Pastry

Peter Druschel, *Rice University*

Antony Rowstron, *Microsoft Research UK*

Some slides are taken from the authors original presentation

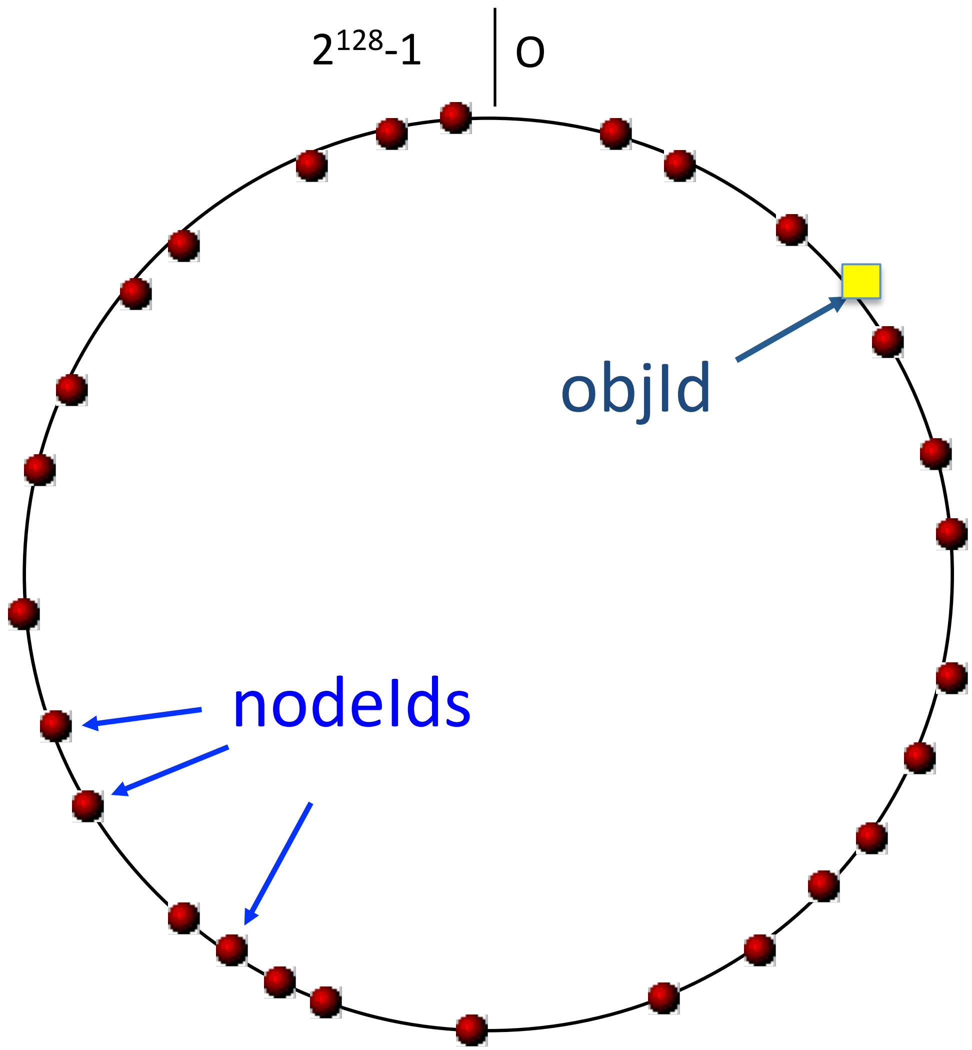
What is Pastry?

Pastry is a structured P2P network

What is Pastry

- Self-organizing overlay network
- Lookup/insert object in $< \log_{16} N$ routing steps (expected)
- $O(\log N)$ per-node state (for routing table)
- Network proximity routing

Pastry: Object distribution



Consistent hashing [*Karger et al. '97*]

128 bit circular id space

nodeIds (uniform random)

objIds (uniform random)

Invariant: node with numerically closest nodeId maintains object

Pastry: Routing

0x	1x	2x	3x	4x	5x		7x	8x	9x	ax	bx	cx	dx	ex	fx
60x	61x	62x	63x	64x		66x	67x	68x	69x	6ax	6bx	6cx	6dx	6ex	6fx
650x	651x	652x	653x	654x	655x	656x	657x	658x	659x		65bx	65cx	65dx	65ex	65fx

Log_{16} N rows

Leaf set

Routing table for node 65a1fc (b=4, so $2^b = 16$)

Pastry Node State

State of node 10233102

Leaf set

SMALLER		LARGER	
10233033	10233021	10233120	10233122
10233001	10233000	10233230	10233232

Set of nodes with |L|/2 smaller and |L|/2 larger numerically closest NodeIds

Routing table

-0-2212102	1	-2-2301203	-3-1203203
0	1-1-301233	1-2-230203	1-3-021022
10-0-31203	10-1-32102	2	10-3-23302
102-0-0230	102-1-1302	102-2-2302	3
1023-0-322	1023-1-000	1023-2-121	3
10233-0-01	1	10233-2-32	
0		102331-2-0	
		2	

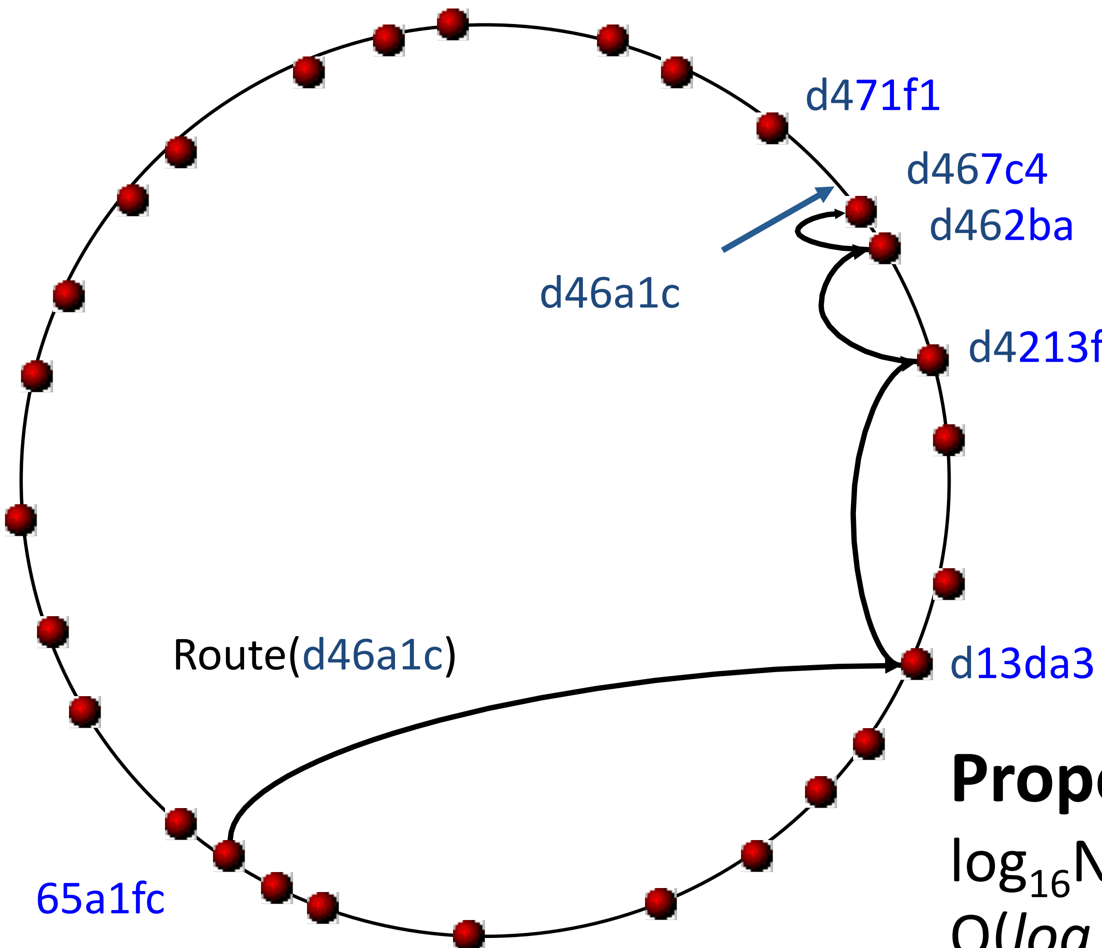
Prefix-based routing entries

Neighborhood set

13021022	10200230	11301233	31301233
02212102	22301203	31203203	33213321

|M| "physically" closest nodes

Pastry: Routing

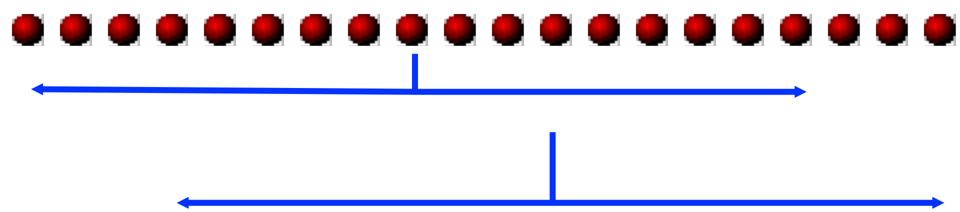


Properties

$\log_{16} N$ steps to search

O(*log N*) size of routing table

Pastry: Leaf sets



Each node maintains IP addresses of the nodes with the L/2 numerically closest larger and smaller nodeIds, respectively.

- routing efficiency/robustness
- fault detection (keep-alive)
- application-specific local coordination

Pastry: Routing procedure

if (destination is "within range of our leaf set")
    forward to numerically closest member
else
    let *l* = length of shared prefix
    let *d* = value of *l*-th digit in *D*'s address
    if (R_l^d exists) forward to R_l^d
        (R_l^d = l^{th} row & d^{th} col of routing table)
    else forward to a known node that
        (a) shares at least as long a prefix, and
        (b) is numerically closer than this node

[Prefix routing]

Pastry: Performance

Integrity of overlay/ message delivery:

- guaranteed unless *L/2* simultaneous failures of nodes with adjacent nodeIds occur

Number of routing hops:

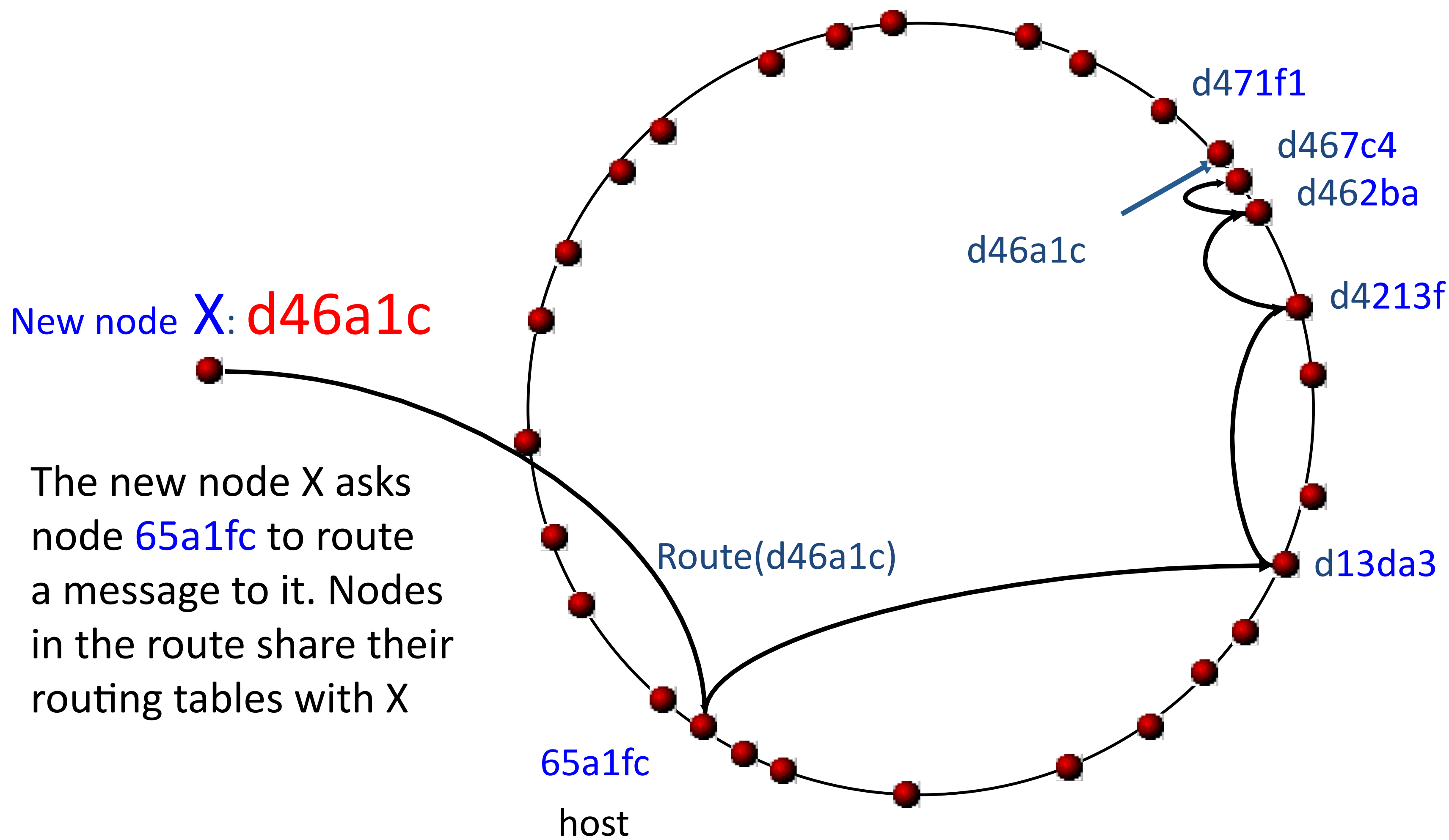
- No failures: < $log_{16} N$ expected
- *O(N)* worst case (why?), average case much better

Pastry: Self-organization

Initializing and maintaining routing tables and leaf sets

- Node addition
- Node departure (failure)

Pastry: Node addition



The new node X asks node 65a1fc to route a message to it. Nodes in the route share their routing tables with X

Node departure (failure)

Leaf set members exchange heartbeat messages

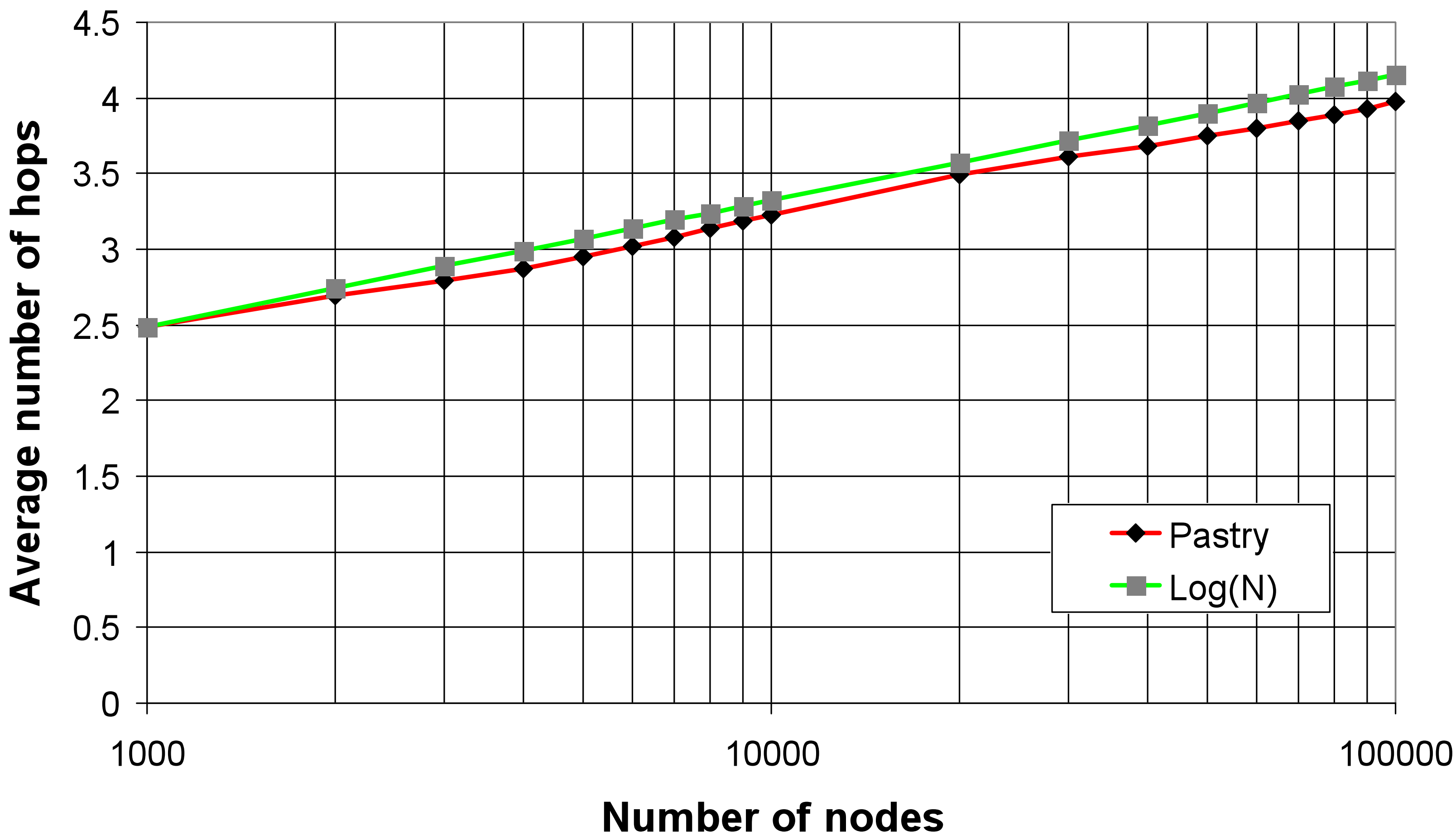
- **Leaf set repair (eager):** request set from farthest live node in set
- **Routing table repair (lazy):** get table from peers in the same row, then higher rows

Node departure (failure)

Leaf set members exchange heartbeat

- **Leaf set repair (eager):** request the set from farthest live node
- **Routing table repair (lazy):** get table from peers in the same row, then higher rows

Pastry: Average # of hops



L=16, 100k random queries

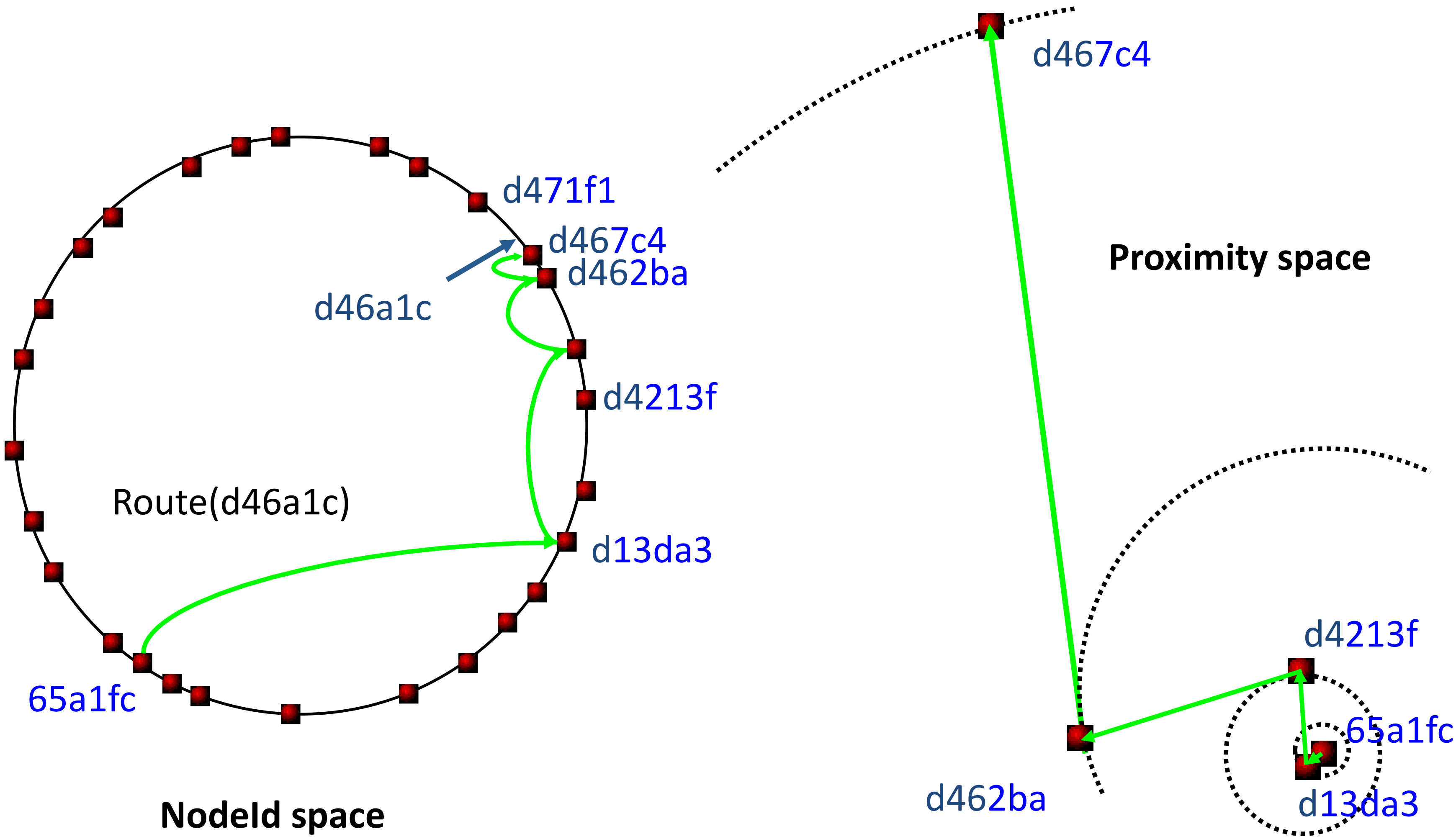
Pastry: Proximity routing

Proximity metric = time delay estimated by a ping

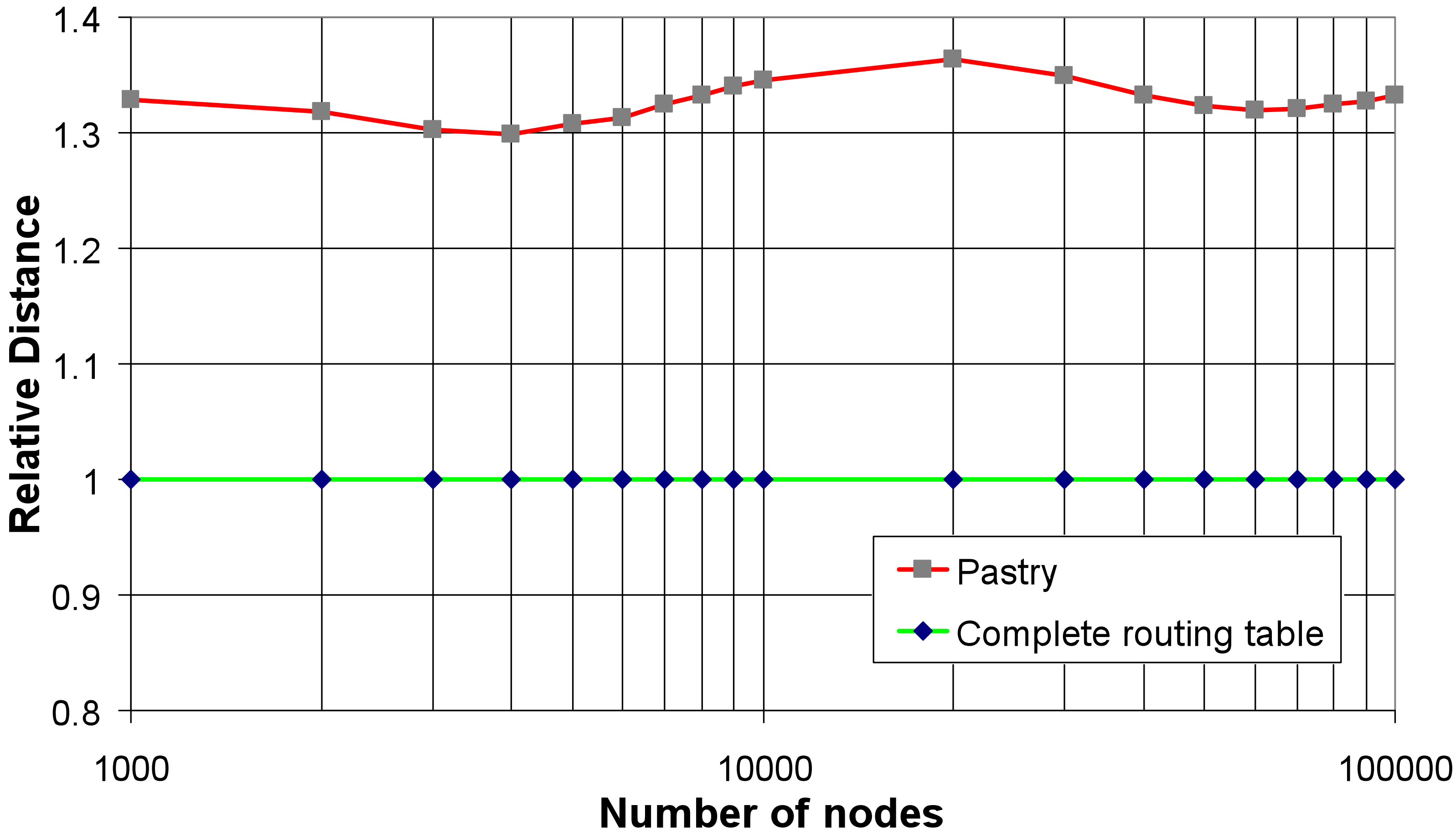
A node can probe distance to any other node

Each routing table entry uses a node close to the local node (in the proximity space), among all nodes with the appropriate node Id prefix.

Pastry: Routes in proximity space

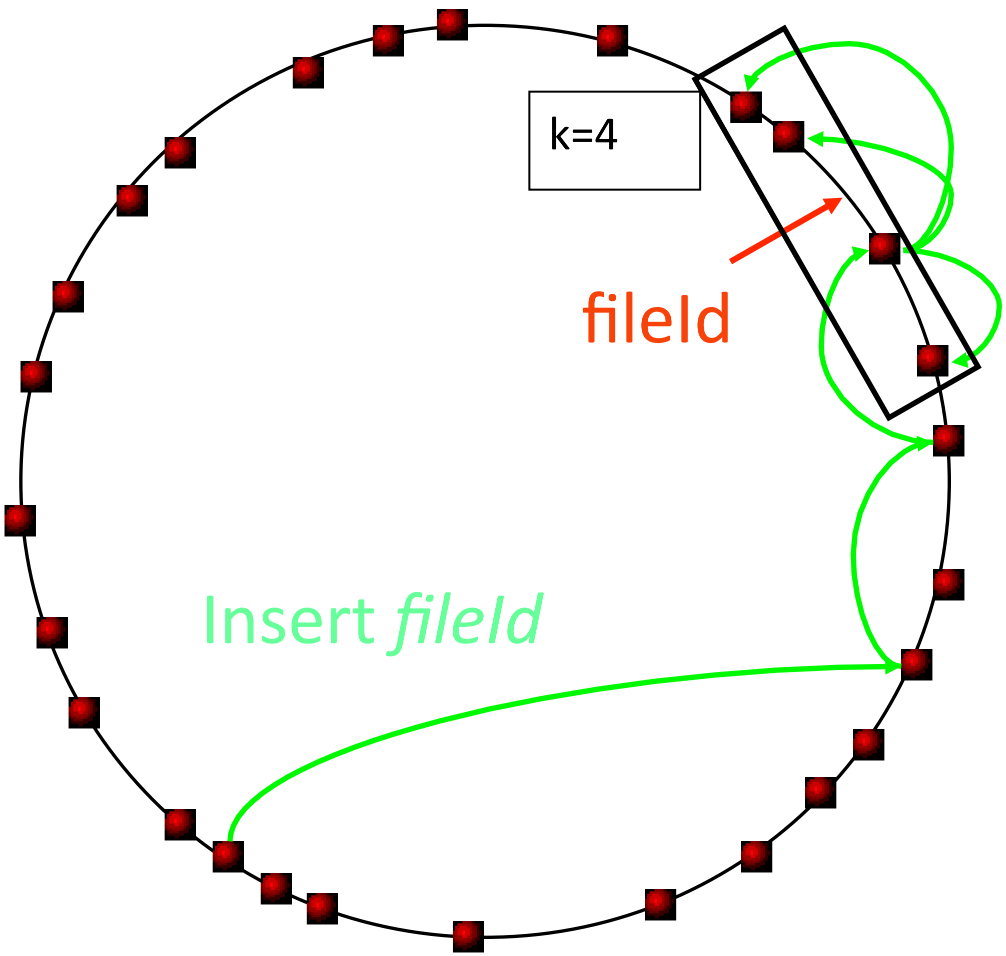


Pastry: Distance traveled



L=16, 100k random queries, Euclidean proximity space

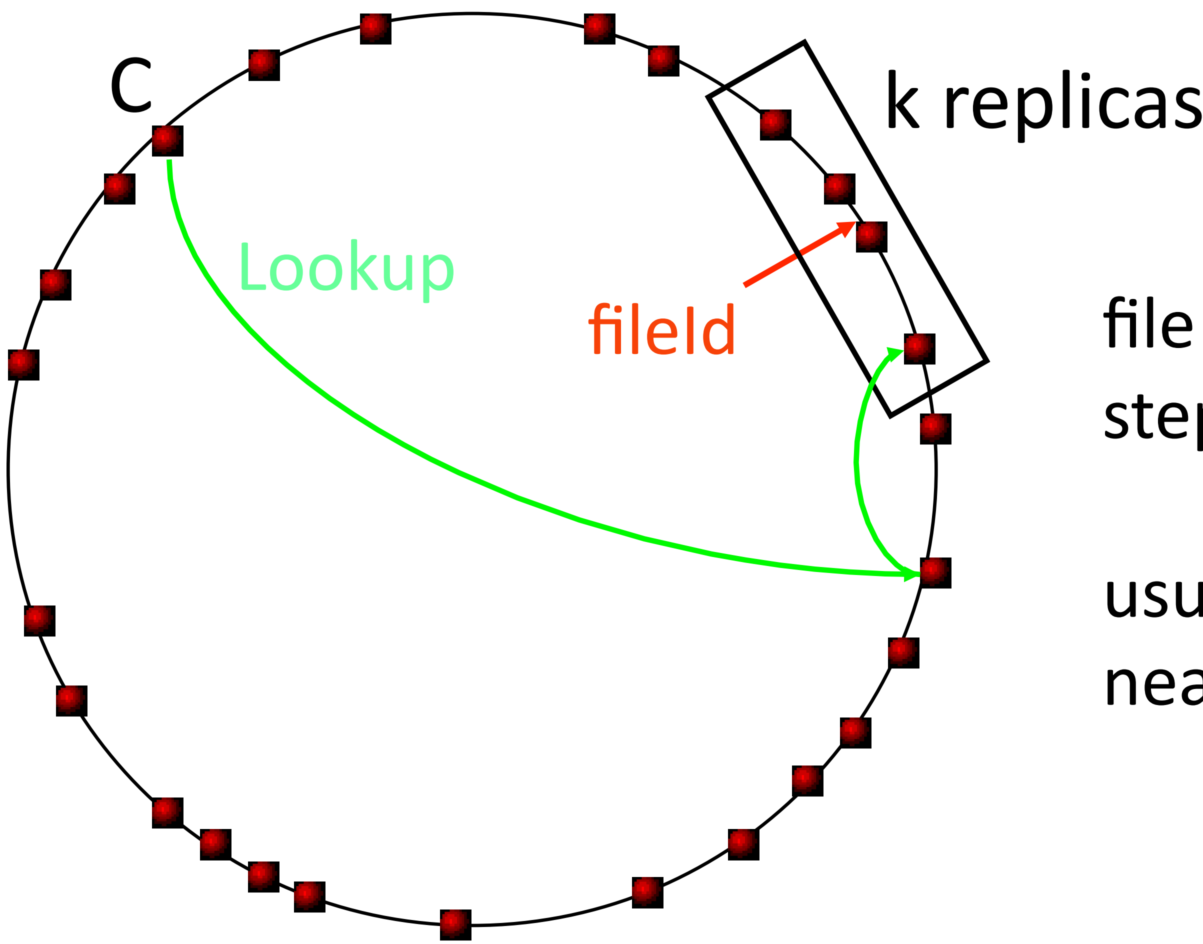
PAST: File storage



Storage Invariant:
File "replicas" are stored on k nodes with nodeIds closest to fileId

(k is bounded by the leaf set size)

PAST: File Retrieval



file located in $\log_{16} N$ steps (expected)

usually locates replica nearest to client C

PAST API

- *Insert* - store replica of a file at *k* diverse storage nodes
- *Lookup* - retrieve file from a nearby live storage node that holds a copy
- *Reclaim* - free storage associated with a file

Files are *immutable*

SCRIBE: Large-scale, decentralized multicast

- Infrastructure to support topic-based publish-subscribe applications
- Scalable: large numbers of topics, subscribers, wide range of subscribers/topic
- Efficient: low delay, low link stress, low node overhead