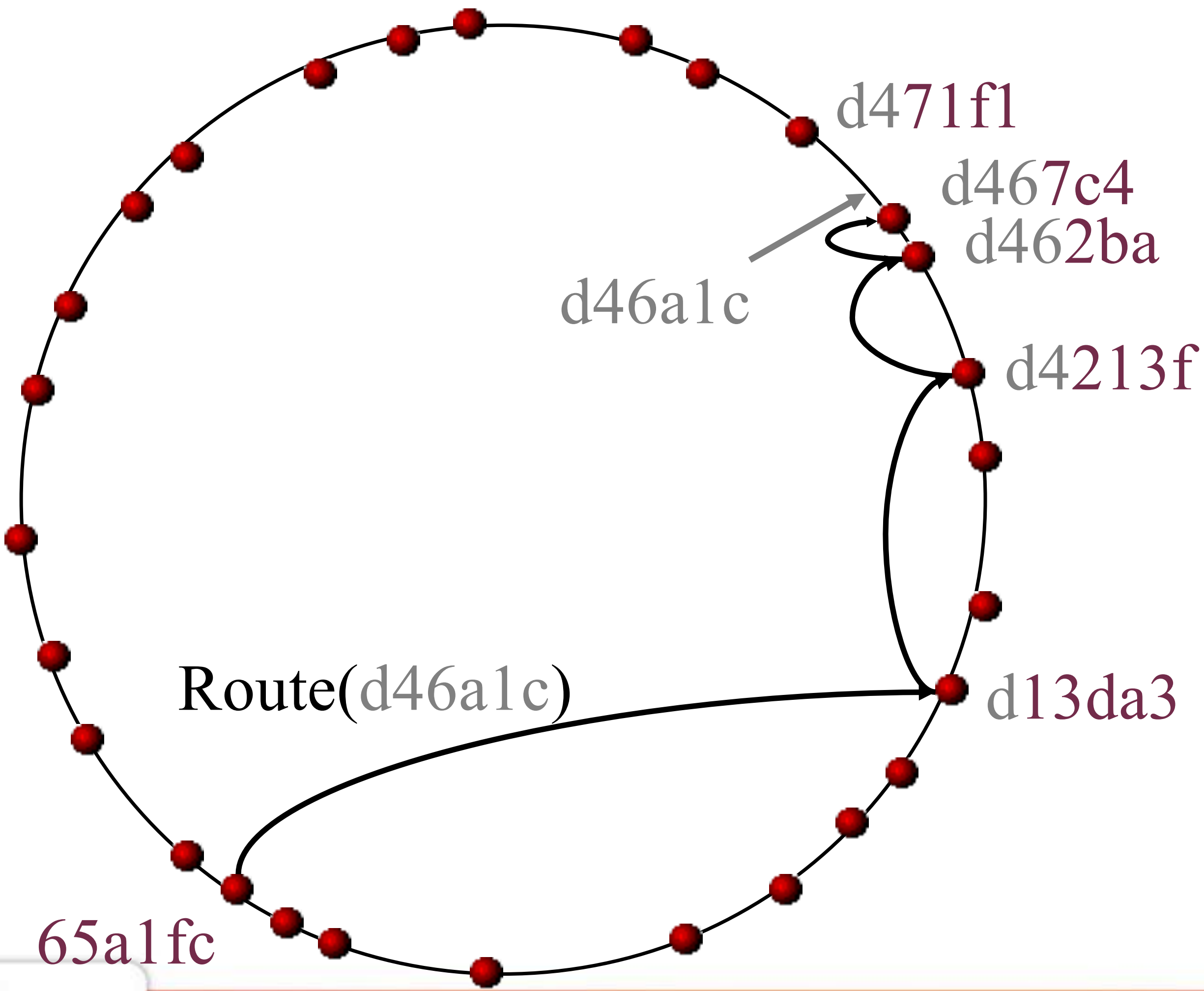
- *Bias selection of nodes*

Pastry: Routing table(#65a1fcx)

Line 0	0x	1x	2x	3x	4x	5x		7x	8x	9x	ax	bx	cx	dx	ex	fx
Line 1	60x	61x	62x	63x	64x		66x	67x	68x	69x	6ax	6bx	6cx	6dx	6ex	6fx
Line 2	650x	651x	652x	653x	654x	655x	656x	657x	658x	659x		65bx	65cx	65dx	65ex	65fx
Line 3	65a0x		65a2x	65a3x	65a4x	65a5x	65a6x	65a7x	65a8x	65a9x	65aax	65abx	65acx	65adx	65aex	65afx

$\log_{16}$ N liges

Pastry: Routing



Properties

- $\log_{16} N$ hops
- Size of the state maintained (routing table): $O(\log N)$

Routing algorithm (on node A)

if $(L_{-\lfloor |L|/2 \rfloor} \leq D \leq L_{\lfloor |L|/2 \rfloor})$ {
    // D is within range of our *leaf set*
    forward to L_i, s.th. $|D - L_i|$ is minimal;
} else {
    // use the routing table
    Let $l = shl(D, A)$;
    if $(R_l^{D_l} \neq null)$ {
        forward to $R_l^{D_l}$;
    }
    else {
        // rare case
        forward to $T \in L \cup R \cup M$, s.th.
            $shl(T, D) \geq l$,
            $|T - D| < |A - D|$
    }
}

R_l^i : entry of the routing table R, $0 \leq i \leq 2^b$, line l, $0 \leq l \leq \lfloor 128/b \rfloor$

L_i : ith closest nodeId in the leafset

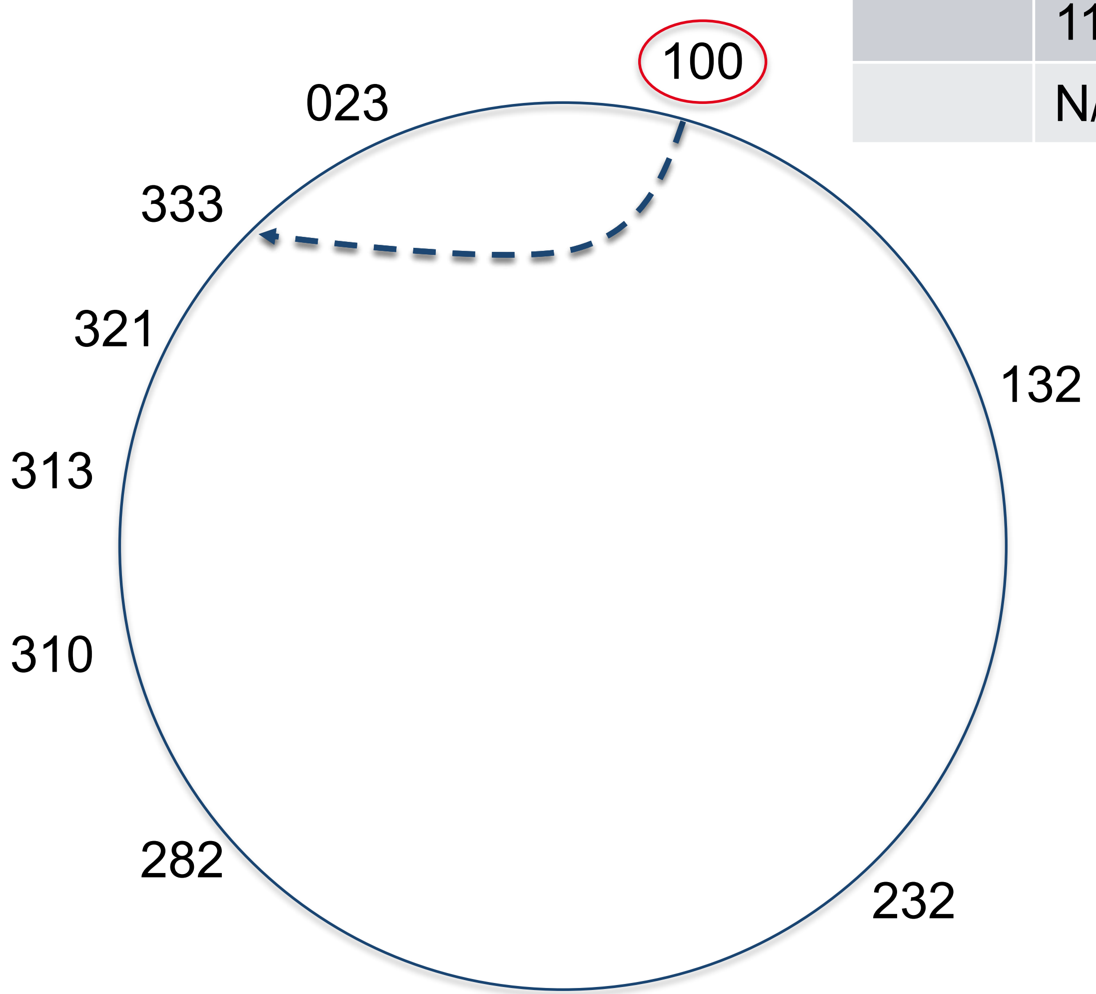
D_l : value of the l digits of key D

$SHL(A, B)$: length of the shared prefix between A and B

Pastry Example

b=4

Route to 311

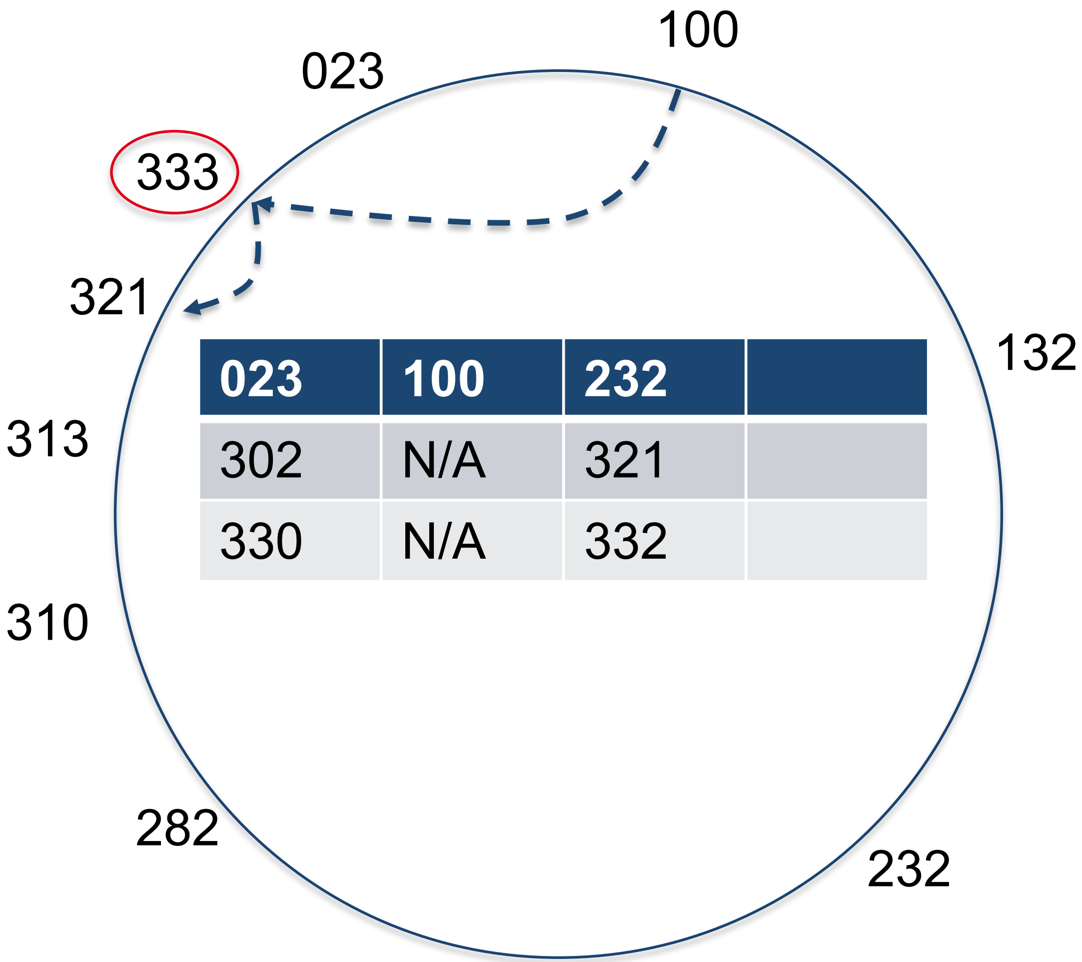


023		232	333
	113	122	132
	N/A	102	103

Pastry Example

b=4

Route to 311

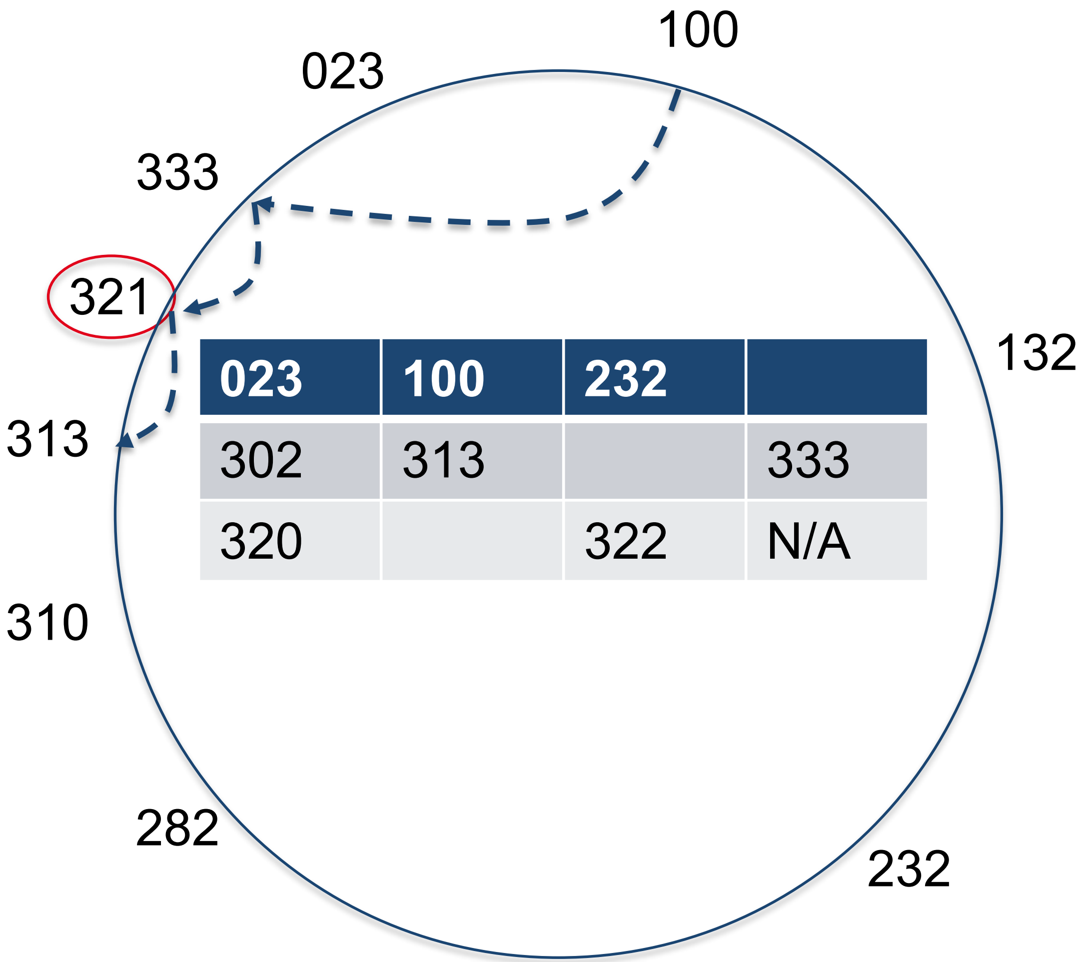


023	100	232	
302	N/A	321	
330	N/A	332	

Pastry Example

b=4

Route to 311

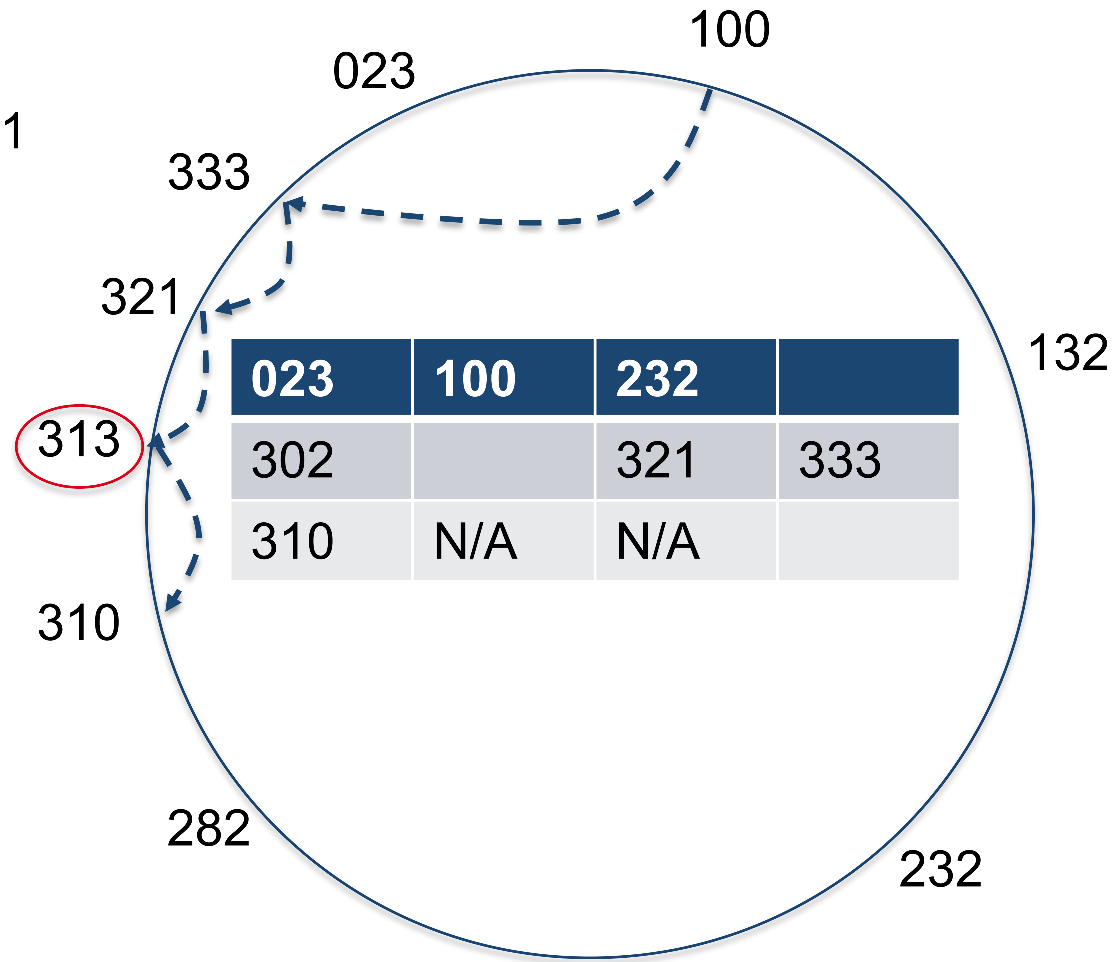


023	100	232	
302	313		333
320		322	N/A

Pastry Example

b=4

Route to 311

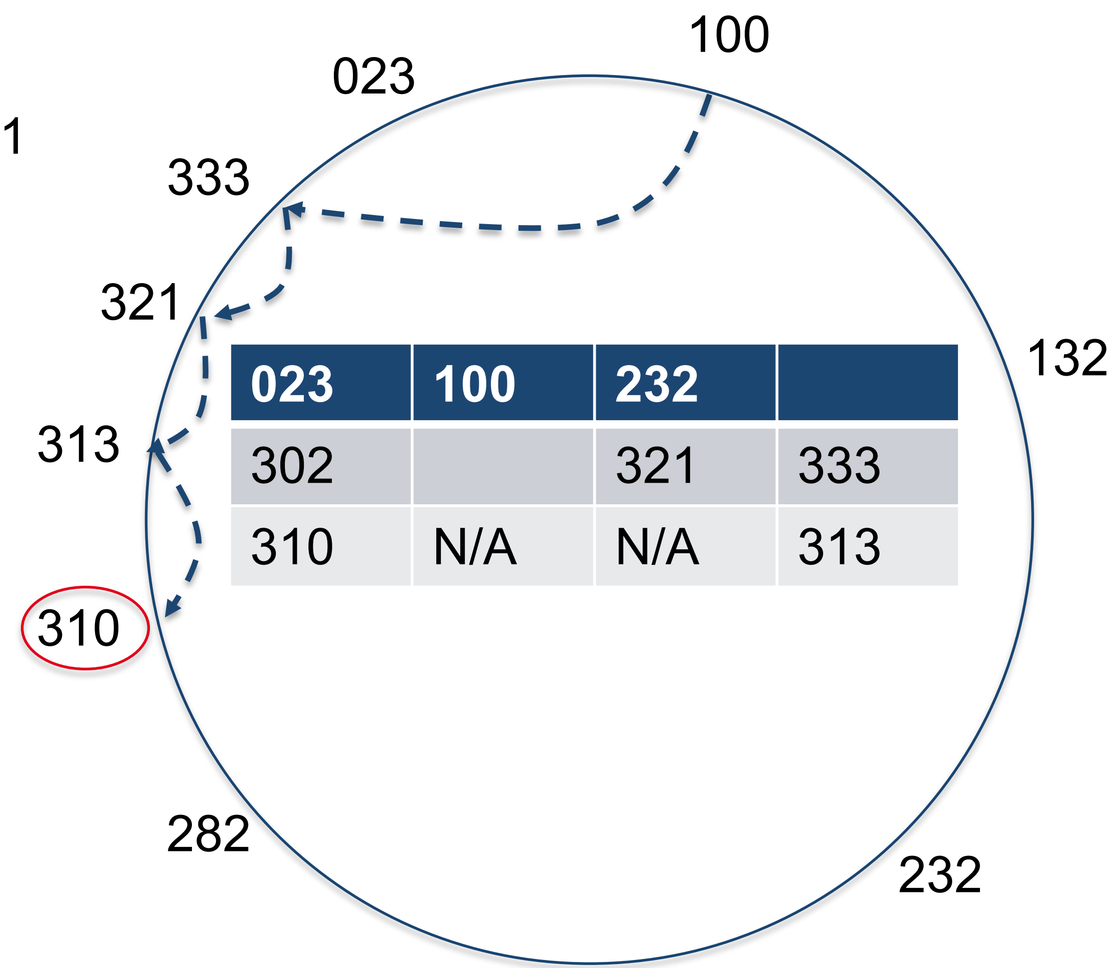


023	100	232	
302		321	333
310	N/A	N/A	

Pastry Example

b=4

Route to 311



023	100	232	
302		321	333
310	N/A	N/A	313

Node departure

Explicit departure or failure

Replacement of a node

The leafset of the closest node in the leafset contains the closest new node, not yet in the leafset

Update from the leafset information

Update the application

Failure detection

Detected when immediate neighbours in the name space (leafset) can no longer communicate

Detected when a contact fails during the routing

- Routing uses an alternative route

Fixing the routing table of A

- Repair

R_l^d : entry of the routing table of A to repair

A contacts another entry (at random) R_l^i from the same line so that $(i \neq d)$ and asks for entry R_l^d, otherwise another entry from R_{l+1}^i $(i \neq d)$ if no node in line l answers the request.

State maintenance

Leaf set

- is aggressively monitored and fixed

Routing table

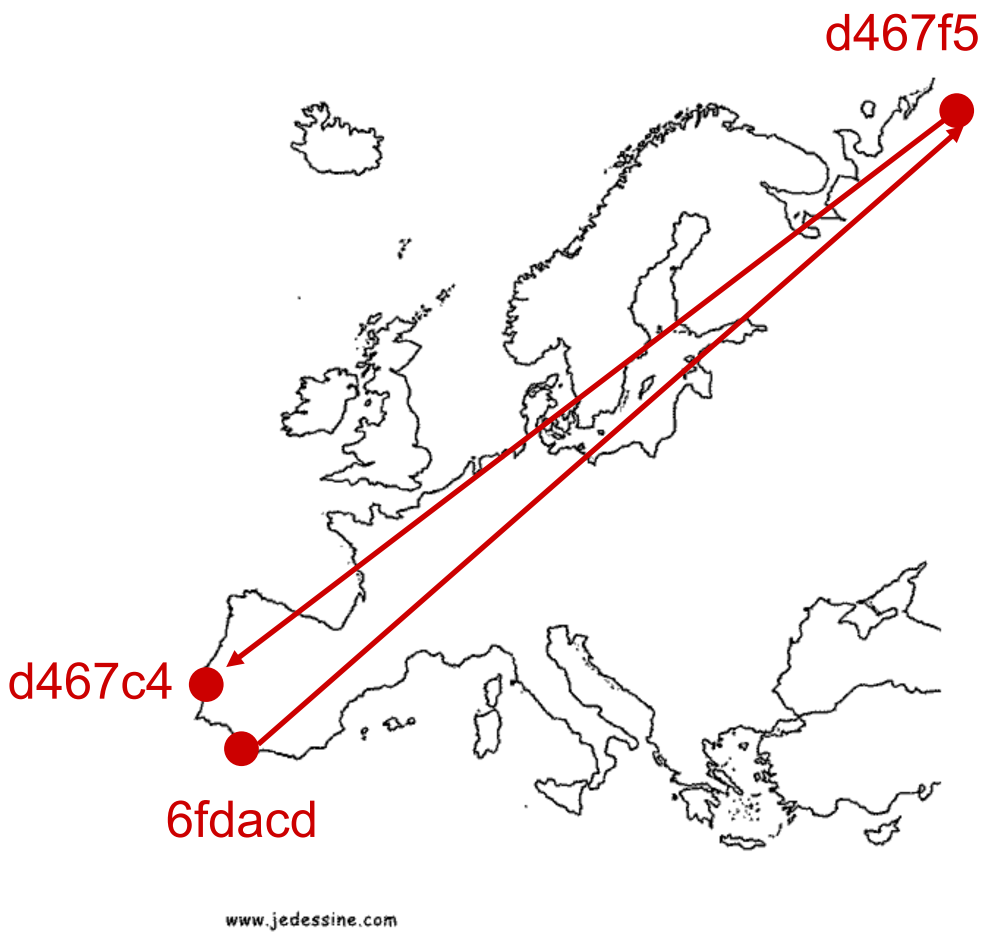
- are lazily repaired

  When a hole is detected during the routing

- Periodic gossip-based maintenance

Reducing latency

- **Random assignment of nodeId**: Nodes numerically close are geographically (topologically) distant
- **Objective**: fill the routing table with nodes so that routing hops are as short (latency wise) as possible



Exploiting locality in Pastry

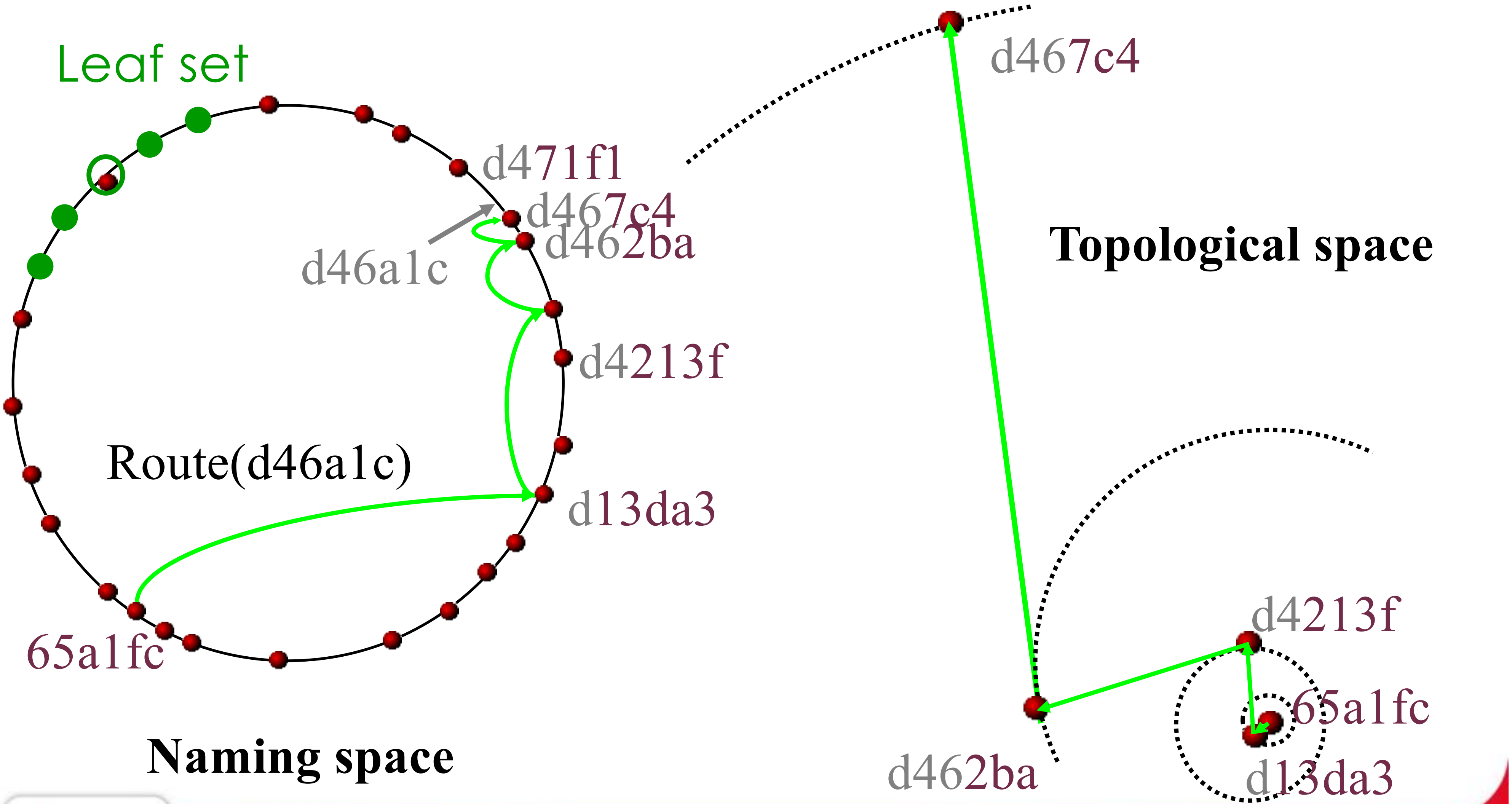
Neighbour selected based of a network proximity metric:

- Closest topological node
- Satisfying the constraints of the routing table

routeTable(i,j):

- *nodeId* corresponding to the current *nodeId* up to level i
  next digit = j
- nodes are close at the top level of the routing table
- Farther nodes at the bottom levels of the routing tables

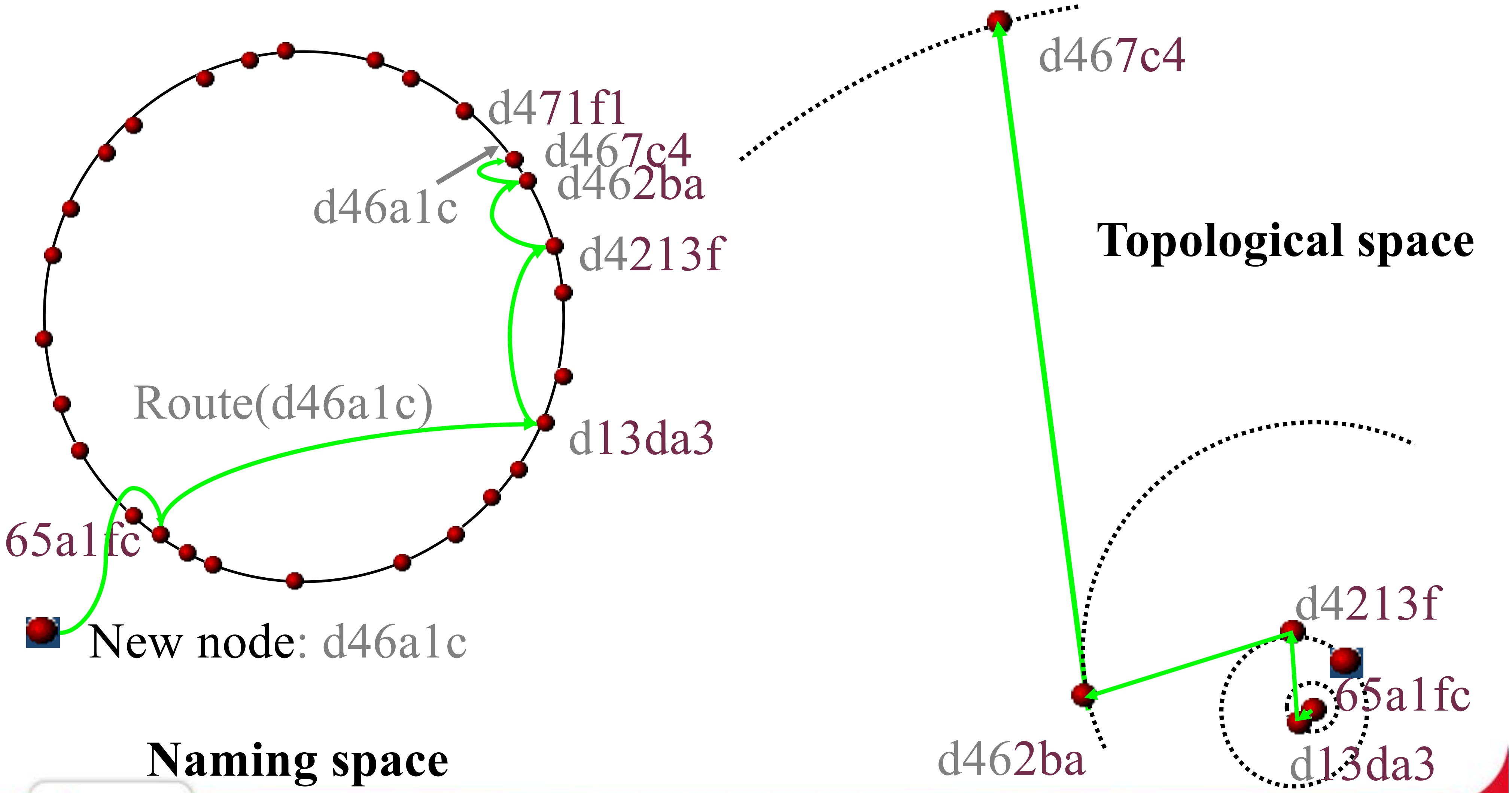
Proximity routing in Pastry



Locality

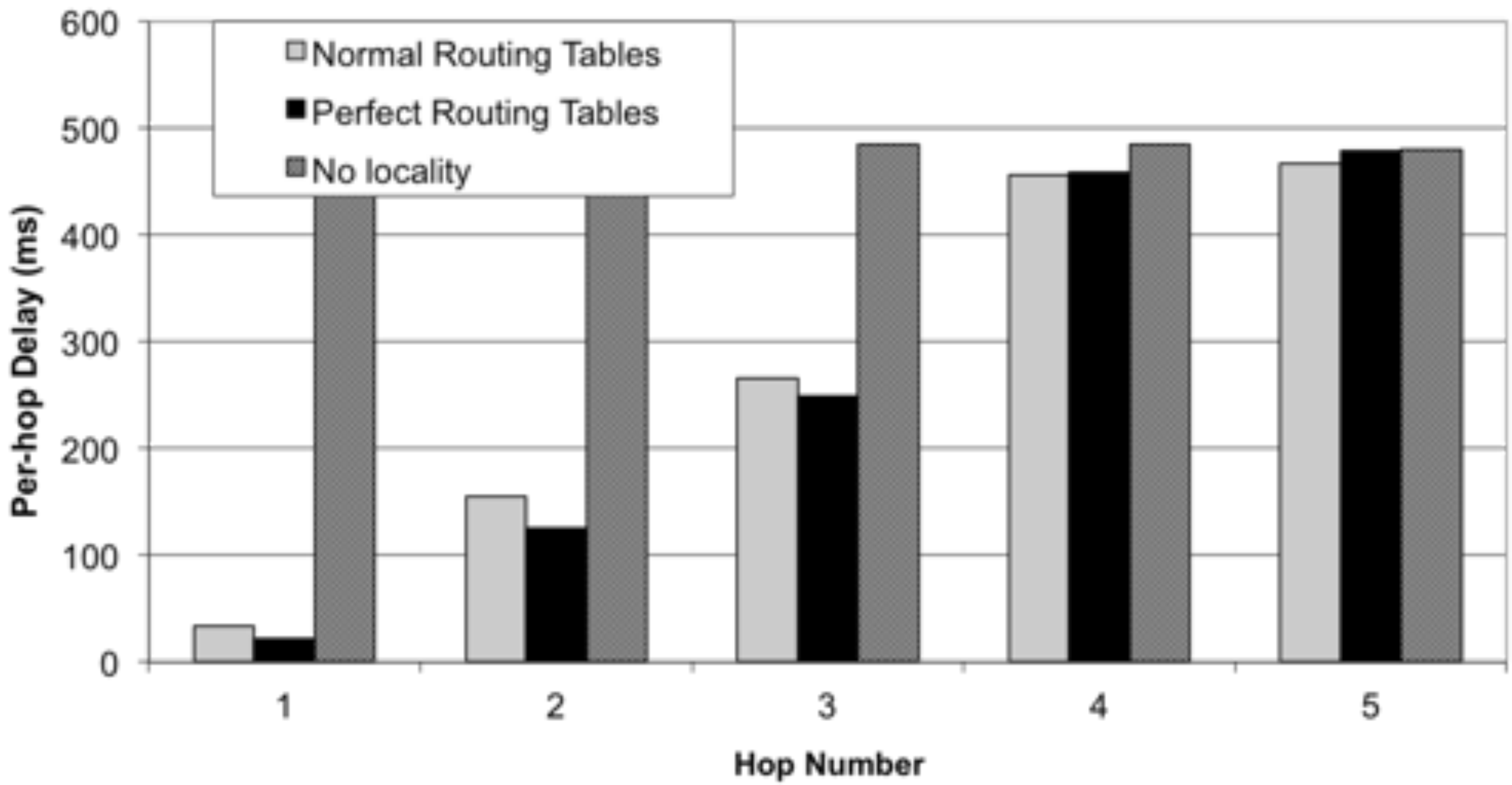
1. Joining node X routes asks A to route to X
   - Path A,B,… -> Z
   - Z numerically closest to X
   - X initializes line i of its routing table with the contents of line i of the routing table of the *ith* node encountered on the path
2. Improving the quality of the routing table
   - X asks to each node of its routing table its own routing state and compare distances
   - Gossip-based update for each line (20mn)
     - Periodically, an entry is chosen at random in the routing table
     - Corresponding line of this entry sent
     - Evaluation of potential candidates
     - Replacement of better candidates
     - New nodes gradually integrated

Node insertion in Pastry



Performance

1.59 slower than IP on average



References

- Rowstron and P. Druschel, "**Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems**", *Middleware'2001*, Germany, November 2001.

Content Adressable Network -CAN

UCB/ACIRI

Virtual-coordinate Cartesian space

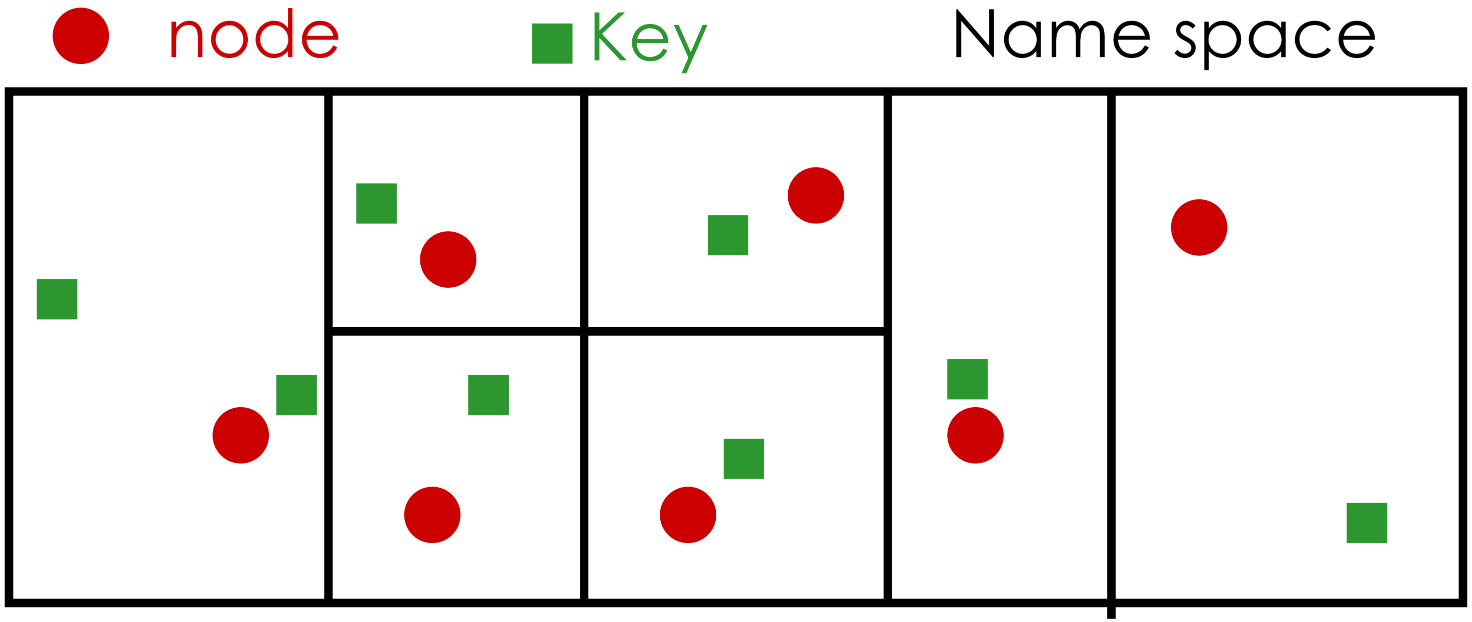
Space shared among peers

- Each node is responsible for a part (zone)

Abstraction

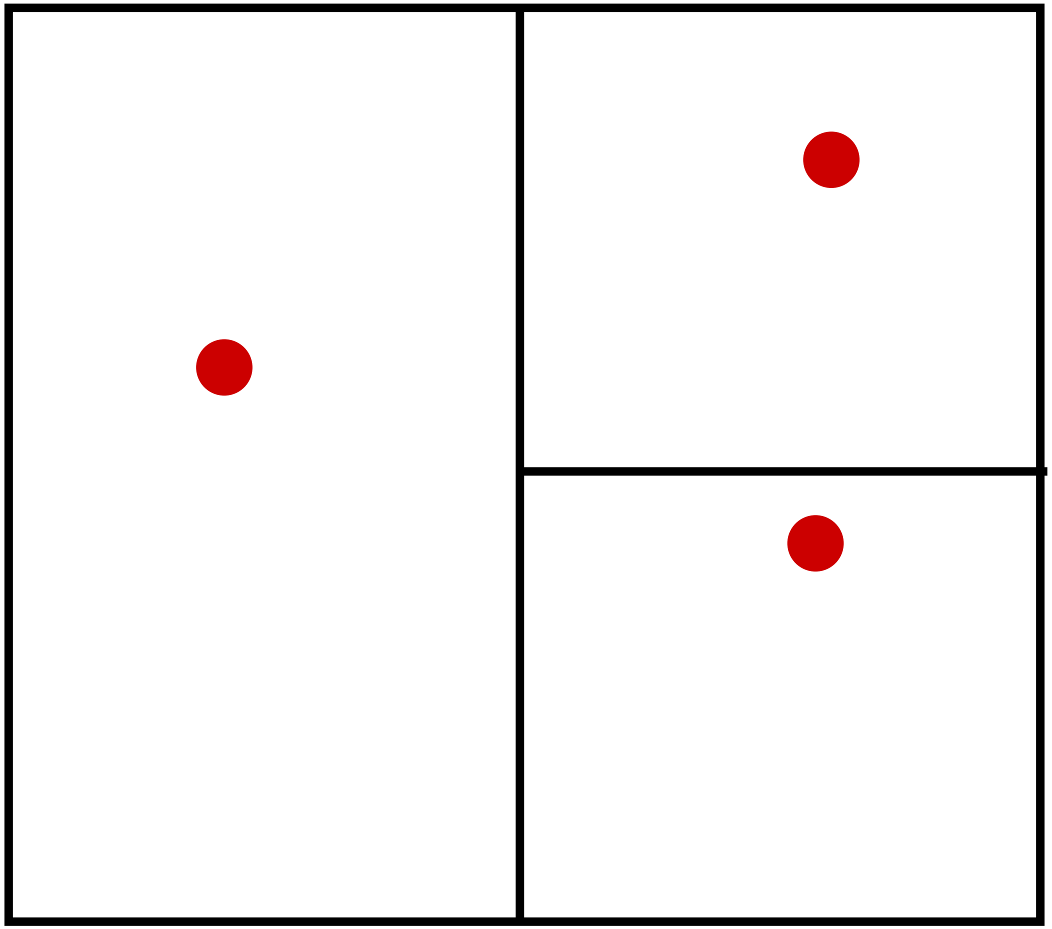
- CAN stores data at specific points in the space
- CAN routes information from one point of the space to another (DHT functionality)
- A point is associated with the node that owns the zone in which the point lies

Space organisation in CAN

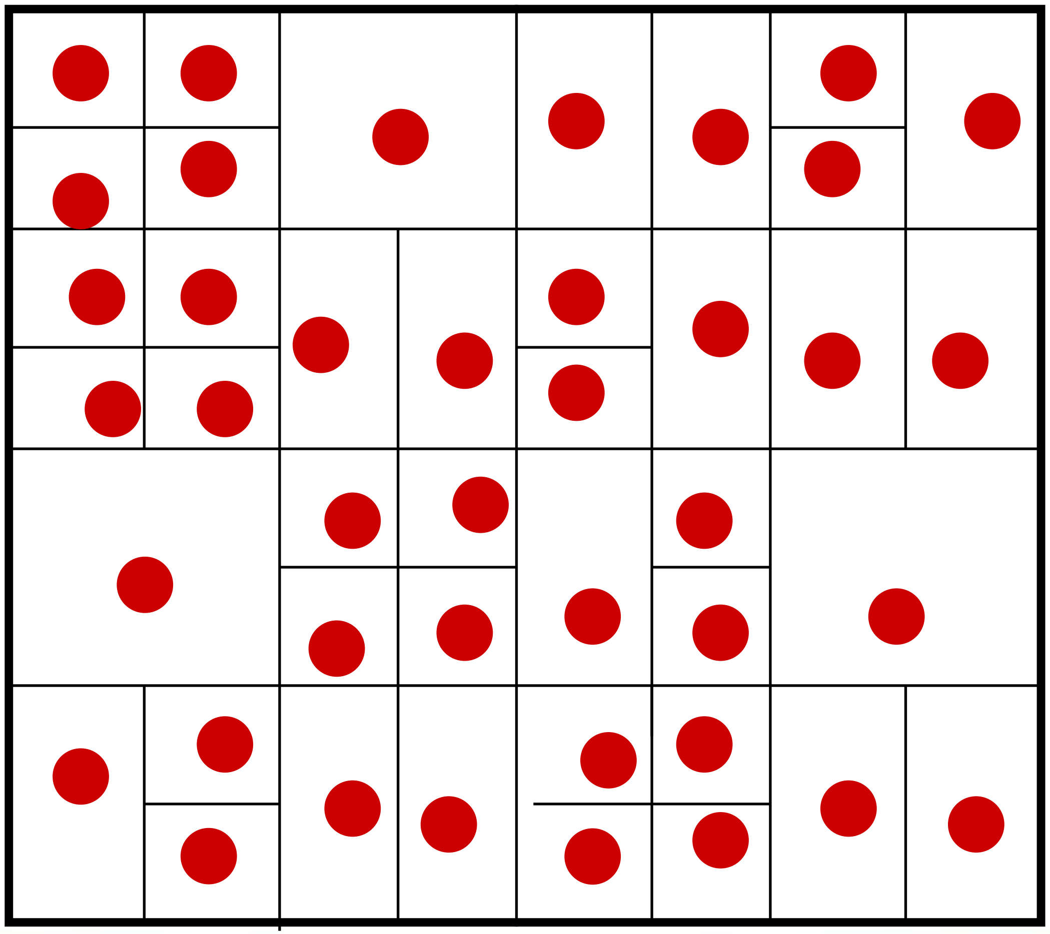


D-dimension space
Routing: progression within the space towards
The destination

CAN: Example



CAN: Example



CAN: routing

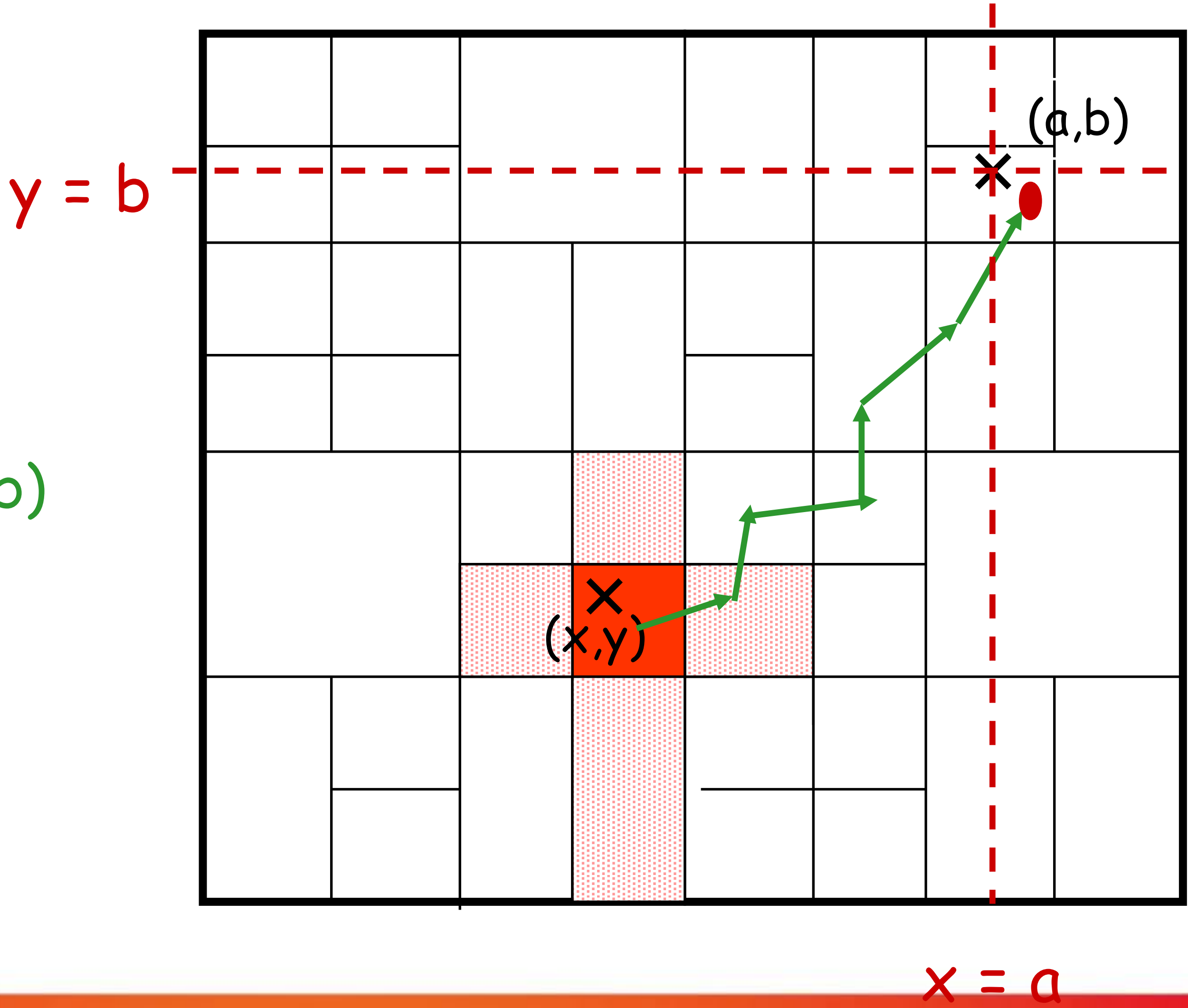
node X ::insert(K,V)

(1) $a = h_x(K)$

$b = h_y(K)$

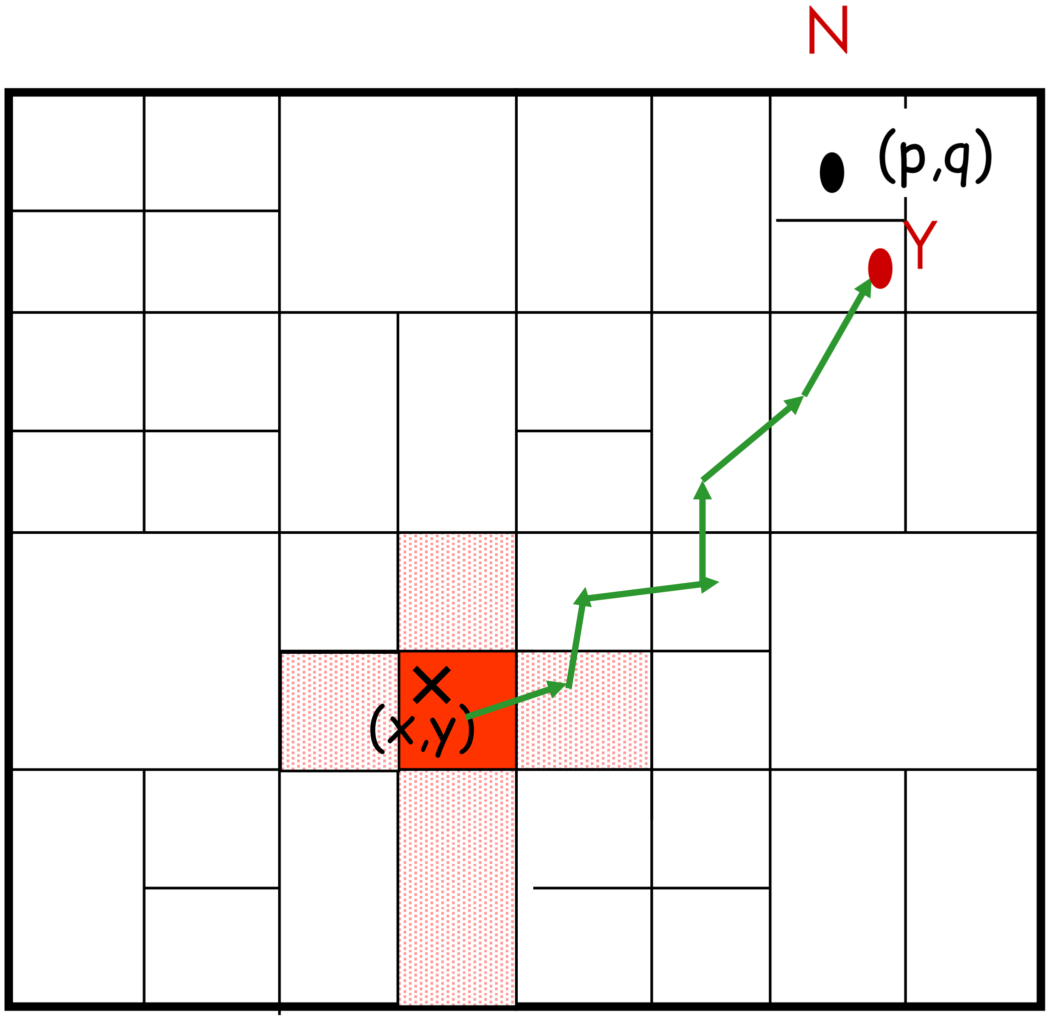
(2) route(K,V) -> (a,b)

(3) (a,b) stores (K,V)



CAN: node insertion

(1) Bootstrap : discovery of a contact node already participating to the CAN overlay network
(2) Selection of a random point (p,q) in the space
(3) Routing to (p,q) and discovery of node Y
(4) Zone splitting between Y and N



Insertion affects only Y and its immediate neighbours

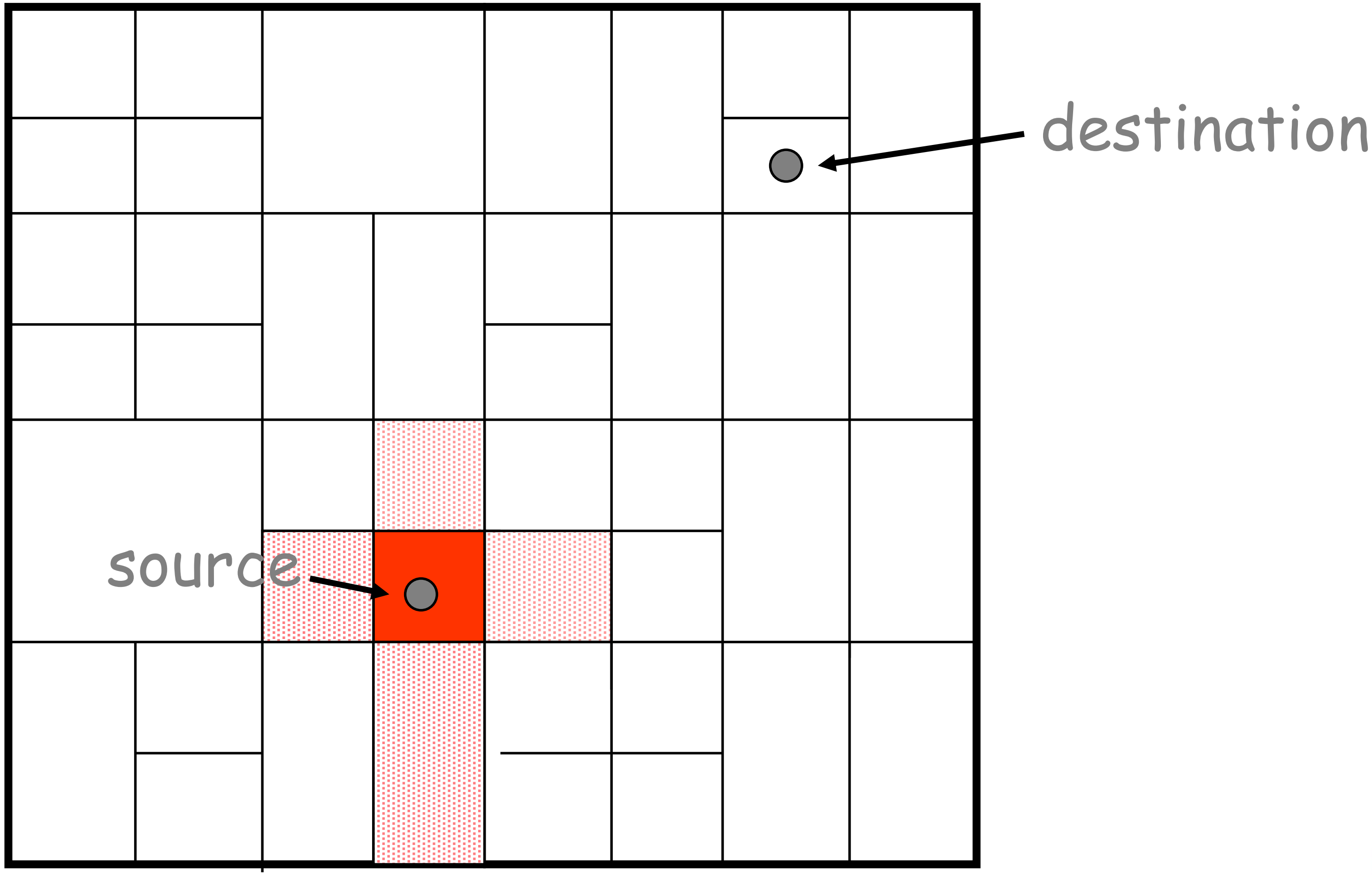
Routing information

- The joining node gets the IP @ of its neighbors from the previous owner of the zone
- Set of neighbors of the joining node is a sub-set of neighbors of the previous owner
- The previous owner updates its own list of neighbors
- The neighbors of the joining also update their state

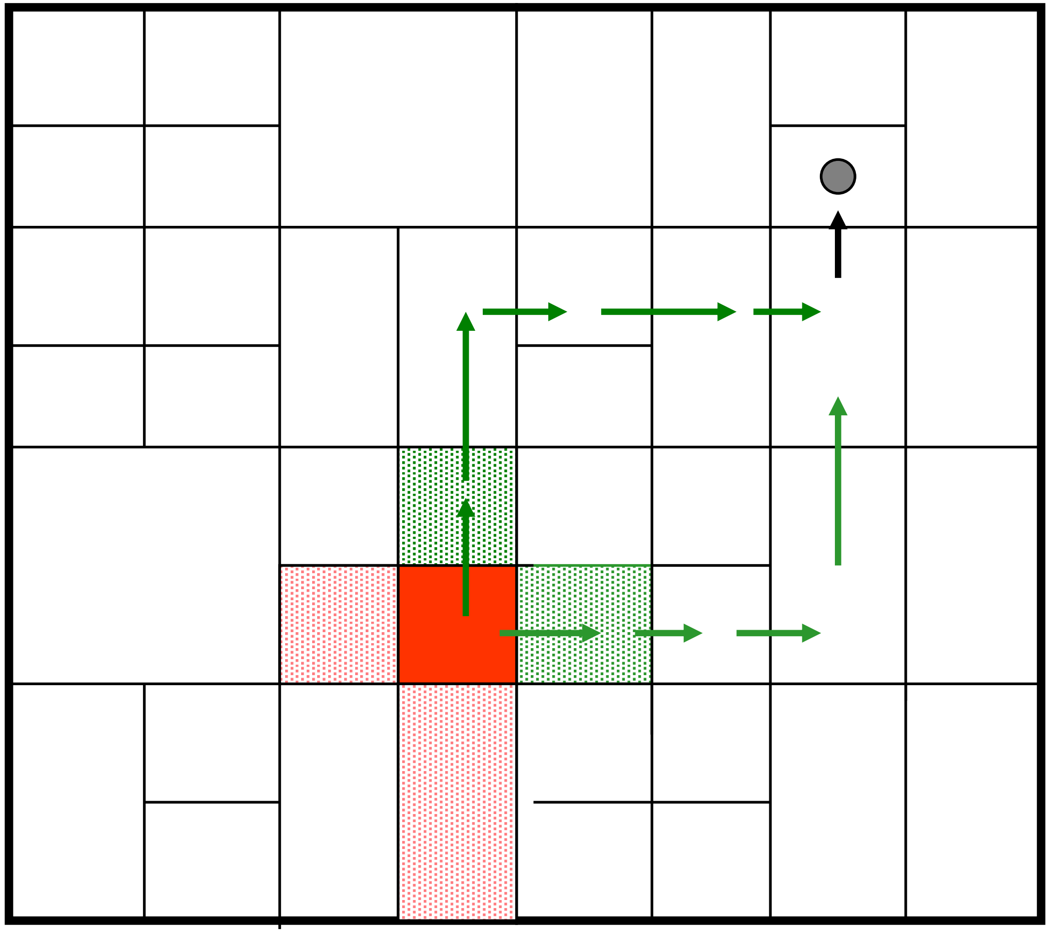
CAN: properties

- Each node maintains pointers to its immediate neighbours = 2d *O(d)*
- Routing in a N node network
  - Number of hops in a d-dimension space $O(d(n^{1/d}))$
  - In case of failure: selection of an alternative neighbor
- Optimizations
- Multiple dimensions
- Multiple reality
- RTT Measures
- Zone overloading
- Locality awareness: *landmarks*

Failure resilience



Failure resilience



Failure resilience

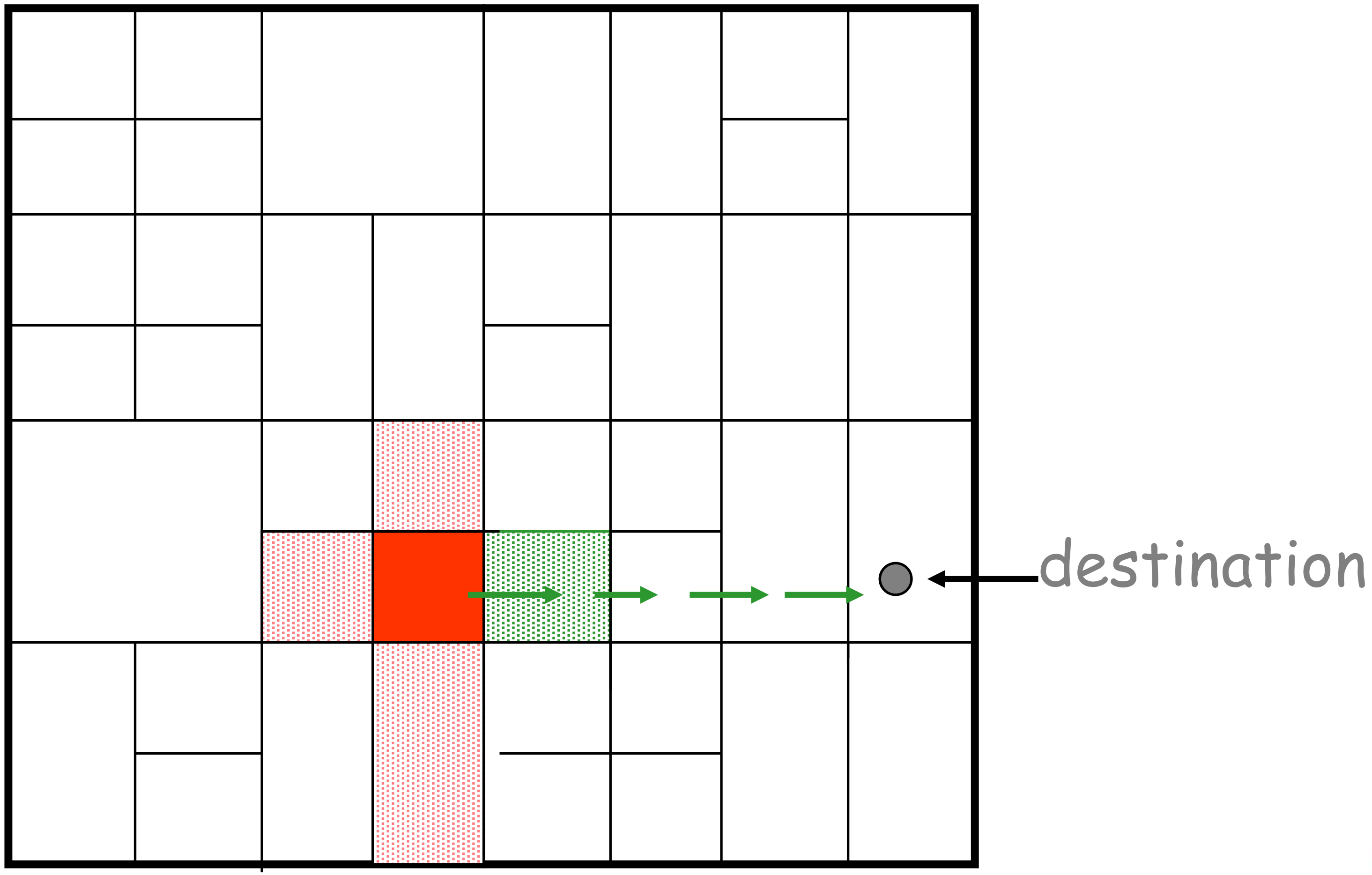
Node X::route(D)

If (X cannot progress directly towards D)

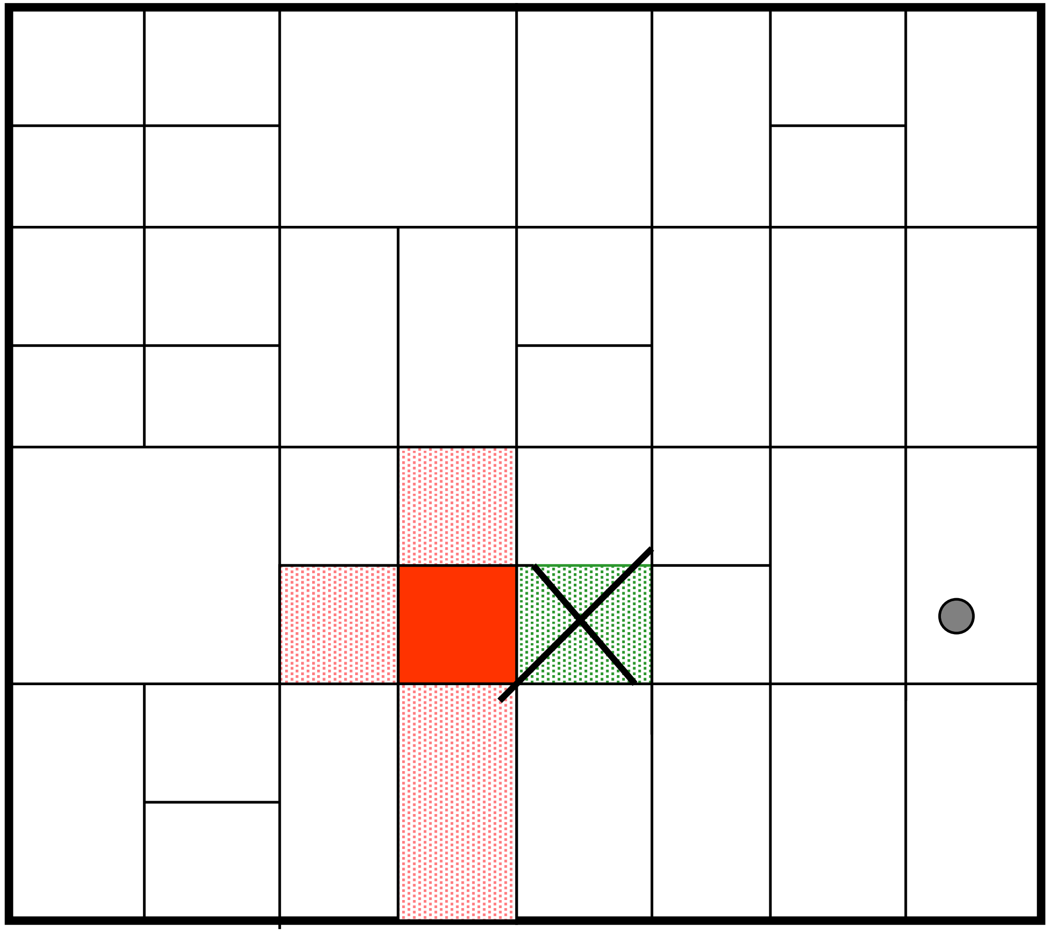
Check if one neighbour can progress towards the destination

If so, forward the message

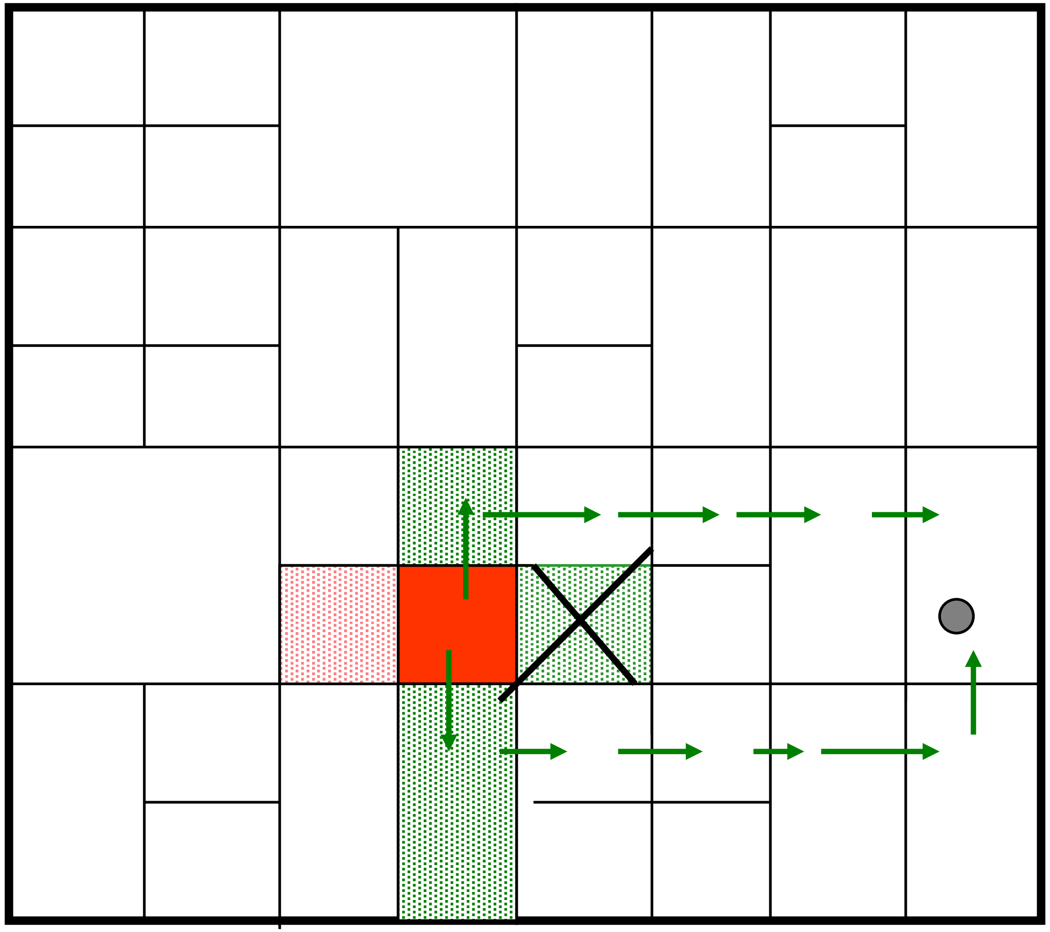
Failure resilience



Failure resilience



Routing resilience



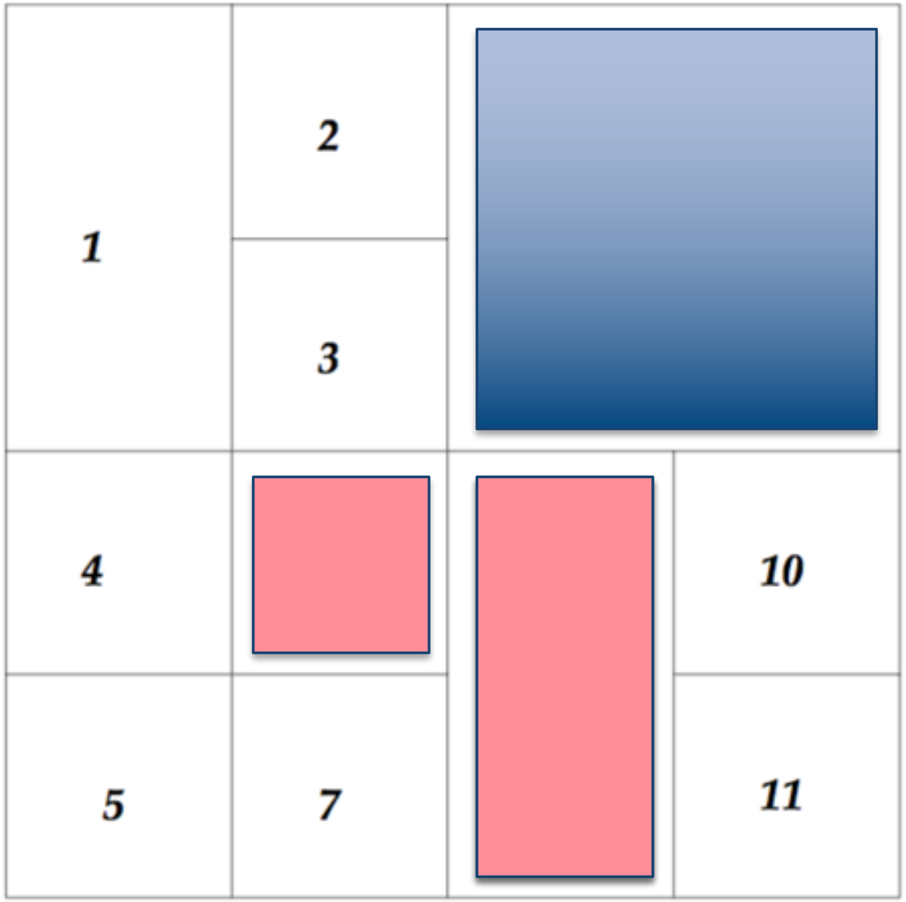
Departure, arrival, maintenance

Node departure

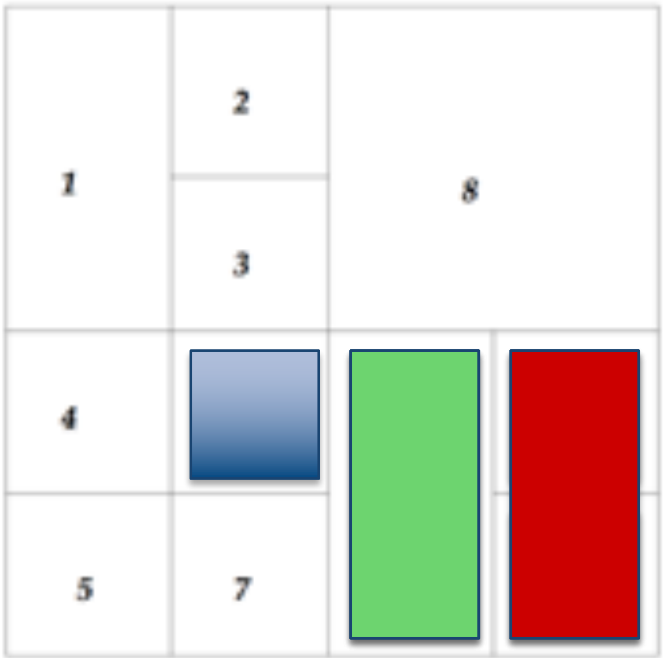
- A leaving node explicitly hands over its own zone (and associated database) to one of its neighbors
- Failure: Nodes send periodic heartbeat messages from to their neighbors. Missing heartbeats trigger Takeover.

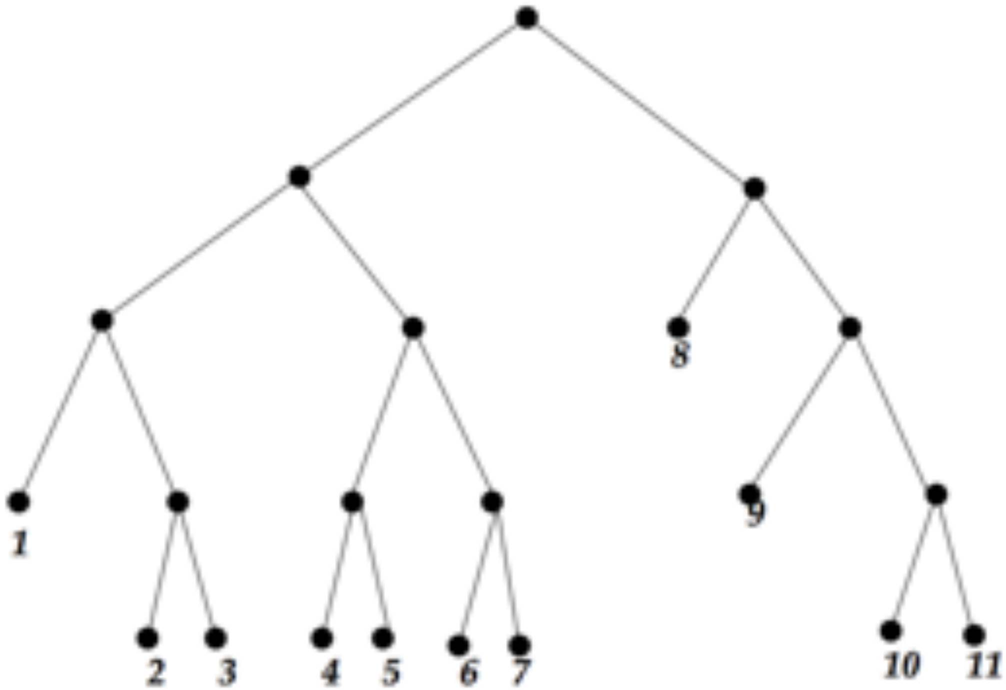
Important difference between the two
What is it?

Departure, arrival, maintenance



Background node reassignment





Multiple dimensions

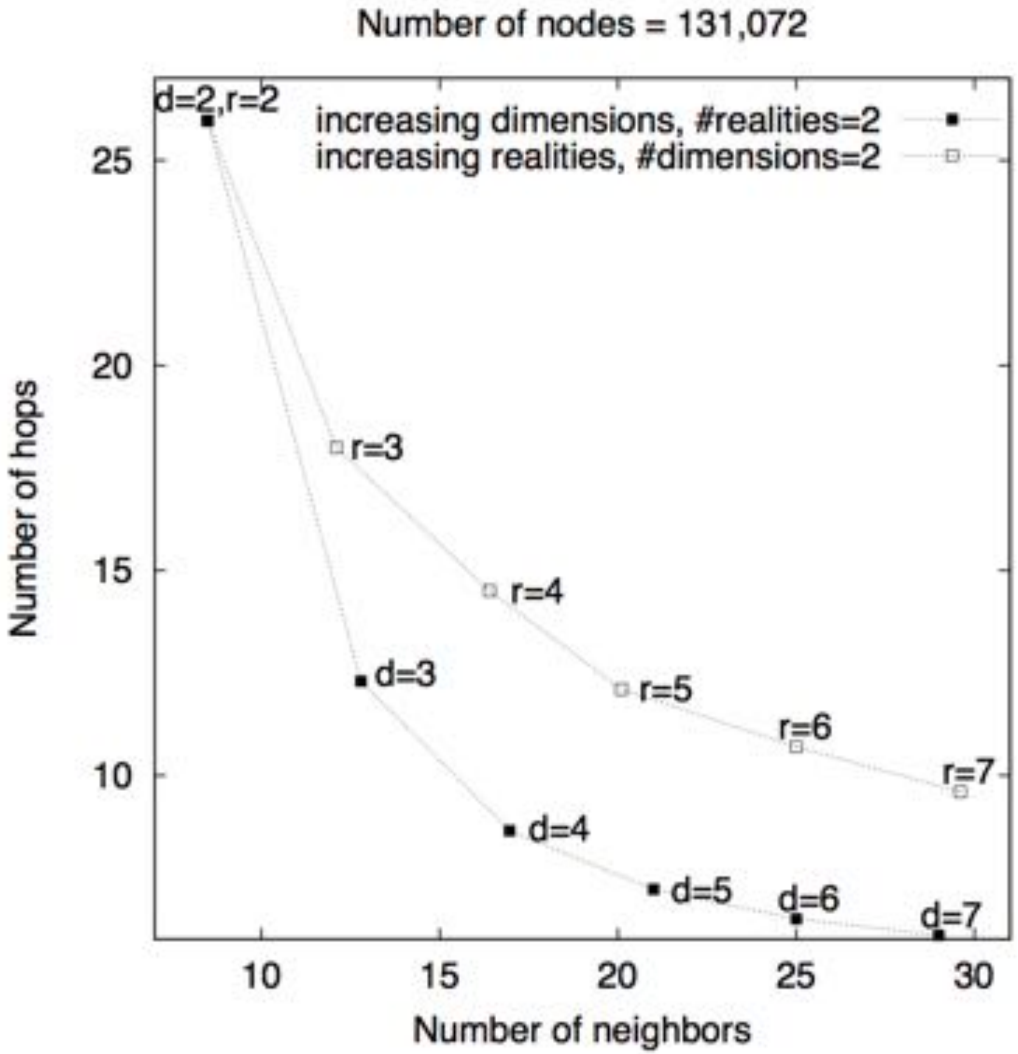
Increasing the number of dimensions

- The average path length is improved
- The number of neighbours increases linearly with the dimension
- Enhanced availability: potentially more nodes available

Multiple Realities

- Multiple independent coordinate spaces
- Associate each node with a different zone in each reality: r sets of coordinates
- Enhanced availability
  - DHT content can be replicated across realities
    - Ex: a pointer to a file stored at (x,y,z) is stored on three nodes responsible of point (x,y,z) in 3 realities
  - Improves average path length as well: depending on the destination, the most relevant reality is chosen

Dimensions vs Realities



RTT measures

- So far, the metric used to progress in the space in the path length in the Cartesian space
- Better criterion to take into account the underlying topology
- RTT to each neighbor
- Message forwarded to the neighbor for which the ratio progress/RTT is the best
  - Avoid long hops

Summary on structured overlay networks

Chord, Pastry and Tapestry use a generalized hypercube routing: prefix matching

- State maintained: O(Log(N))
- Number of routing hops: O(Log(N))
- Proximity routing in Pastry and Tapestry

CAN uses progression in a multidimensional Cartesian space

- State maintained: O(D)
- Number of routing hops: $O(N^{1/D})$
- Proximity routing more difficult to exploit

DHT Functionality=Exact match interface

Referencesqwertop[]
\
[p;oir ewQ /?>x Z
[]

- Sylvia Ratnasamy et al. **Scalable Content-Addressable Network.** *SIGCOMM 2001*
- Ion Stoica, et al. 2003. **Chord: a scalable peer-to-peer lookup protocol for internet applications**. *IEEE/ACM Trans. Netw.* 11, 1 (February 2003), 17-32.
- Many more: Google P2P structured overlay networks