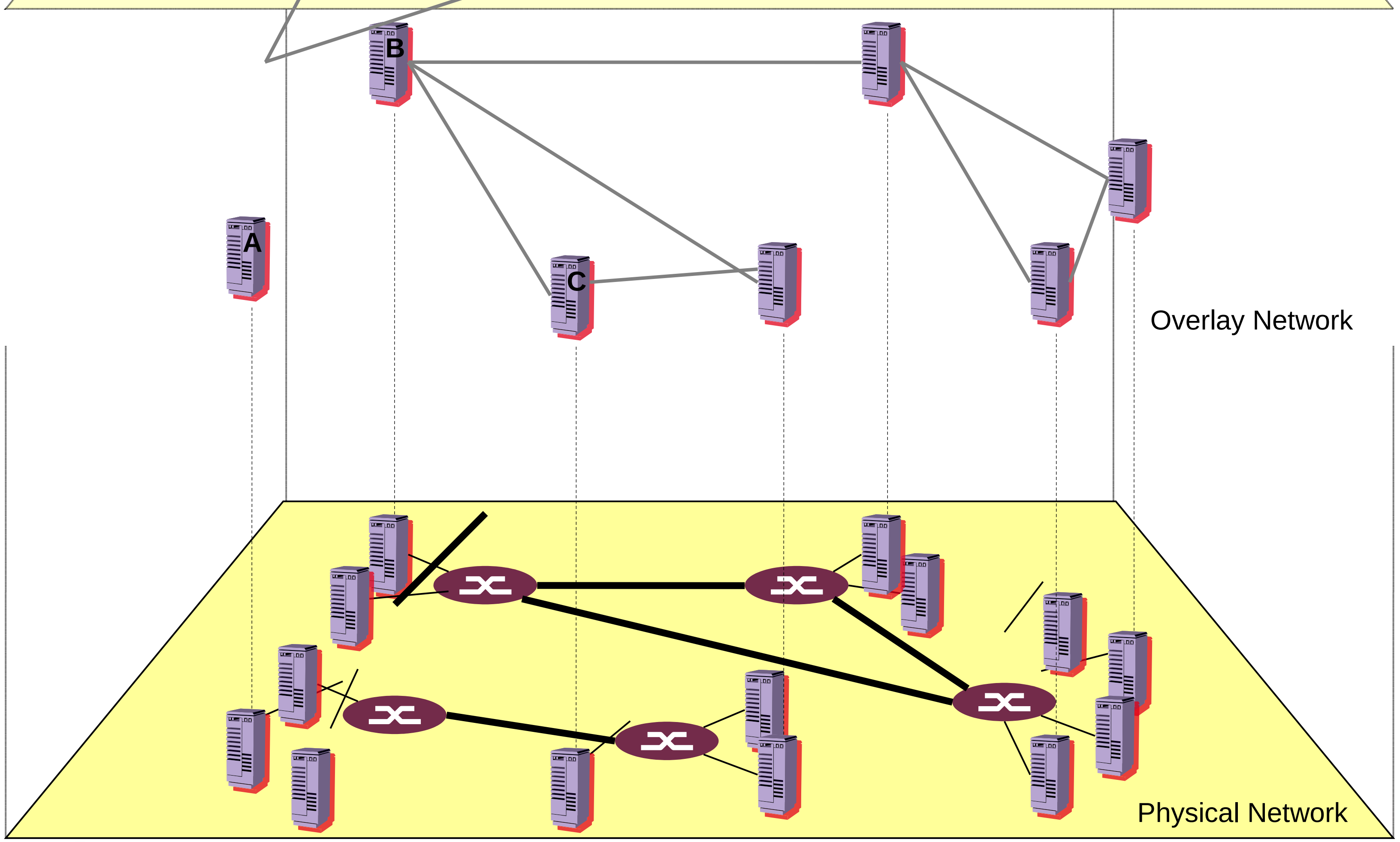
# TLC
# Pastry

Quentin Dufour
Credits : Davide Frey

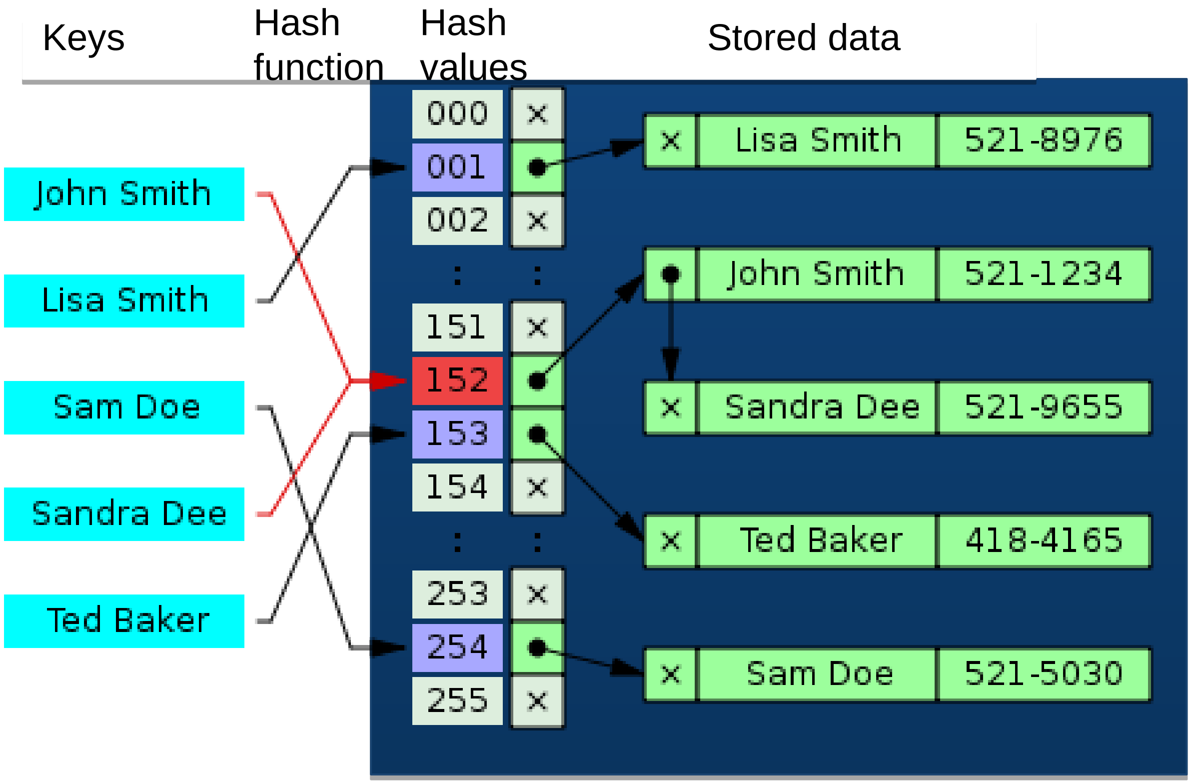
# Key Concept: Overlay Network



# Overlay types

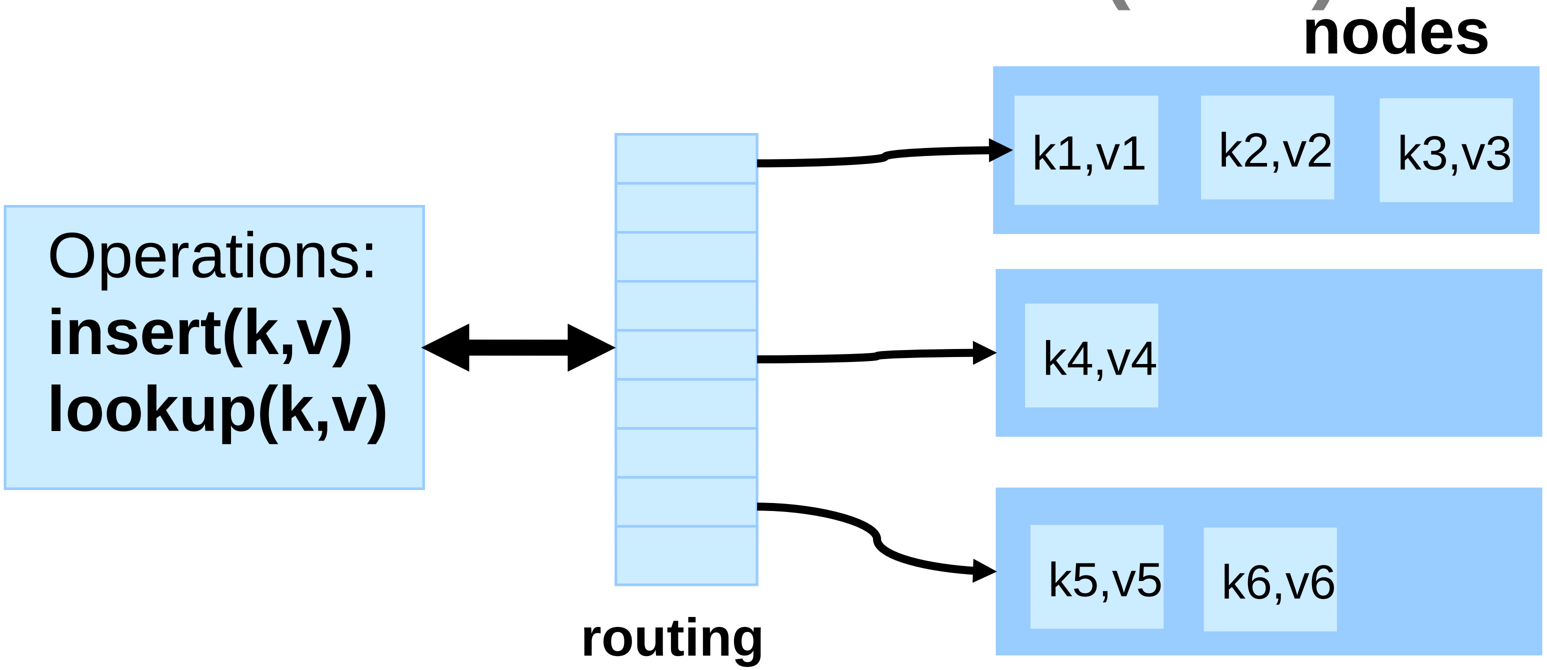
| Unstructured P2P | Structured P2P |
|---|---|
| ○ Any two nodes can establish a link<br>○ Topology evolves at random<br>○ Topology reflects desired properties of linked nodes | ○ Topology strictly determined by node IDs |

# Hash Table



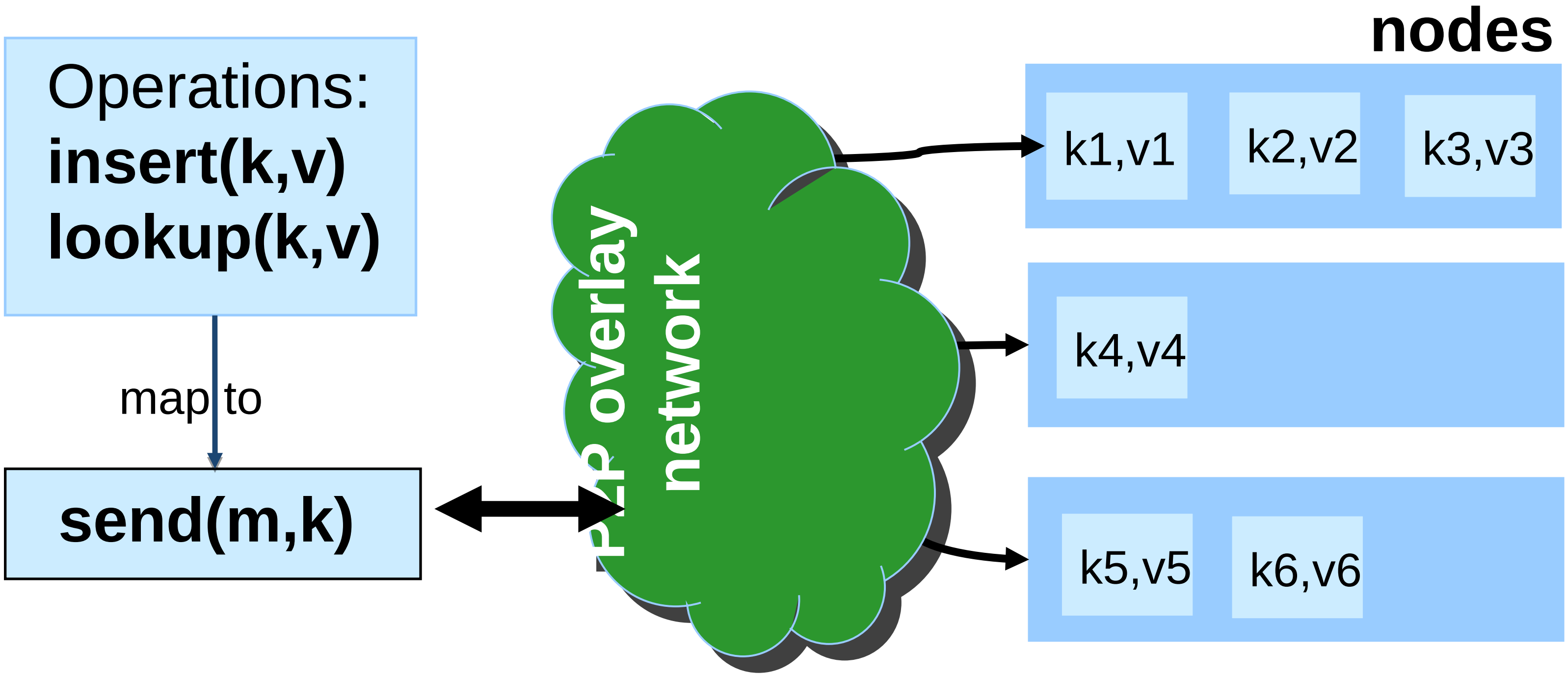

Efficient information lookup

# Distributed Hash Table (DHT)




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

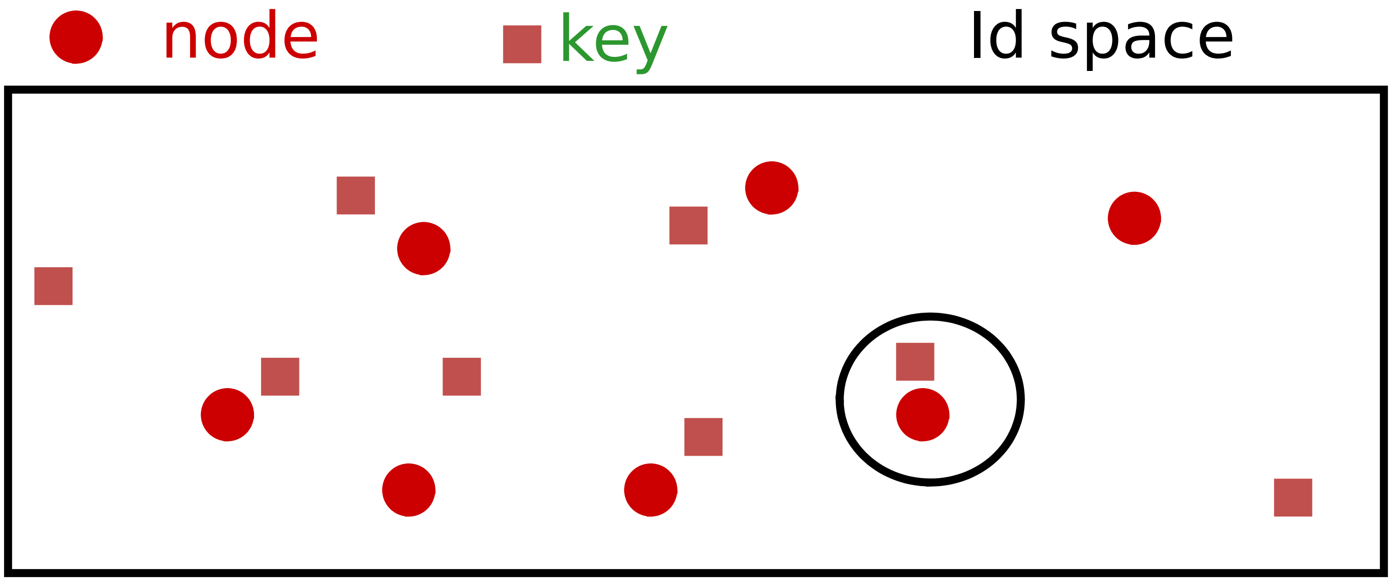
# Distributed Hash Table




- 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*

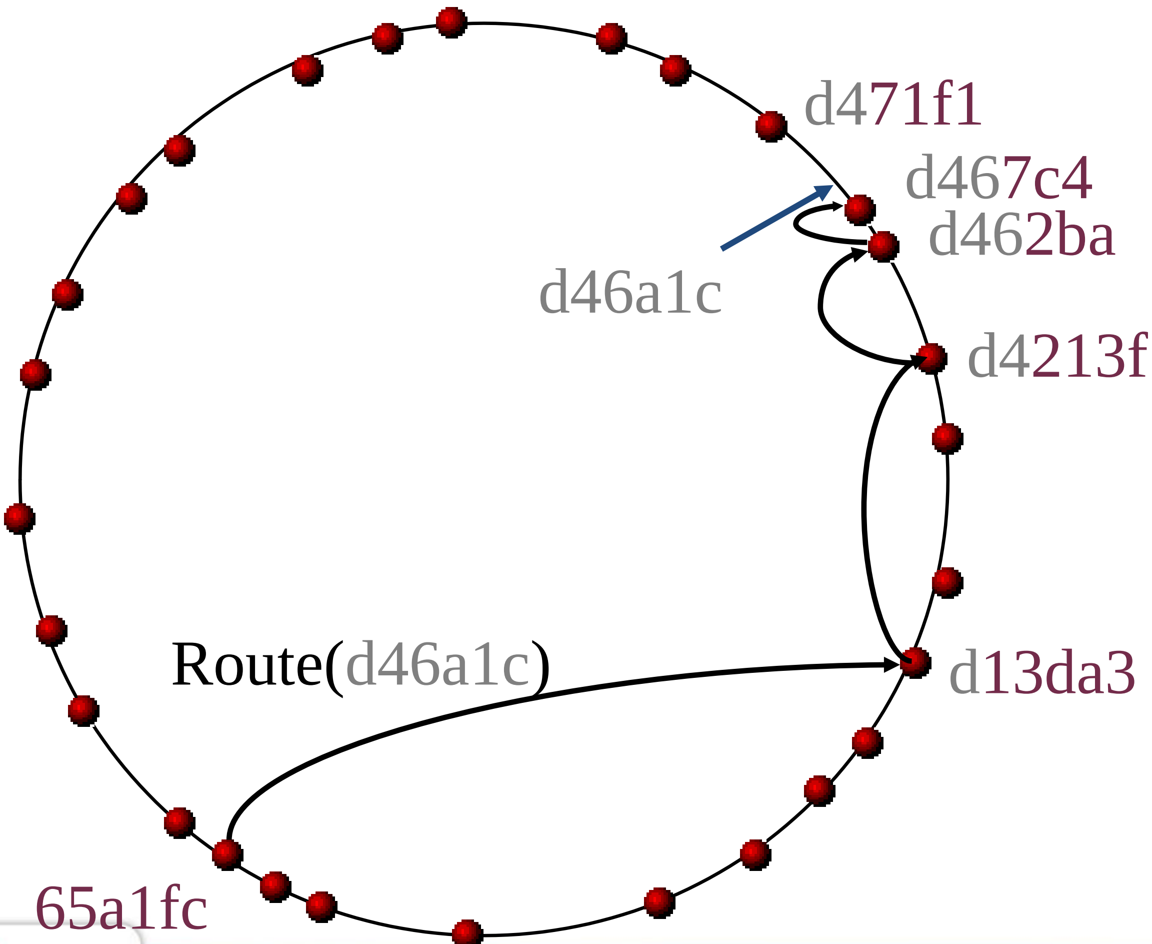
- *Bias selection of nodes*

# 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 $D$

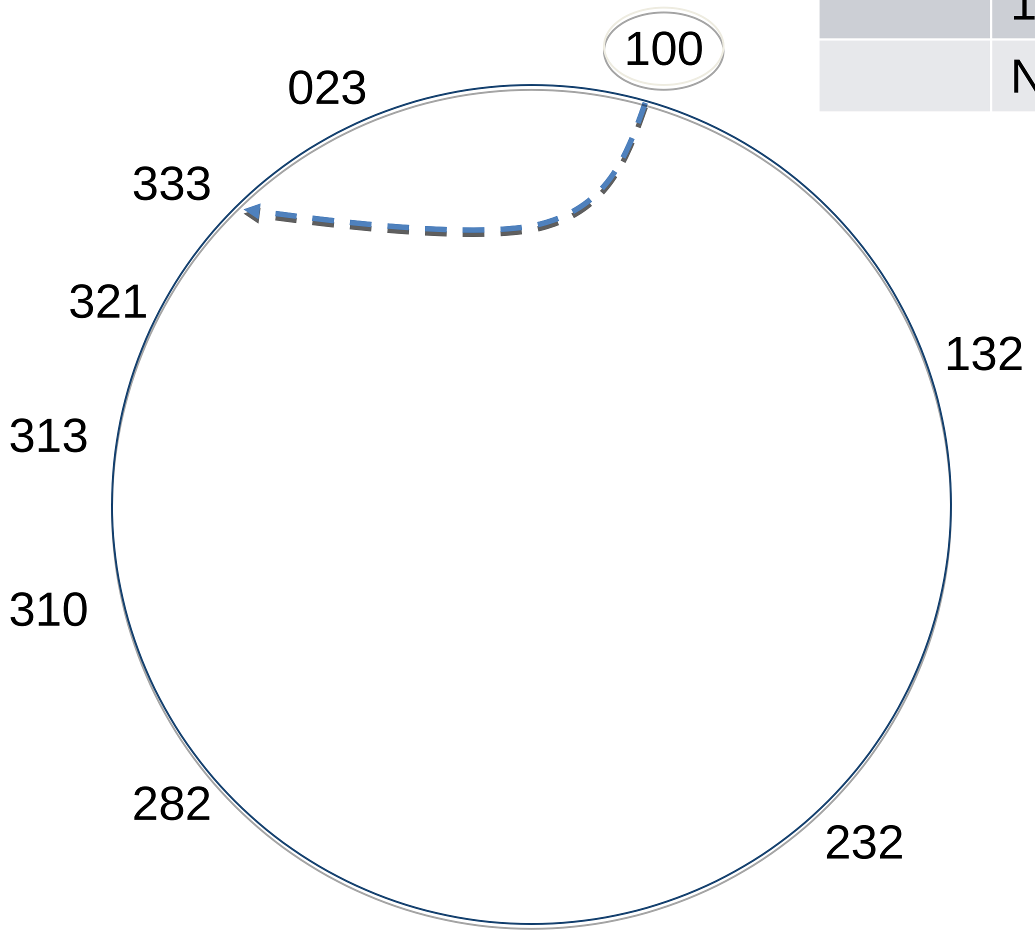
$SHL(A, B)$ : length of the shared prefix between A and B

# Pastry Example

| | | | |
|---|---|---|---|
| **023** | | **232** | **333** |
| | 113 | 122 | 132 |
| | N/A | 102 | 103 |

b=4

Route to 311

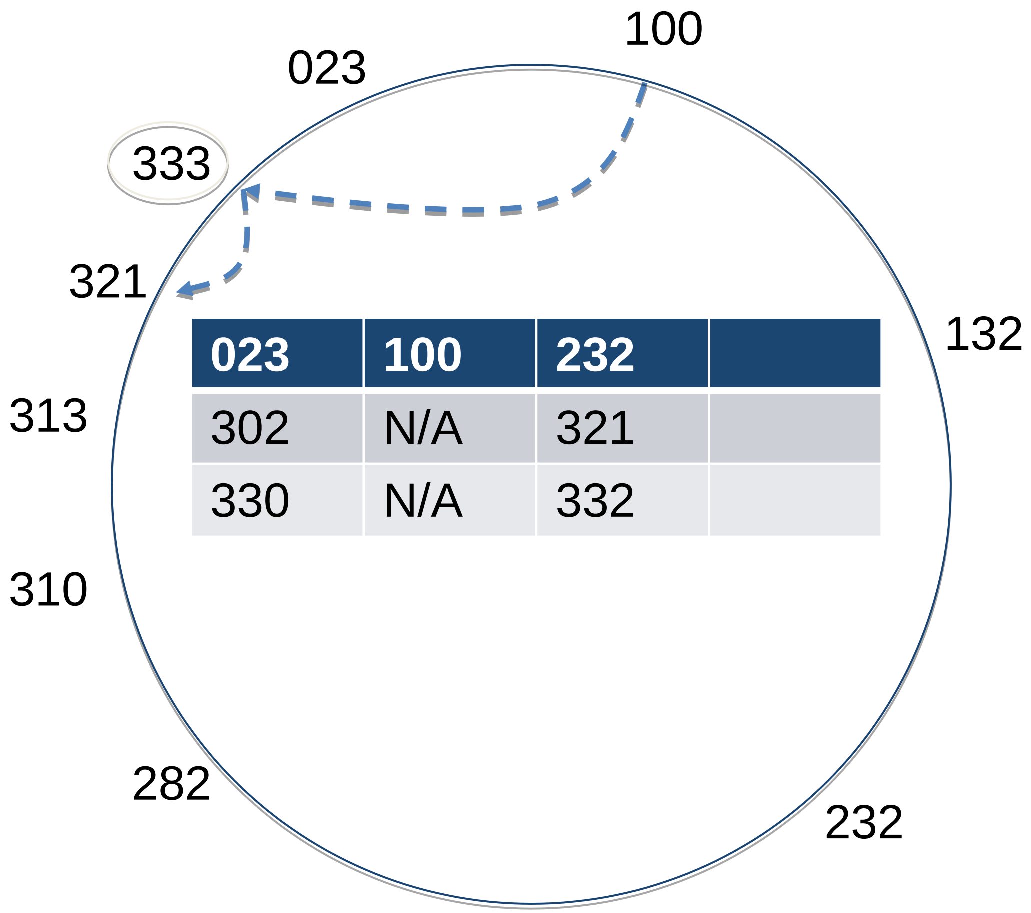



For space reasons some nodes are not shown on the ring

# Pastry Example

b=4

Route to 311



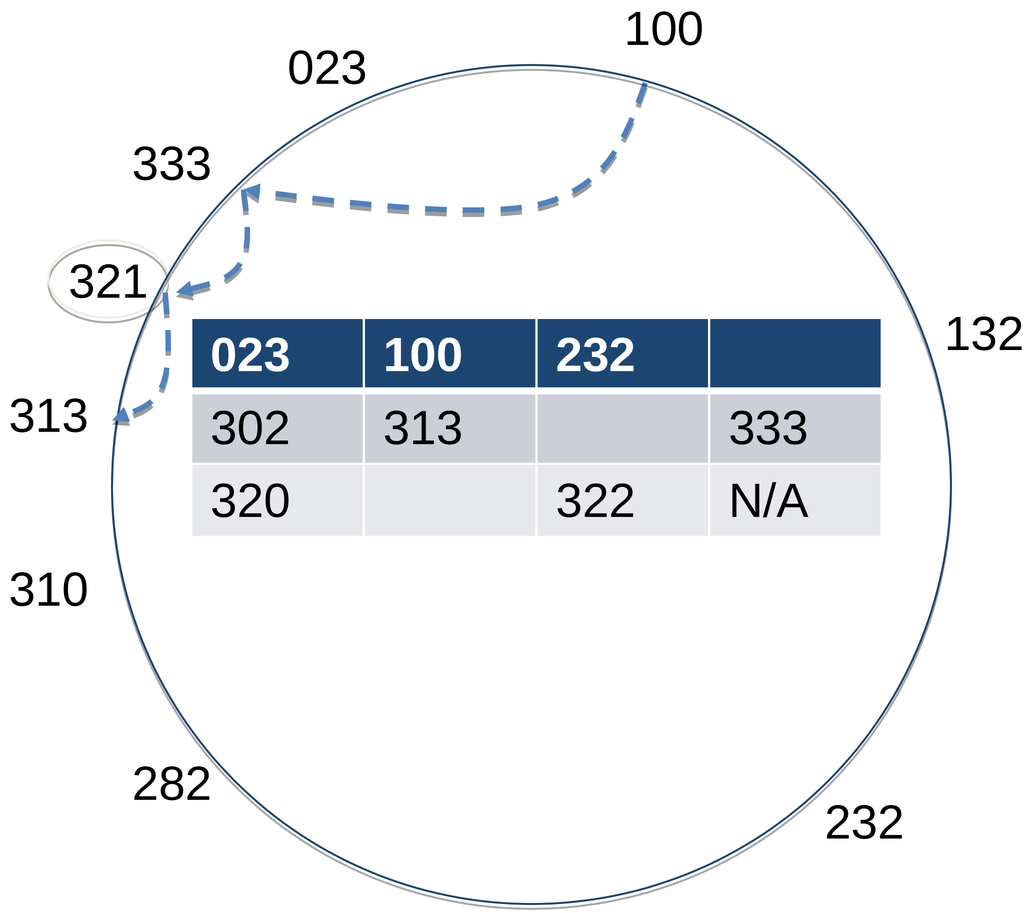

| 023 | 100 | 232 | |
|---|---|---|---|
| 302 | N/A | 321 | |
| 330 | N/A | 332 | |

For space reasons some nodes are not shown on the ring

# Pastry Example

b=4

Route to 311



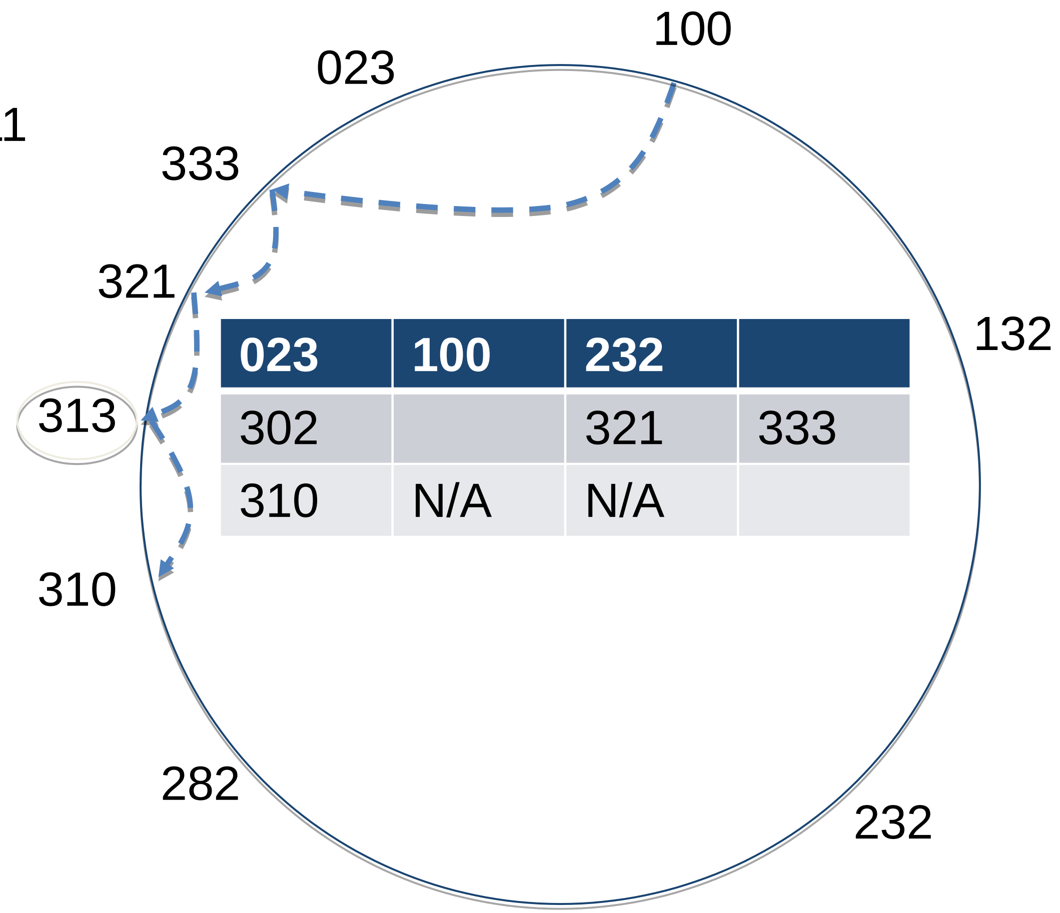

| 023 | 100 | 232 | |
|---|---|---|---|
| 302 | 313 | | 333 |
| 320 | | 322 | N/A |

For space reasons some nodes are not shown on the ring

# Pastry Example

b=4

Route to 311



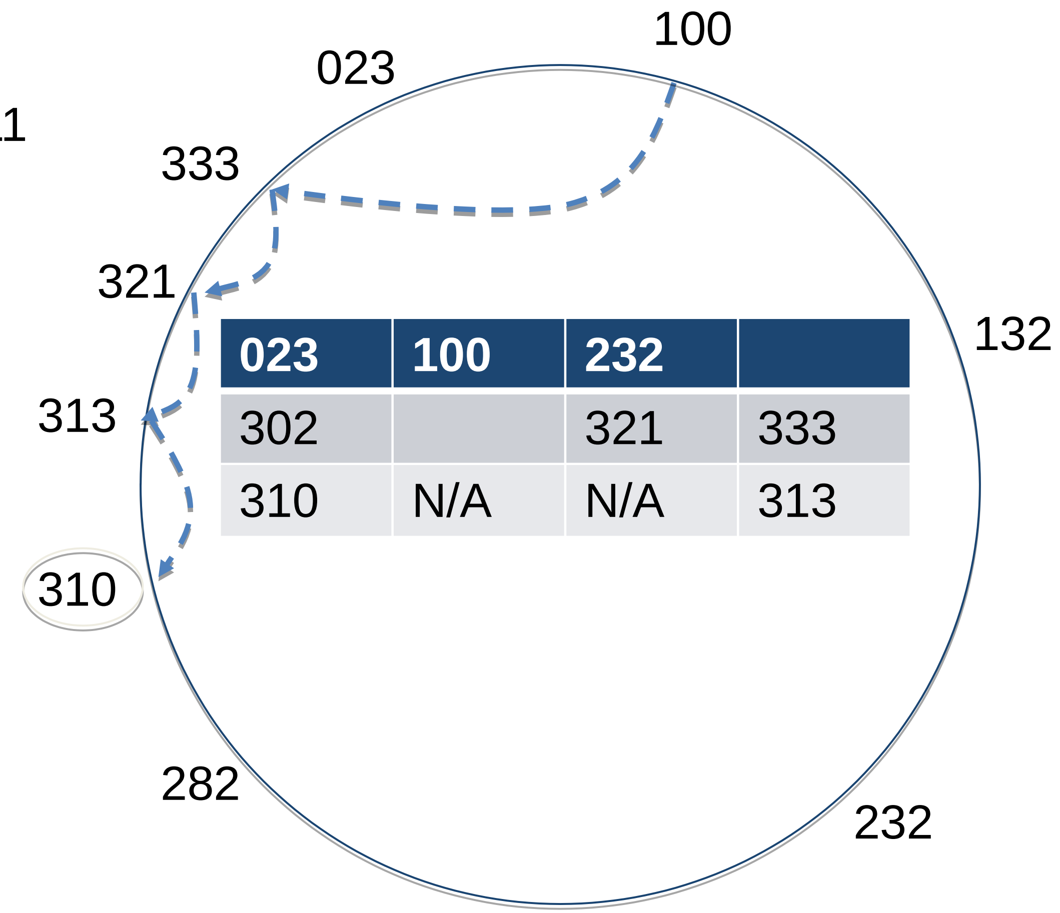

| 023 | 100 | 232 | |
|---|---|---|---|
| 302 | | 321 | 333 |
| 310 | N/A | N/A | |

For space reasons some nodes are not shown on the ring

# Pastry Example

b=4

Route to 311




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

For space reasons some nodes are not shown on the ring

# 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*

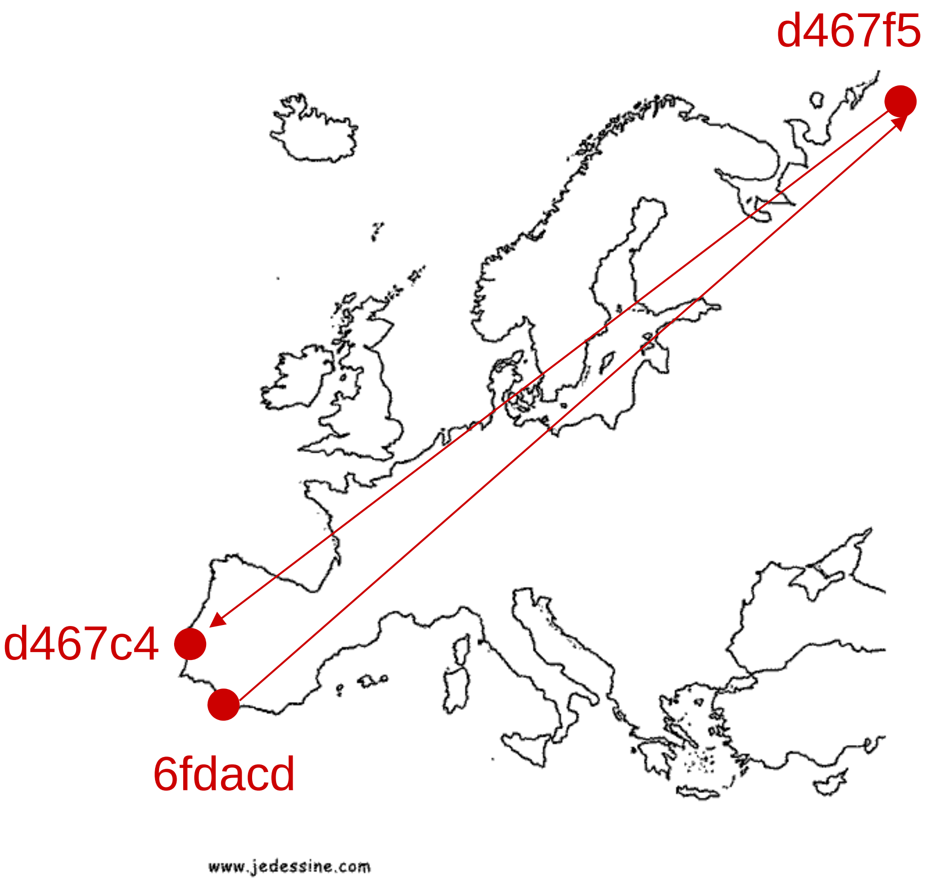
- is aggressively monitored and fixed

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

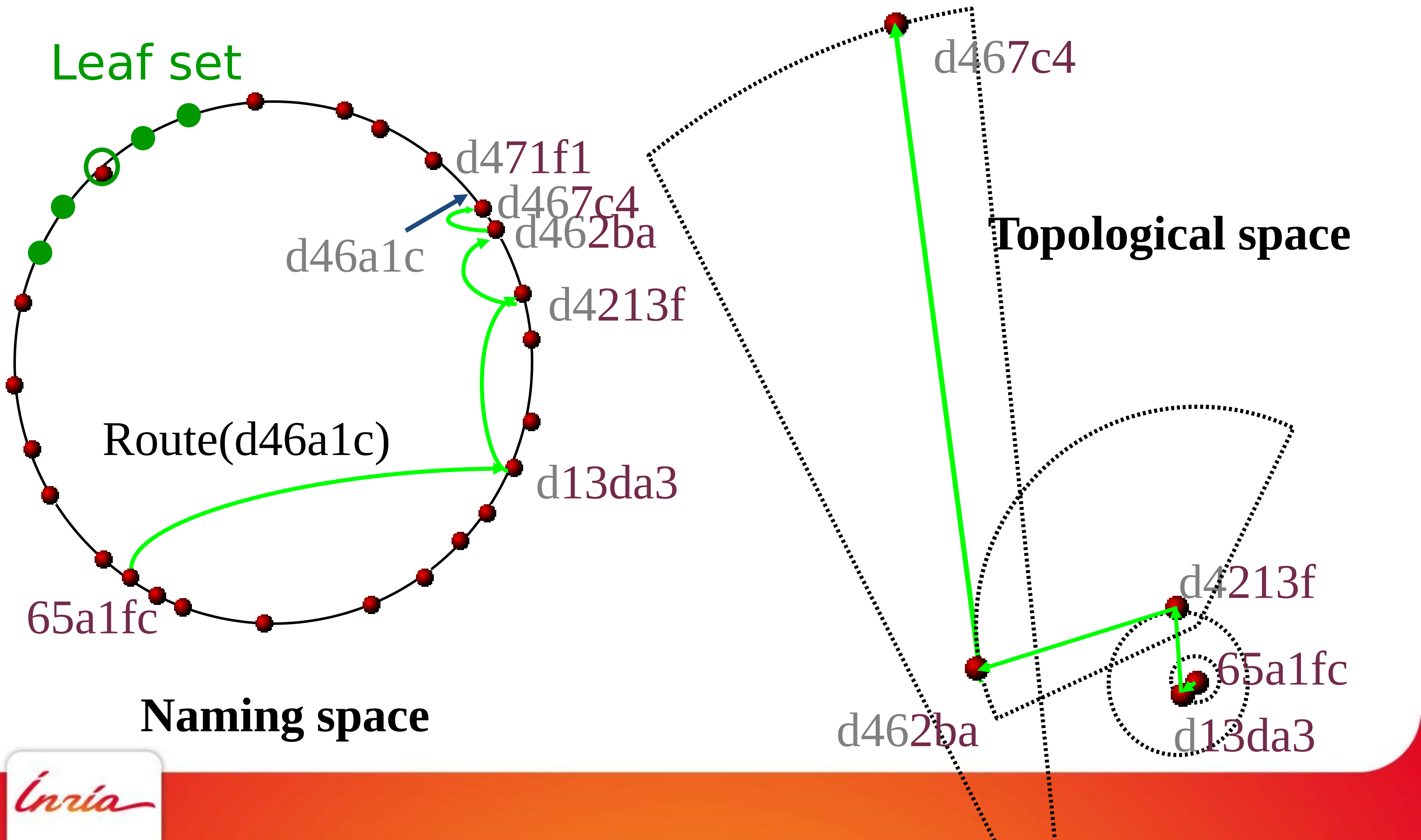
routeTable(i,j):

- *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

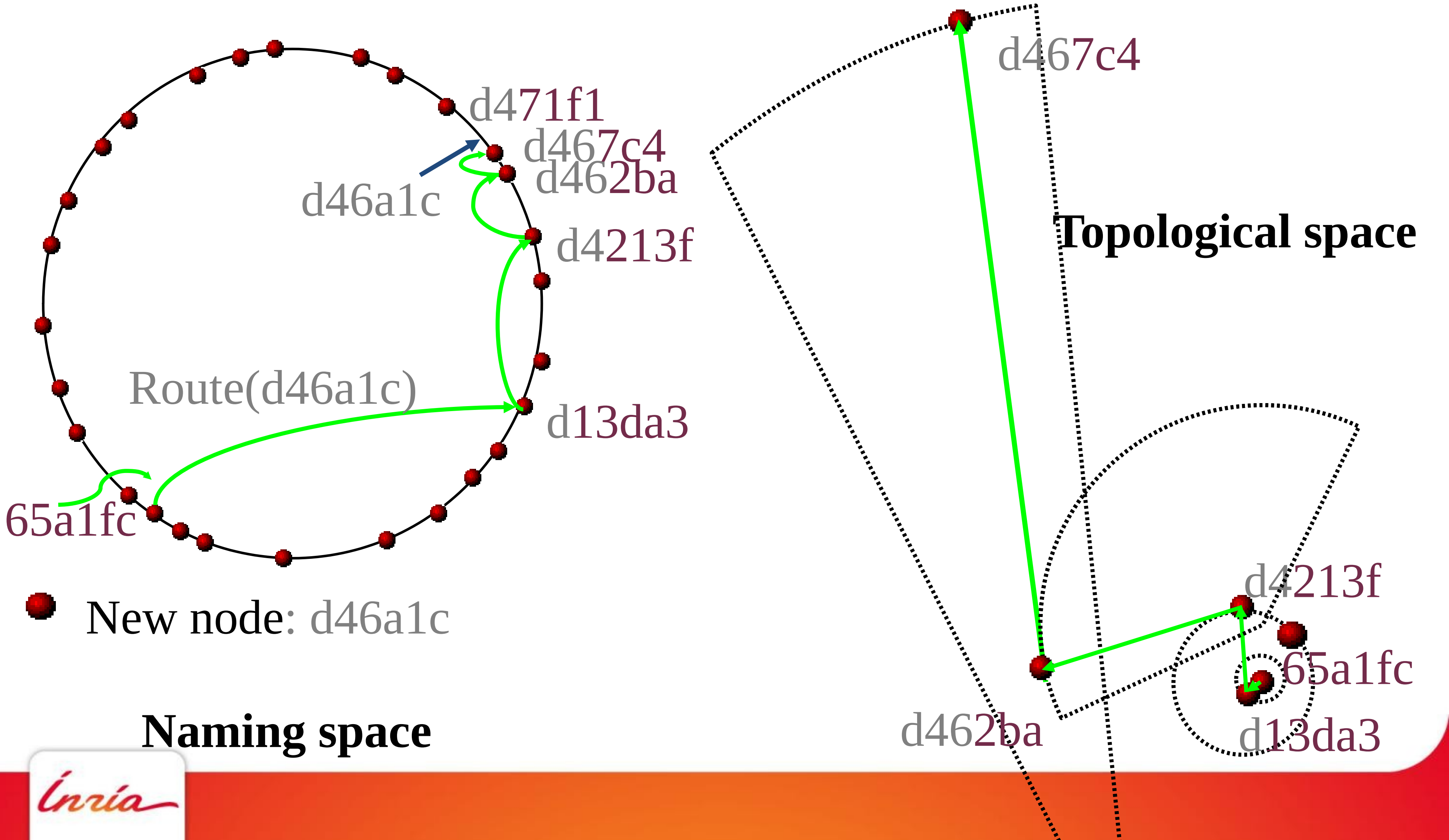
# Proximity routing in Pastry



# Locality

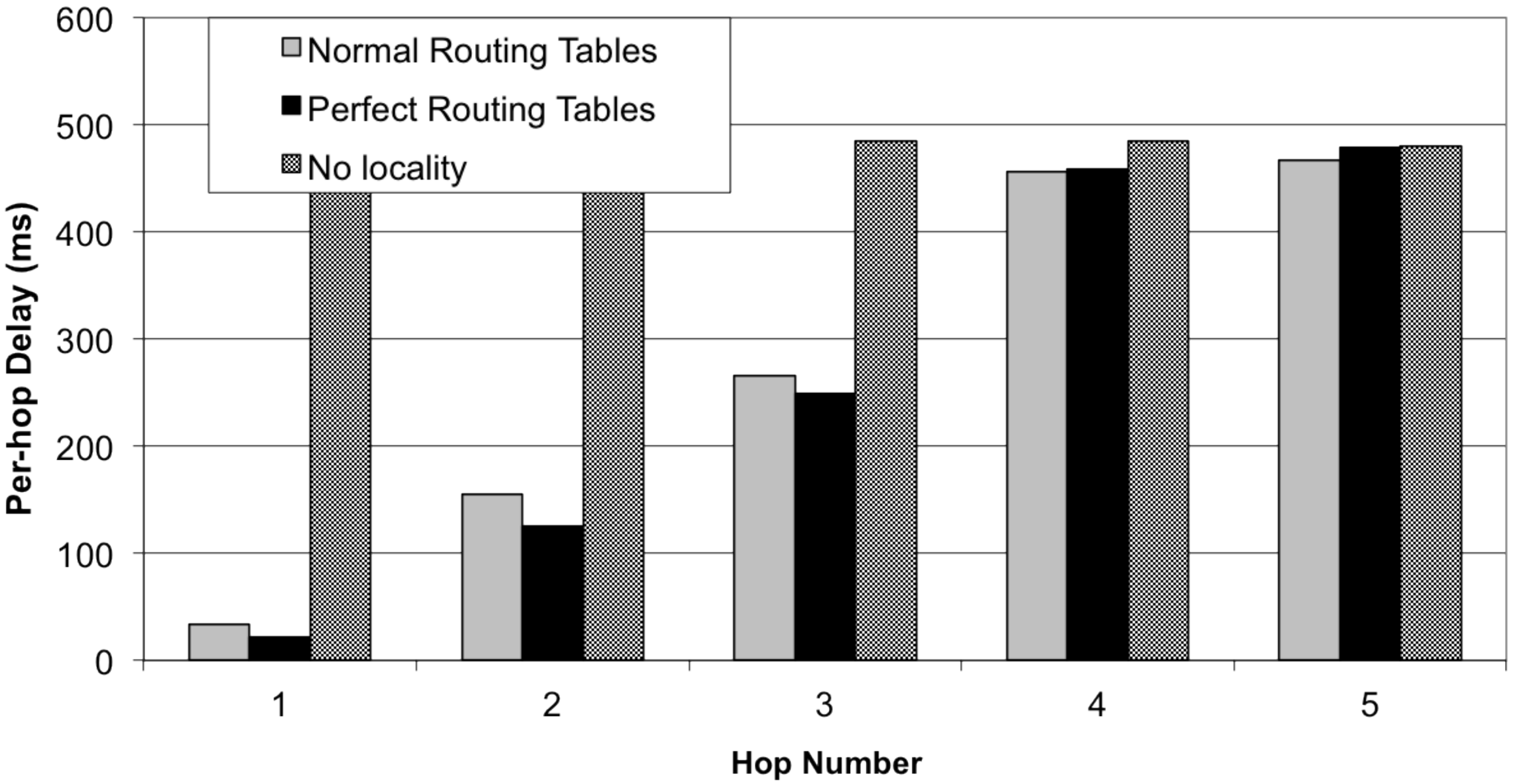
1. Joining node X routes asks A to route to X
   - Path A,B,… -> Z
   - Z numerically closest to X
   - X initializes line i of its routing table with the contents of line i of the routing table of the *ith* node encountered on the path
2. Improving the quality of the routing table
   - X asks to each 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.