Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems

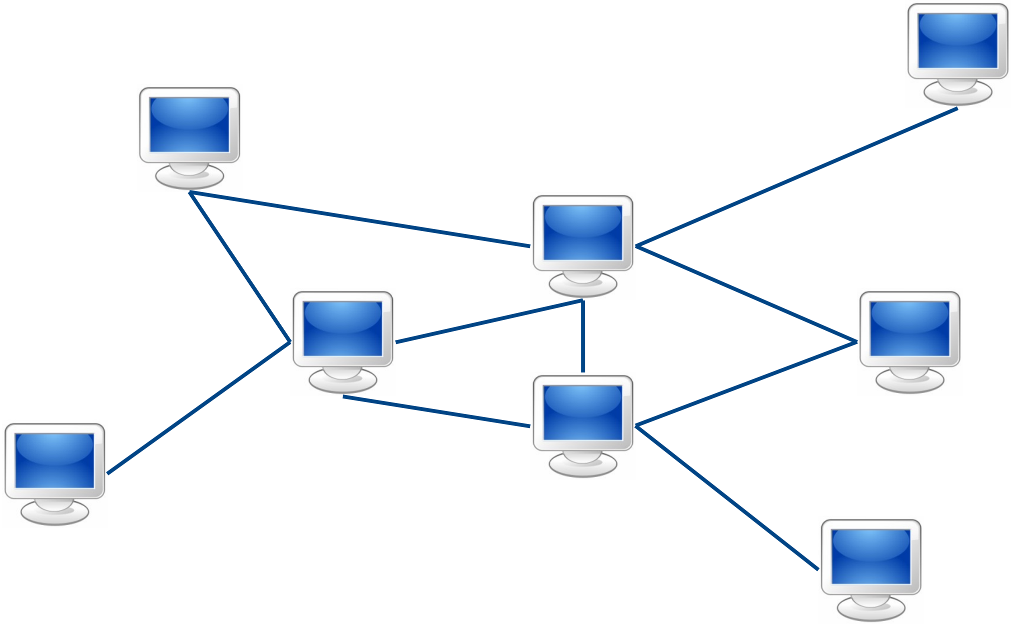
Antony Rowstron - Peter Druschel
Microsoft Research Ltd - Rice University
(IFIP/ACM International Conference on Distributed Systems Platforms - 2001)

Amir H. Payberah (amir@sics.se)
Seif Haridi (haridi@kth.se)

Review

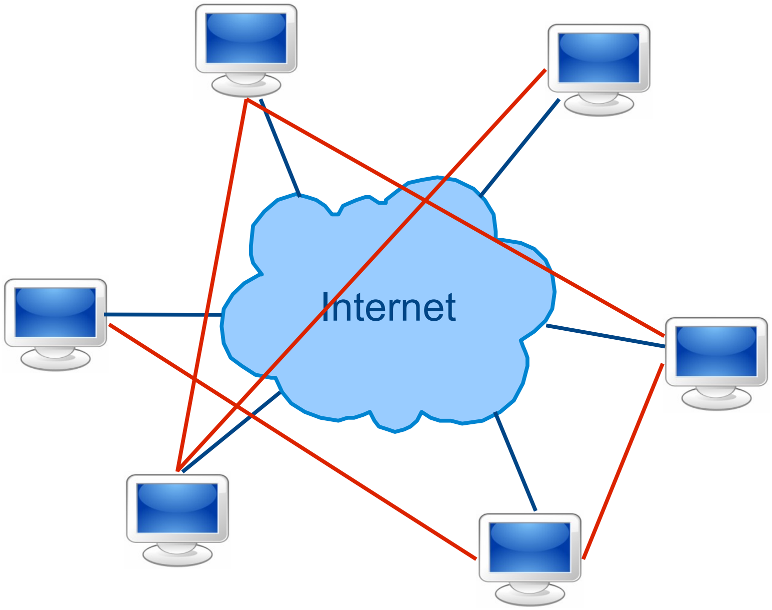
P2P Systems

- Distributed computer system with
  - Symmetric components
  - Decentralized control and state
  - Self-organization



Overlay Networks

- Overlay links rely on unicast service in the Internet.



Key Problem: Object Location

- Objects partitioned among participating nodes.
- Mapping from objects to nodes is dynamic.

- Two solutions:
  - Unstructured overlay
  - Structured overlay

Unstructured vs. Structured

- Unstructured
  - No assumptions about overlay graph structure.
  - Object placement: Inserting node, random walk target or ...
  - Object lookup: Scoped flooding or random walk.
  - Examples: Gnutella, Kazaa, ...

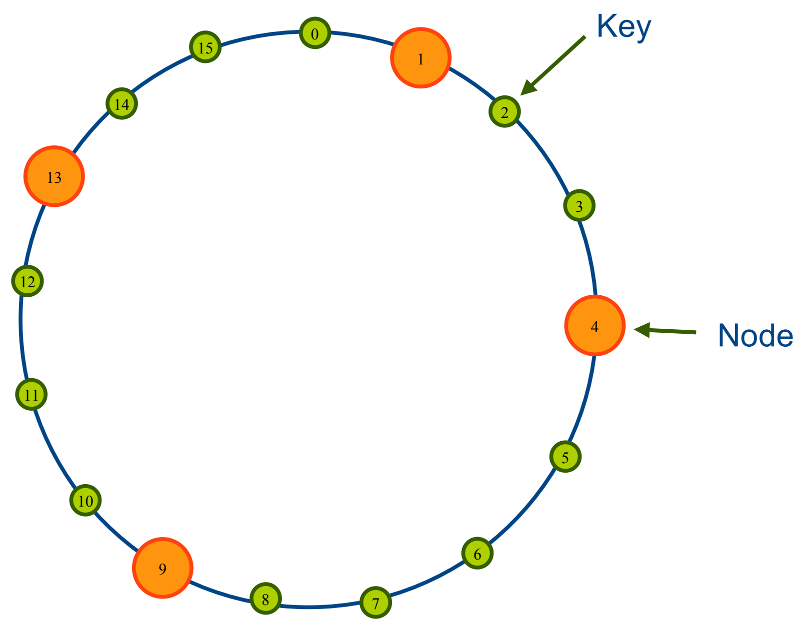
Unstructured vs. Structured

- Unstructured
  - No assumptions about overlay graph structure.
  - Object placement: Inserting node, random walk target or ...
  - Object lookup: Scoped flooding or random walk.
  - Examples: Gnutella, Kazaa, ...
- Structured
  - Overlay graph conforms to a specific graph structure.
  - Key-based routing
  - Examples: Chord, Pastry, Kademlia, SkipNet, ...

Motivation

Consistent Hashing

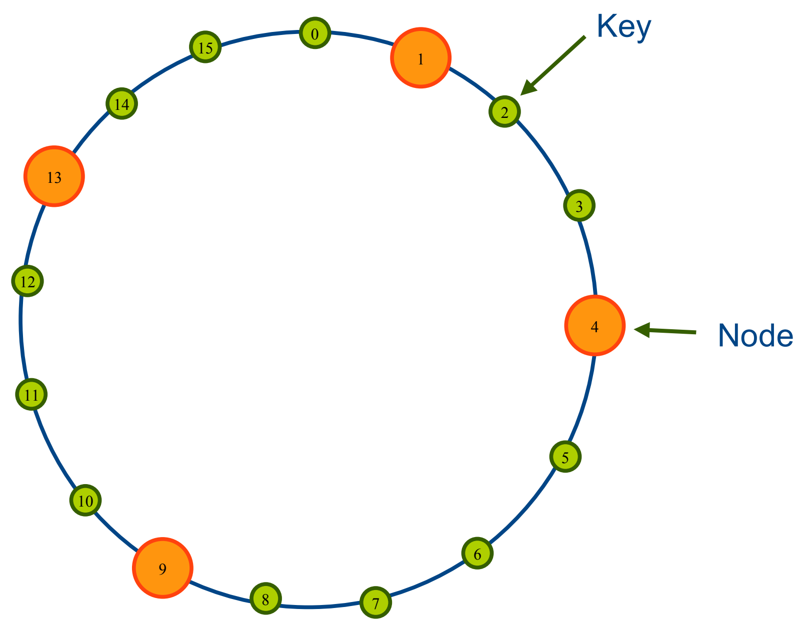
- Both keys and nodes are mapped to the same ID space.
  - Example: Chord



ID Space: [0, 15]

Consistent Hashing

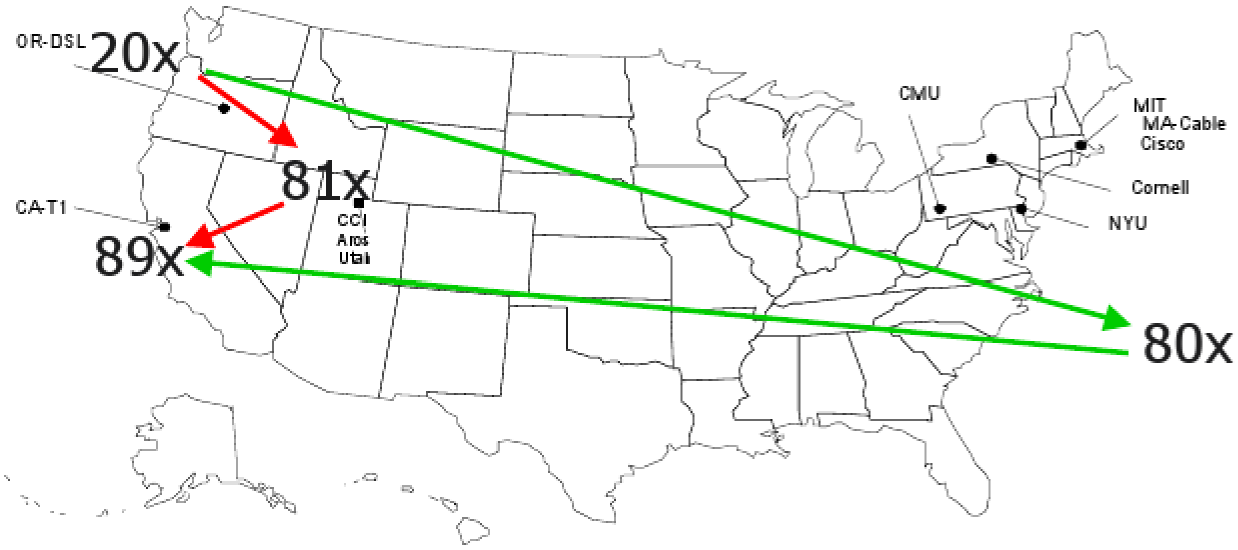
- Both keys and nodes are mapped to the same ID space.
  - Example: Chord



ID Space: [0, 15]

- Result:
  - Uniformly distributed

Challenge: Overlay Route Efficiency



- Nodes close in id space, but far away in Internet.
- Goal: choose routing table entries that yield few hops and low latency.

Plaxton Mesh

Goal

- A set of shared objects in a network.
- Goal: Designing an access scheme that is efficient w.r.t both time and space.

Goal

- A set of shared objects in a network.
- Goal: Designing an access scheme that is efficient w.r.t both time and space.

- What is the challenge here (think about this: if every node stores the location of each object in the system)? [d]

Goal

- A set of shared objects in a network.
- Goal: Designing an access scheme that is efficient w.r.t both time and space.
- What is the challenge here (think about this: if every node stores the location of each object in the system)? [d]
  - Read is fast.
  - Insert and delete are very expensive.
  - Storage overhead grows.

Main Idea

- Map the nodes and objects to b-ary numbers of m digits.
- Assign each object to the node with which it shares the largest prefix.
  - e.g. b = 4 and m = 6:

321002

321302 → 321333

Routing Table

- Level i matches i prefix entries.
- Number of rows = m.
- Number of entries per level = b.

- b = 4, m = 6
- nodeID = 110223

	d = 0	d = 1	d = 2	d = 3
p = 0	0xxxxx	110223	2xxxxx	3xxxxx
p = 1	10xxxx	110223	12xxxx	13xxxx
p = 2	110223	111xxx	112xxx	113xxx
p = 3	1100xx	1101xx	110223	1103xx
p = 4	11020x	11021x	110223	11023x
p = 5	110200	110221	110222	110223

Routing Table

- b = 4, m = 6
- nodeID = 110223

	d = 0	d = 1	d = 2	d = 3
p = 0	023120		212320	300123
p = 1	100233		121100	132121
p = 2		111023	112333	113222
p = 3	110021	110121		110301
p = 4	110203	110213		110233
p = 5	110200	110221	110222	

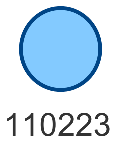
Routing

- At node A find object O:
  - Let the shared prefix of A and O be of length n,
  - Look at level n+1in routing table of A,
  - Find the entry at level n+1 that matches digit n+1of O's id (call it the node B),
  - Send the message to node B,
  - Eventually the message gets relayed to the destination.

- Move closer to the target one digit at the time.

Routing Sample (1/6)

- b = 4, m = 6, nodeID = 110223 → lookup(322010)

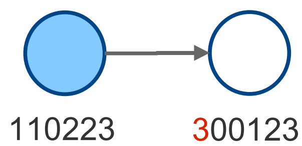


Node 110223 routing table

	d = 0	d = 1	d = 2	d = 3
p = 0				300123
p = 1				
p = 2				
p = 3				
p = 4				
p = 5				

Routing Sample (2/6)

- b = 4, m = 6, nodeID = 110223 → lookup(322010)

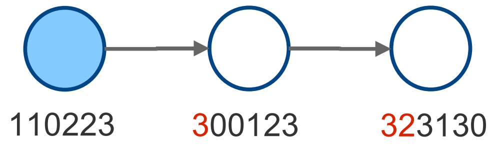


Node 300123 routing table

	d = 0	d = 1	d = 2	d = 3
p = 0				
p = 1			323130	
p = 2				
p = 3				
p = 4				
p = 5				

Routing Sample (3/6)

- b = 4, m = 6, nodeID = 110223 → lookup(322010)

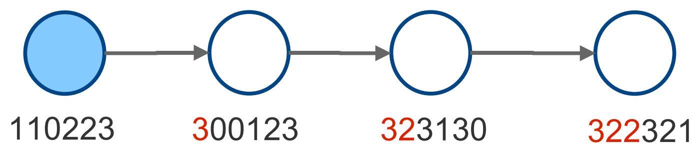


Node 323130 routing table

	d = 0	d = 1	d = 2	d = 3
p = 0				
p = 1				
p = 2			322321	
p = 3				
p = 4				
p = 5				

Routing Sample (4/6)

- b = 4, m = 6, nodeID = 110223 → lookup(322010)

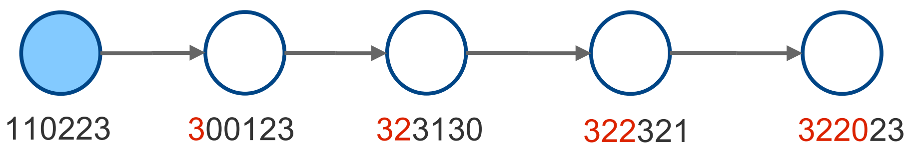


Node 322321 routing table

	d = 0	d = 1	d = 2	d = 3
p = 0				
p = 1				
p = 2				
p = 3	322023			
p = 4				
p = 5				

Routing Sample (5/6)

- b = 4, m = 6, nodeID = 110223 → lookup(322010)

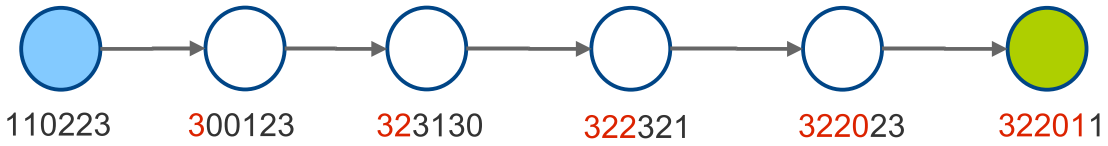


Node 322023 routing table

	d = 0	d = 1	d = 2	d = 3
p = 0				
p = 1				
p = 2				
p = 3				
p = 4		322011		
p = 5				

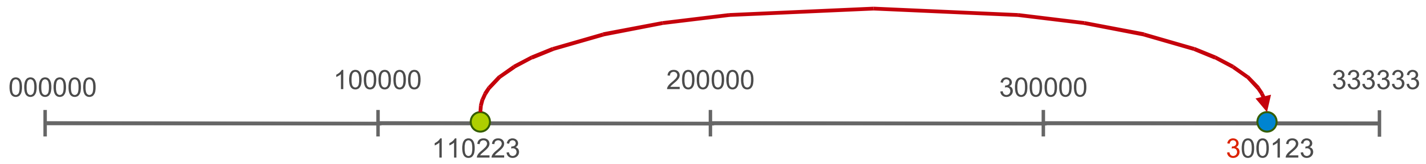
Routing Sample (6/6)

- b = 4, m = 6, nodeID = 110223 → lookup(322010)



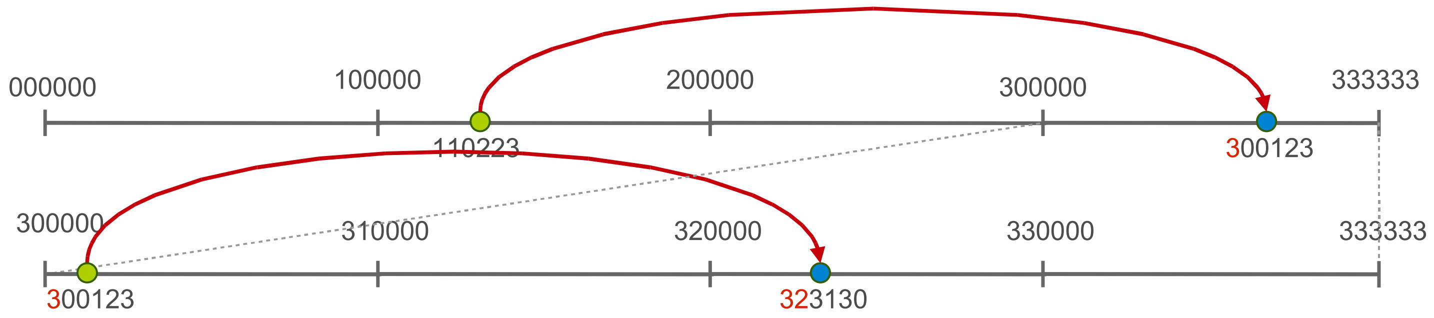
Routing Sample (1/6)

- b = 4, m = 6, nodeID = 110223 → lookup(322010)



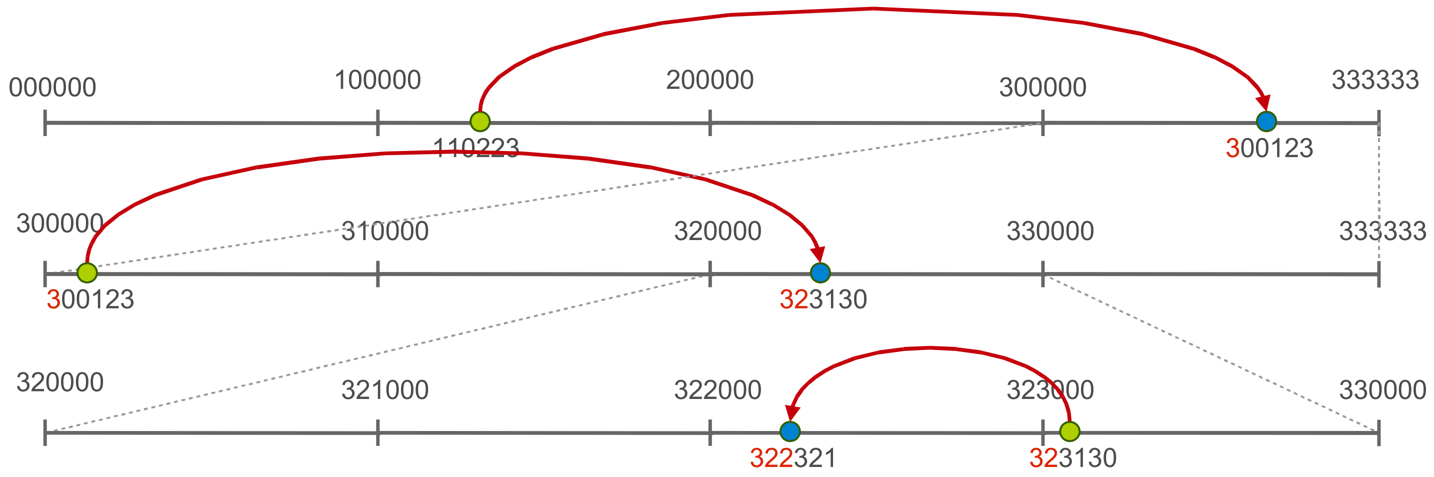
Routing Sample (2/6)

- b = 4, m = 6, nodeID = 110223 → lookup(322010)



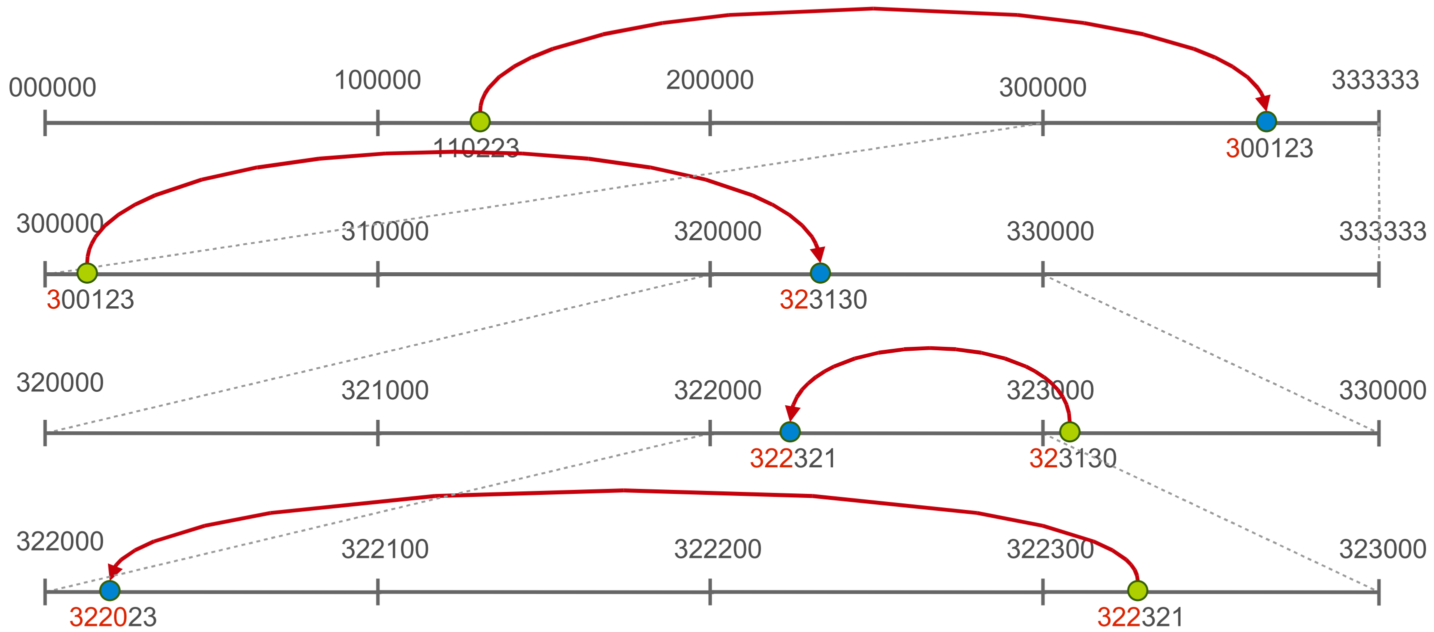
Routing Sample (3/6)

- b = 4, m = 6, nodeID = 110223 → lookup(322010)



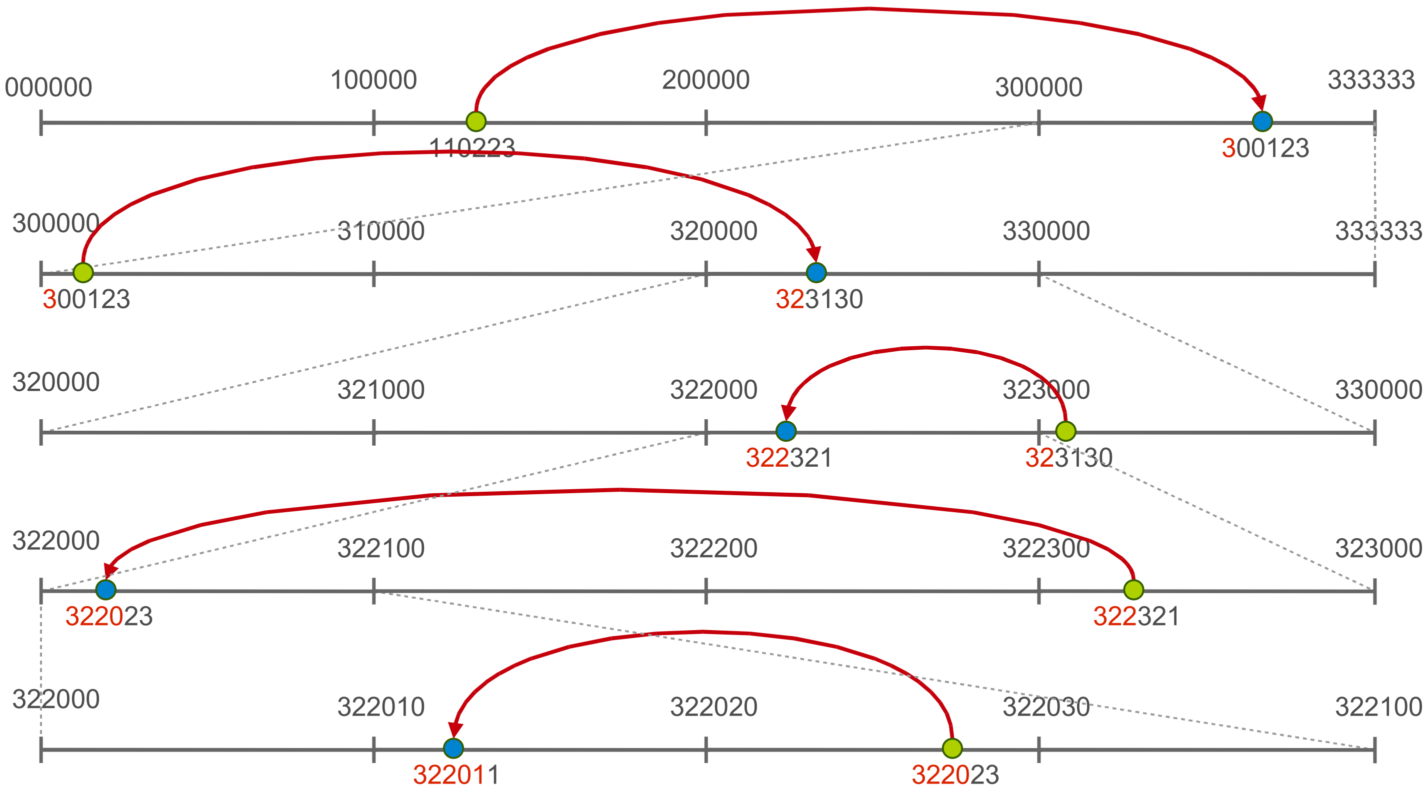
Routing Sample (4/6)

- b = 4, m = 6, nodeID = 110223 → lookup(322010)



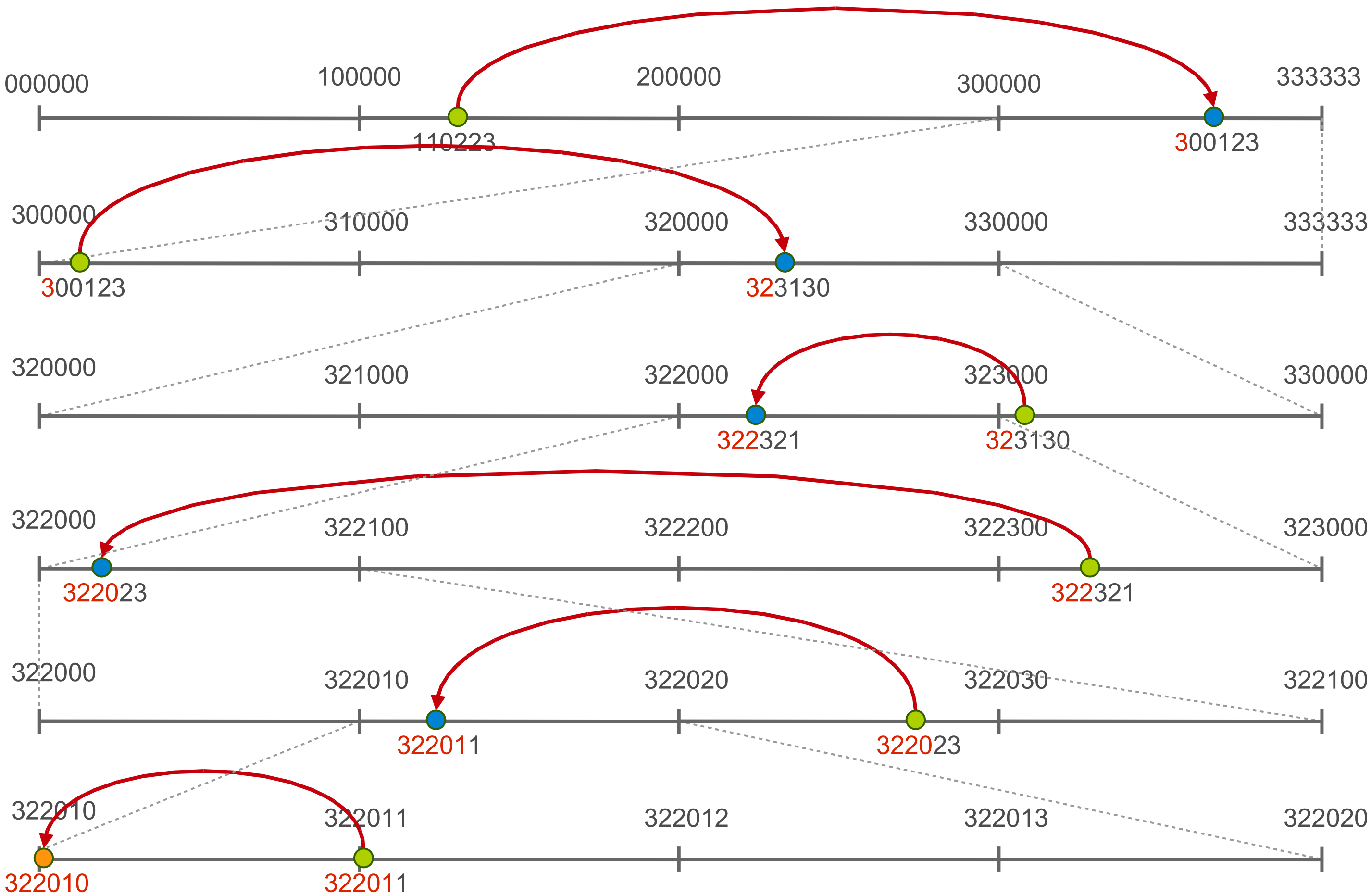
Routing Sample (5/6)

- b = 4, m = 6, nodeID = 110223 → lookup(322010)



Routing Sample (6/6)

- b = 4, m = 6, nodeID = 110223 → lookup(322010)



Pastry

Pastry

- Pastry is an overlay and routing network for the implementation of a distributed hash table similar to Chord.
- Seeks to minimize the distance messages travel.
- Expected number of routing steps is O(log N).
  - N = No. of Pastry nodes in the network.

Routing Table

Leaf set

SMALLER		LARGER	
10233033	10233021	10233120	10233122
10233001	10233000	10233230	10233232

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	

Neighborhood set

13021022	10200230	11301233	31301233
02212102	22301203	31203203	33213321

Leaf set: L/2 numerically closest nodes

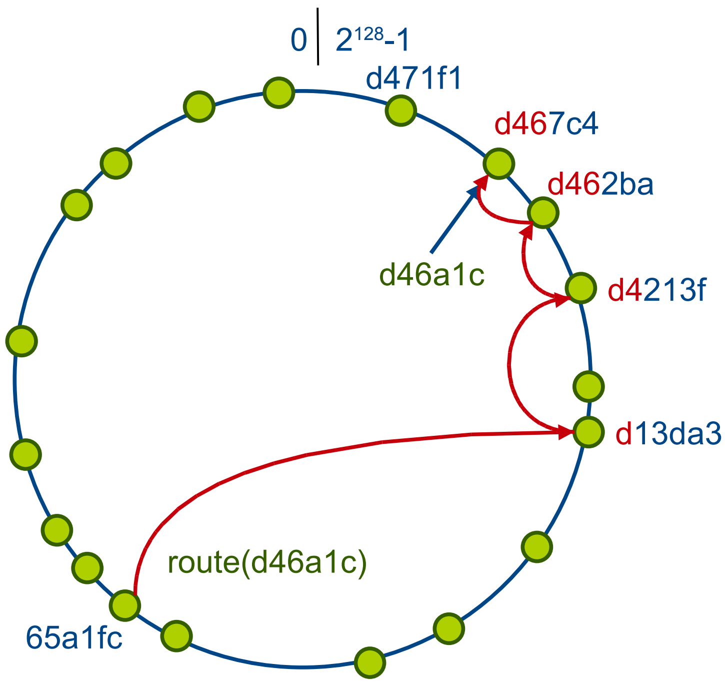
Routing table: prefix-based

Neighbor set: M physically closest nodes

Routing

- If message with key D is within range of leaf set, forward to numerically closest leaf.
- Else forward to node that shares at least one more digit with D in its prefix than current nodeId.
- If no such node exists, forward to node that shares at least as many digits with D as current nodeId but numerically nearer than current nodeId.

Routing Example

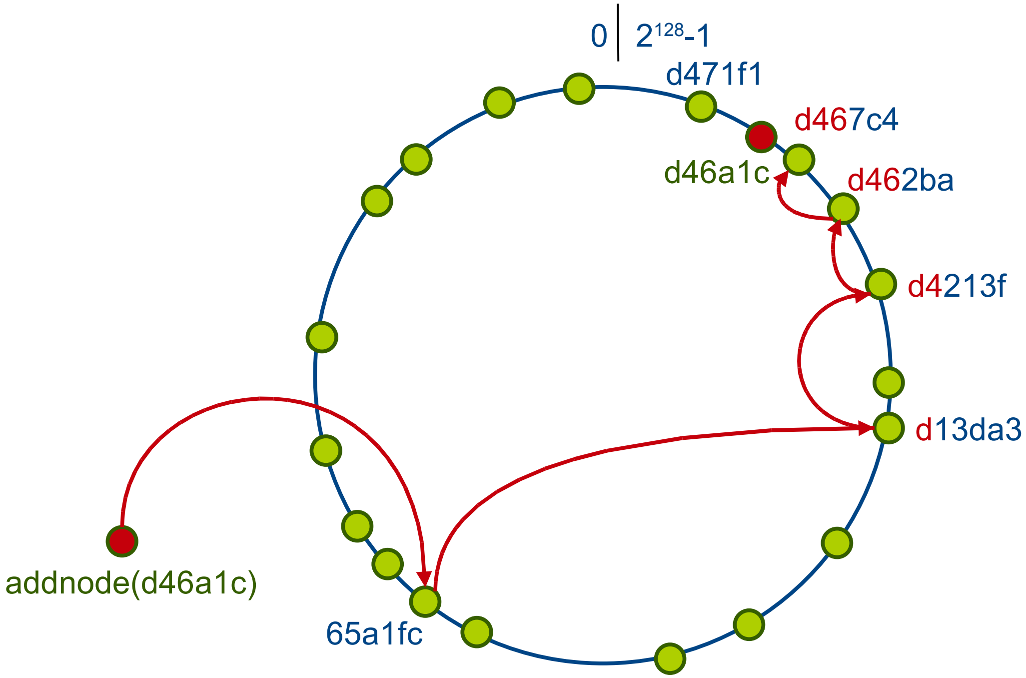


ID Space: [0, 2^{128}-1]

Joining

- Assume the node ID is X, and initially it knows a physically nearby node A in the system.
- X sends A a join message with the key equal to X's ID.
- Pastry routes the message to a node B whose ID is numerically closest to X's ID.
- The message follows a paths through nodes A, B, etc., and eventually reaches node Z.

Joining Example



ID Space: $[0, 2^{128}-1]$

Joining: First Routing Table

- The neighborhood-set is set to that of the first node, A.
- The leaf-set is set to that of the final node, Z.
- The routing table is collected from each node along the route.
  - The i^{th} node on the route should send its i^{th} row of routing table to the new node.

Leaf set	SMALLER	LARGER	
10233033	10233021	10233120	10233122
10233001	10233000	10233230	10233232

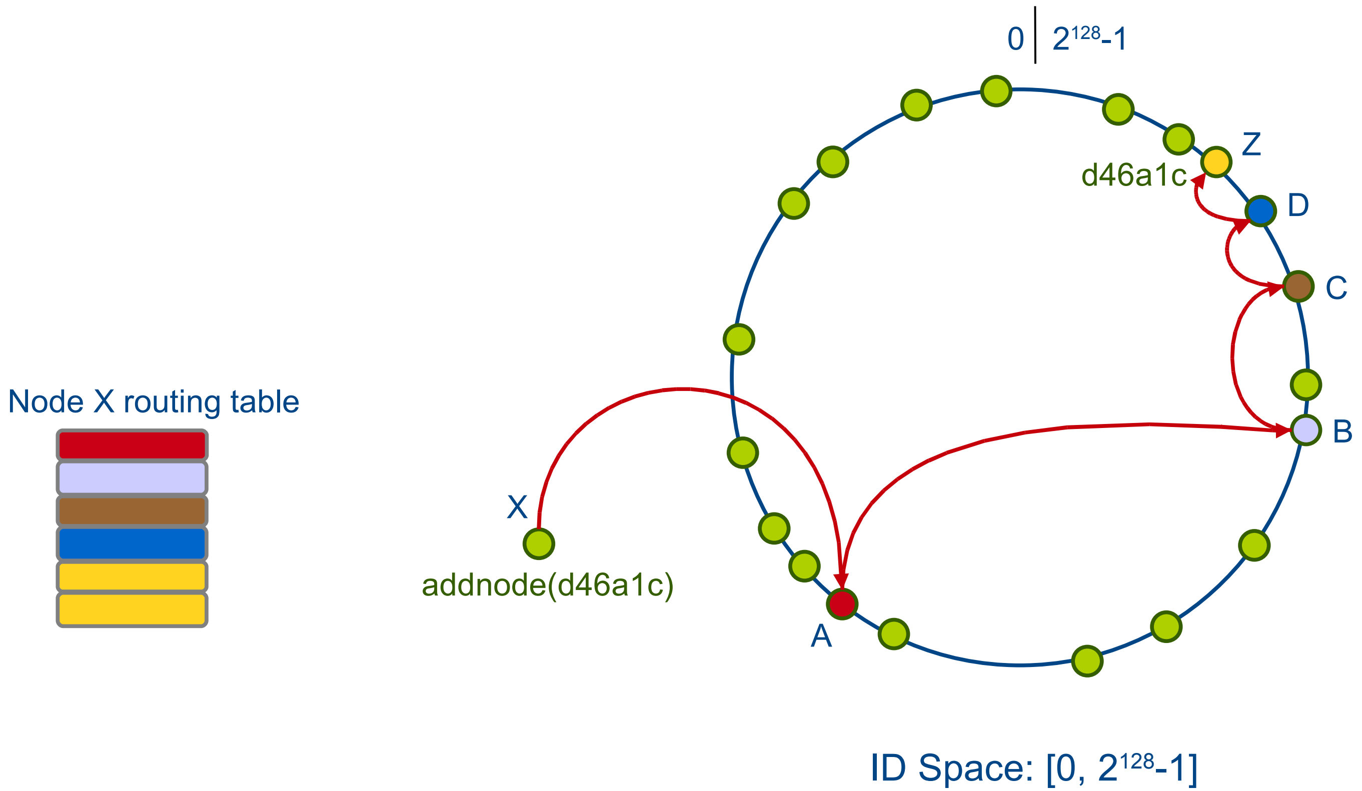
Routing table

-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	

Neighborhood set

13021022	10200230	11301233	31301233
02212102	22301203	31203203	33213321

First Routing Table Example



Failure

- Failure in leaf set (LS):
  - Detected by heartbeat
  - Repair by inserting node from another leaf's (LS).

Failure

- Failure in leaf set (LS):
  - Detected by heartbeat
  - Repair by inserting node from another leaf's (LS).
- Failure in neighborhood set (NS):
  - Detected by heartbeat
  - Query all NS members for their NS tables, choose replacement according to proximity metric.

Failure

- Failure in leaf set (LS):
  - Detected by heartbeat
  - Repair by inserting node from another leaf's (LS).
- Failure in neighborhood set (NS):
  - Detected by heartbeat
  - Query all NS members for their NS tables, choose replacement according to proximity metric.
- Failure int routing table (RT):
  - Entries detected when attempting to route
  - Query nodes in row for replacement entry, if failed
  - Query successive rows until success.

Failure Example

	d = 0	d = 1	d = 2	d = 3
p = 0	023120	1	212320	300123
p = 1	100233	1	121100	132121
p = 2	0	111023	112333	113222
p = 3	110021	110121	2	110301
p = 4	110203	110213	2	110233
p = 5	110200	110221	110222	3

1. Try asking some neighbors in the same row for its 111xxx entry.

Node in red fails.

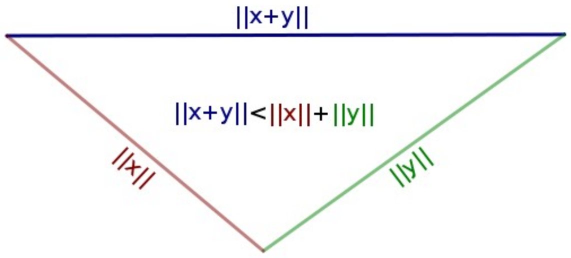
2. If it doesn't have one, try asking some neighbor in the row below, etc.

So, what about locality?

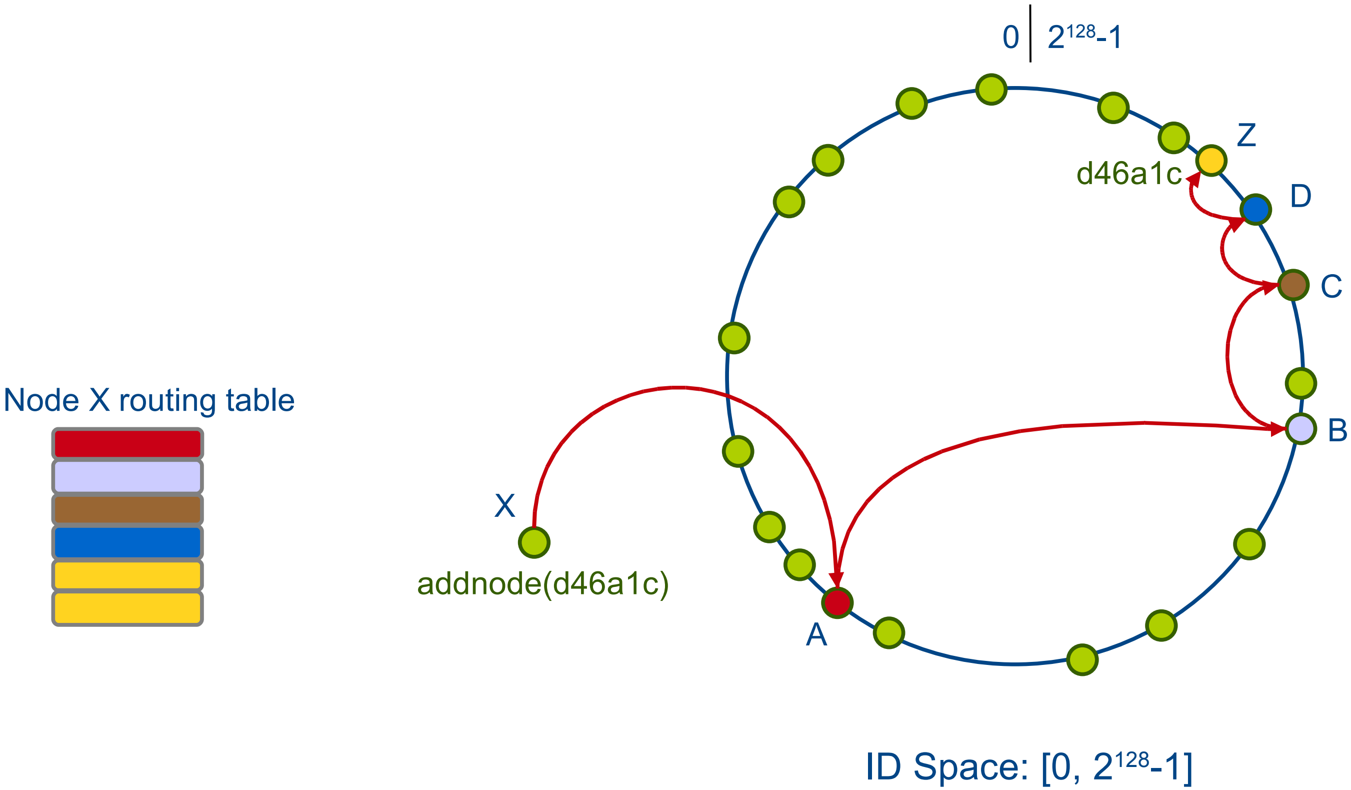
Locality

- Pastry's notion of network proximity is based on a scalar proximity metric.
  - Number of IP routing hops
  - Geographic distance

- We assume that the proximity space defined by the chosen proximity metric is Euclidean; that is, the triangulation inequality holds for distances among Pastry nodes.



Locality in Routing Table



- The property we wish to maintain is that all routing table entries refer to a node that is near the present node.

Locality in Routing Table

- We assume that this property holds prior to node X's joining the system, and show how we can maintains the property as node joins.

Routing Table – Row 0

- First, we require that node A is near X, according to the proximity metric.
- Since the entries in row zero of A's routing table are close to A, and A is close to X, and we assume that the triangulation inequality holds in the proximity space, it follows that the entries are relatively near A. Therefore, the desired property is preserved.

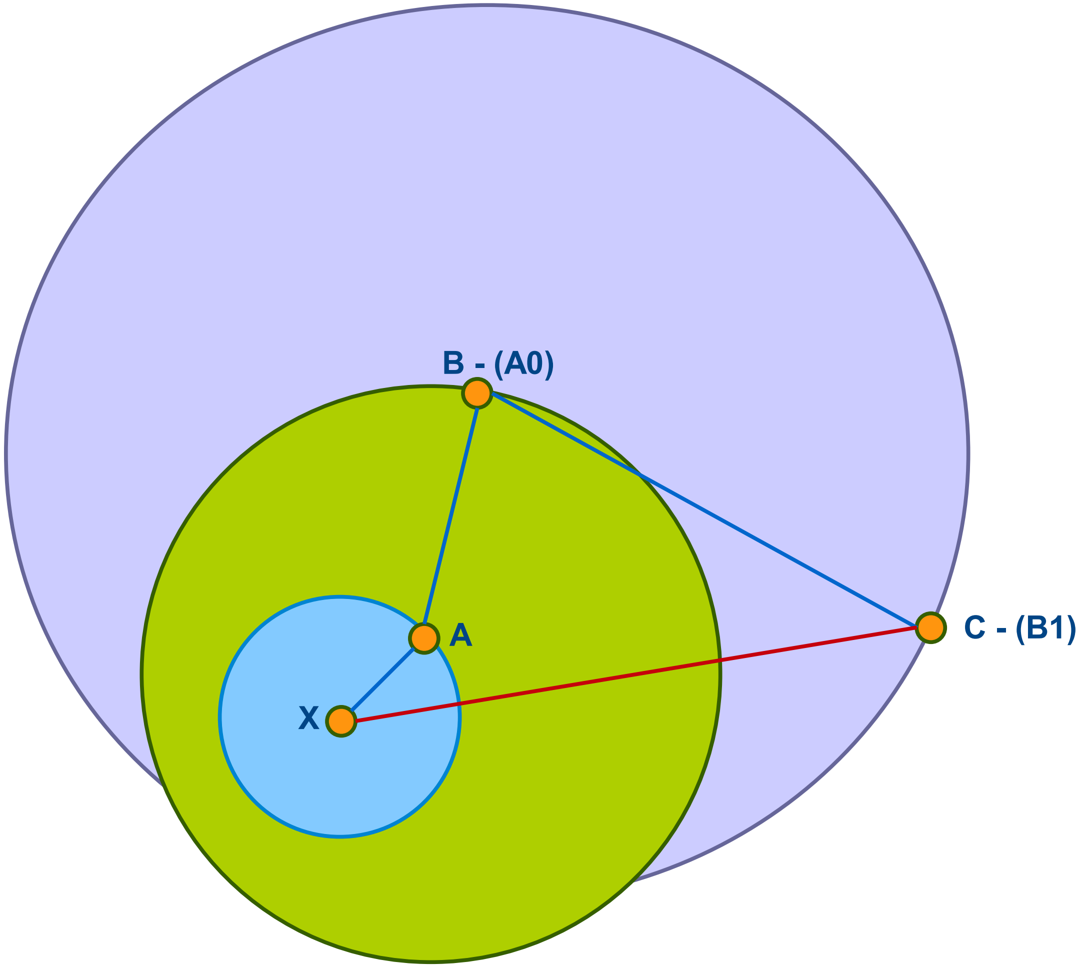
Routing Table – Row 1

- The entries in this row are near B, however, it is not clear how close B is to X.
- Take row one of X's routing table from node B does not preserve the desired property, since the entries are close to B, but not necessarily to X. [d]

Routing Table – Row 1

- For larger row numbers the number of possible choices decreases exponentially.
- For larger row numbers the expected distance to the nearest neighbor increases exponentially.
- Therefore, the expected distance from B to its row one entries (B1) is much larger than the expected distance traveled from node A to B.

Locality



Pastry and other DHTs

Pastry vs. Chord

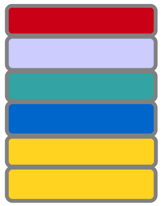
- A Plaxton tree combined with a Chord-like ring
  - Each node has Plaxton-style neighbors.
  - Each node knows its predecessor and successor.
    - Called its leaf set
- like Chord
  - Only leaf set necessary for correctness.
  - Plaxton-neighbors like finger table, only for performance.
- Unlike Chord
  - Chord's metric is asymmetry, while Pastry's metric is symmetry.

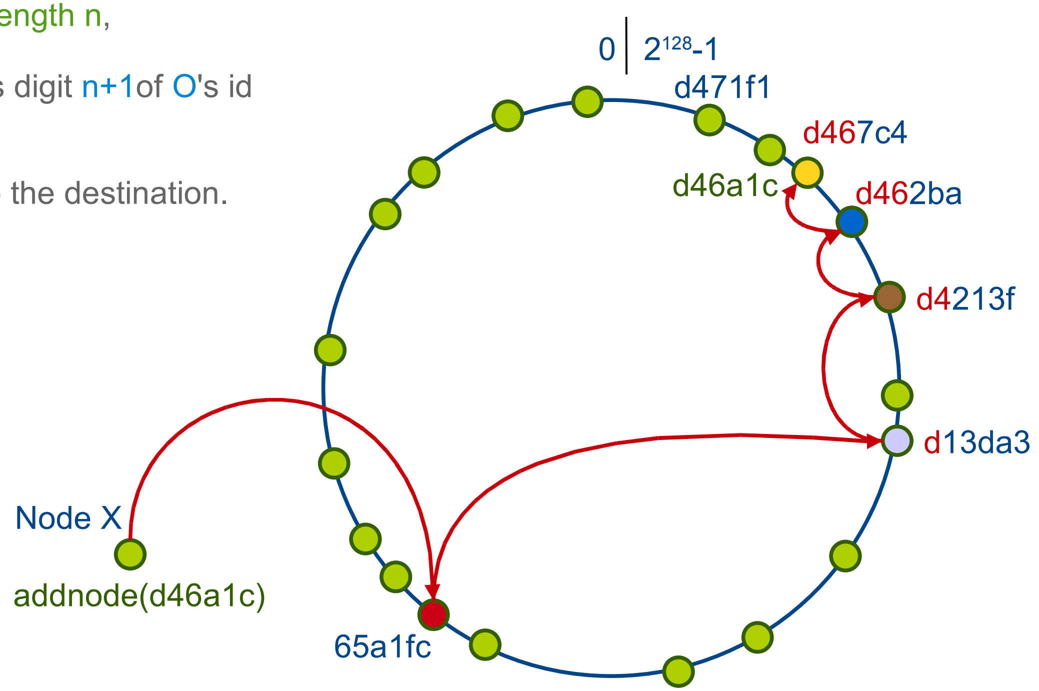
Done!

A Page To Remember

- At node A find object O:
  - Let the shared prefix of A and O be of length n,
  - Look at level n+1in routing table of A,
  - Find the entry at level n+1 that matches digit n+1of O's id (call it the node B),
  - Send the message to node B,
  - Eventually the message gets relayed to the destination.

Node X routing table





ID Space: [0, 2^{128}-1]

References

- [1] C. G. Plaxton, R. Rajaraman, and A. W. Richa. "*Accessing nearby copies of replicated objects in a distributed environment*". In Proceedings of the 9th Annual ACM Symposium on Parallel Algorithms and Architectures, Newport, Rhode Island, pages 311-320, June 1997.

- [2] A. Rowstron and P. Druschel, "*Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems*". IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), Heidelberg, Germany, pages 329-350, November, 2001.

Question?