

Overlays and Distributed Hash Tables

presented by:
Anthony Cozzie
Esteban Meneses

Resilient Overlay Networks

David Anderson, Hari Balakrishnan, Frans
Kaashoek, Robert Morris (MIT)

SOSP 2001

Internet Architecture

- Collection of Autonomous Systems (routers)
- Path information shared via BGP
- Scalability Problem: 200,000 routes!
 - Every node knows all the routes, so only the simplest information is available
 - BGP changes routes slowly to avoid “route flapping”
 - Hides details of topologies and traffic flows
- Internet has redundancy, but scaling problems prevent BGP from giving each router enough information to exploit path diversity

Result: Traffic Outages

- Availability
 - 35% of all routes < 99.99%
 - 10% routes < 95%
 - 40% of all path outages are >30 minutes
 - 5% greater than 2 hours
- Recent: Undersea cables officially cut by a ship's anchor (Terrorists? CIA? Martians?) leave the Middle East without the internet
- Worse: Application with custom requirements

RON: Resilient Overlay Network

- Resilient: Goal is reliability
- Overlay: Network on top of the internet
- Network: you're on your own here
- Basic idea: if the direct path from from Berkeley to MIT is slow, forward packets through UCSD.
 - Intent is that RON nodes are routers, but could be ordinary computers
 - RON is an end-to-end solution, so packets are simply wrapped and sent normally
 - Triangle inequality doesn't hold in the Internet.

Measuring Link State

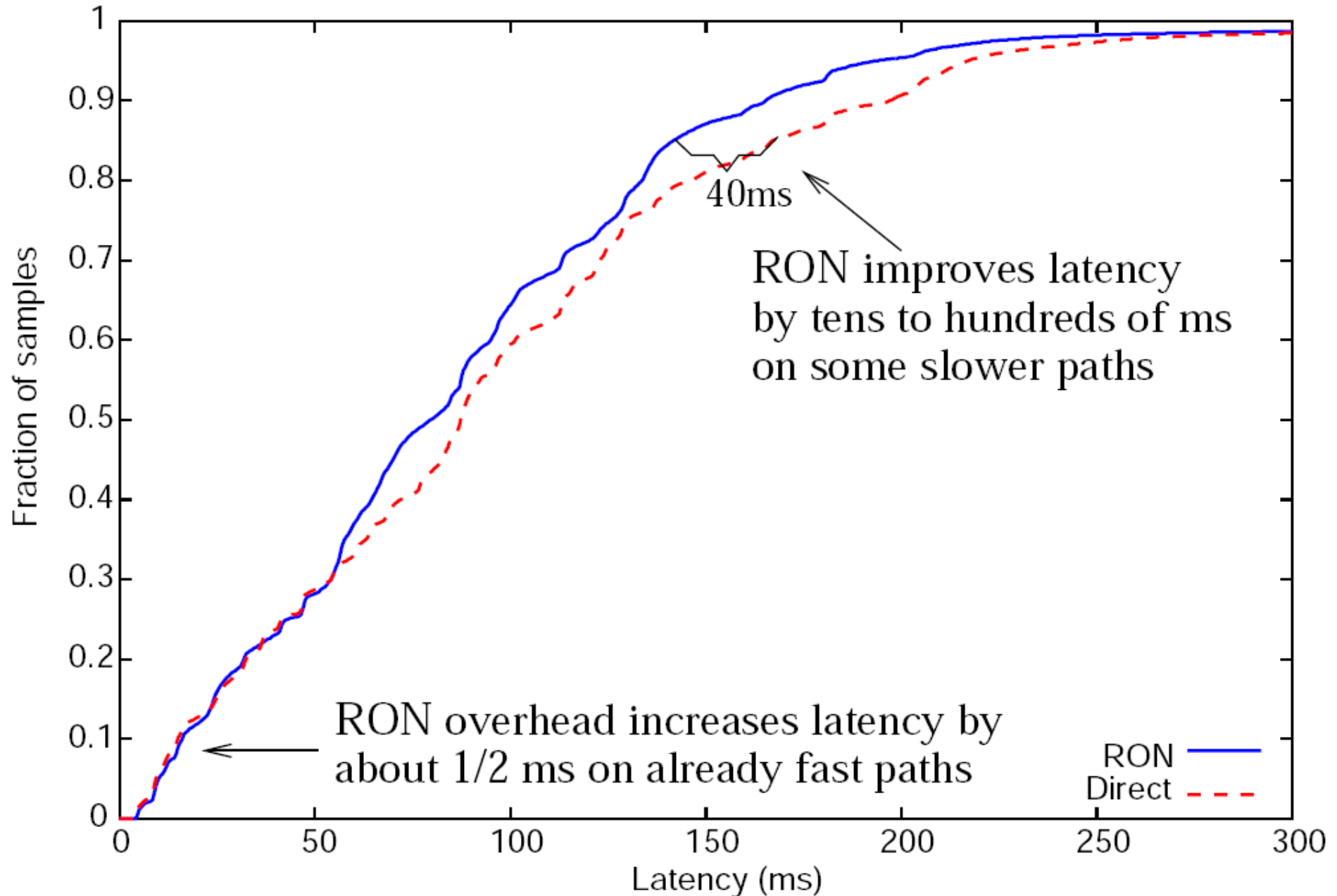
- RONS are small (<50 nodes) so can be fully connected
- Link state is actively measured and averaged
 - Latency, packet loss rate
 - Active → pings, extra traffic, scalability
- These states are broadcast throughout the RON; each node then has enough information to compute an optimal path with given QoS
- **Contrast with BGP: all paths vs 1 path**

Test Setup



- 16 sites across US and Europe
- Mostly universities and dot-coms

Latency: RON vs Internet



Results Summary

- Reduced outages by a factor of 5-10
 - Routed around nearly all complete outages
 - Even fixed nearly all periods of high loss rates
- Key: An average of only 18 seconds (compared to minutes or even hours for BGP)
- As an overlay, easy to deploy and test – a practical solution
- Seems impressive!

Discussion (I): Control Theory

- TCP employs additive increase/multiplicative decrease
 - Ensures fairness and good performance under dedegration
- *RON never backs off*
 - What happens if everyone uses RON?
 - Is RON “antisocial?” Do we care?
 - Is it possible to design a version of RON with provably good global behavior?

Discussion (II): Scalability

- RON's link state monitoring techniques are obvious, but they don't work for BGP due to issues of scale
- What about heterogeneous routing algorithms?
- Is it possible to design a version of BGP that is unaware of some routes?
 - Failure Carrying Packets
 - Most users only access a very small fraction of the web

The Ultimate Verdict



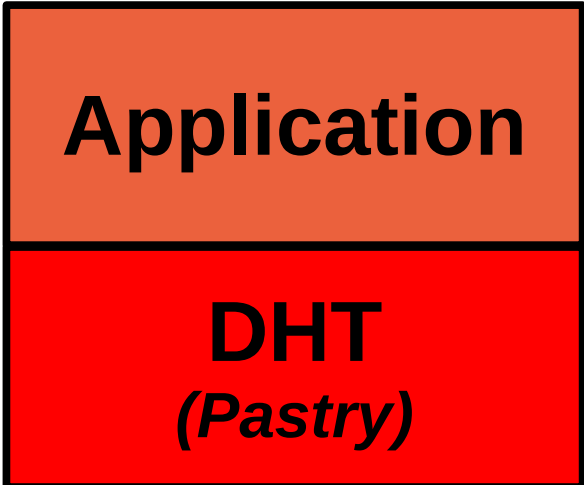
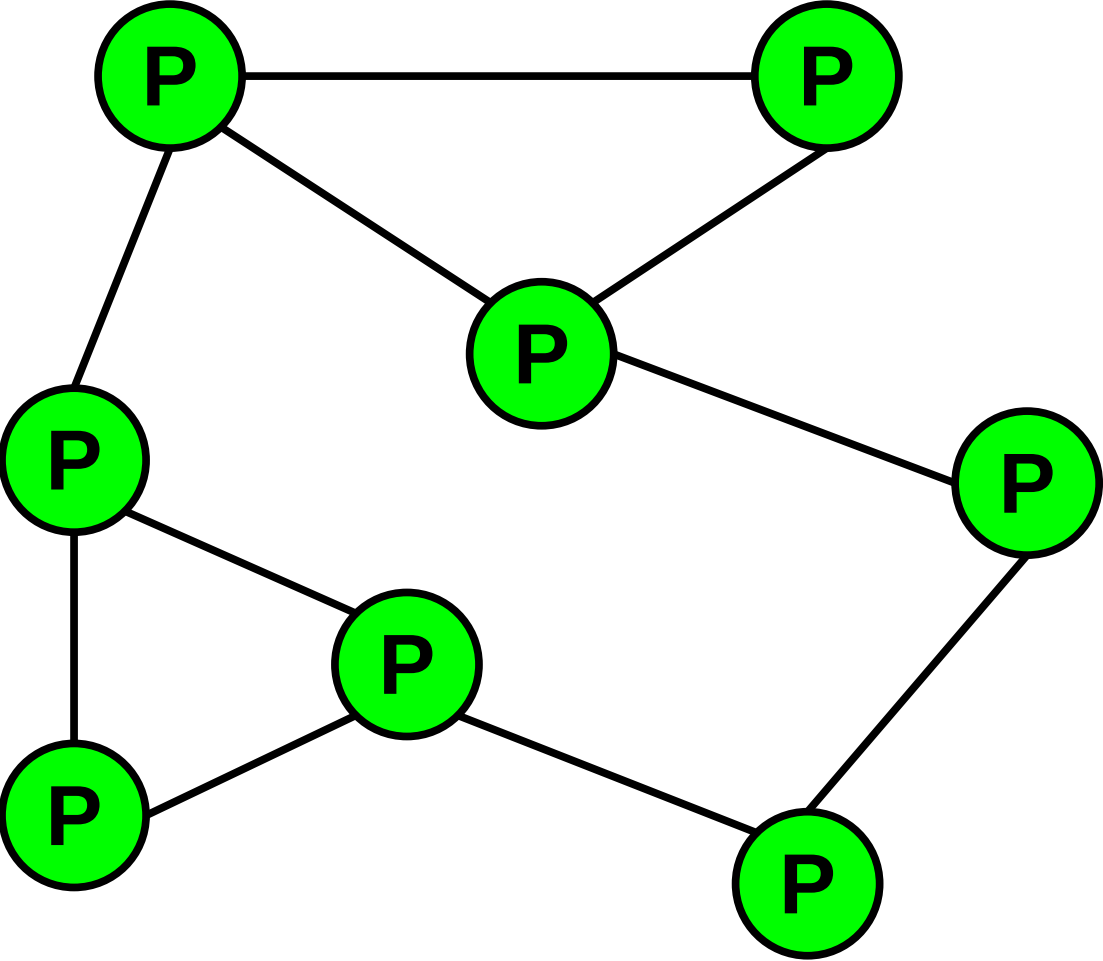
- Would you want your ISP to use a RON?
 - And with whom?
- Would you want your computer as part of a RON?
 - And with whom?

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

Antony Rowstron (Microsoft Research)
Peter Druschel (Rice)

Middleware 2001

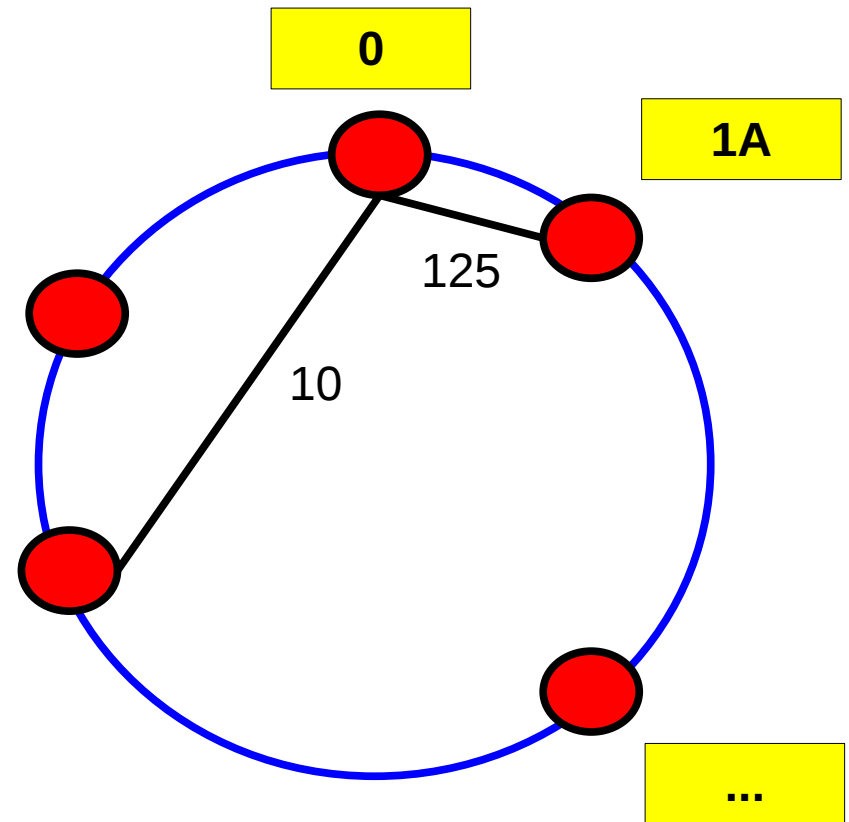
Motivation



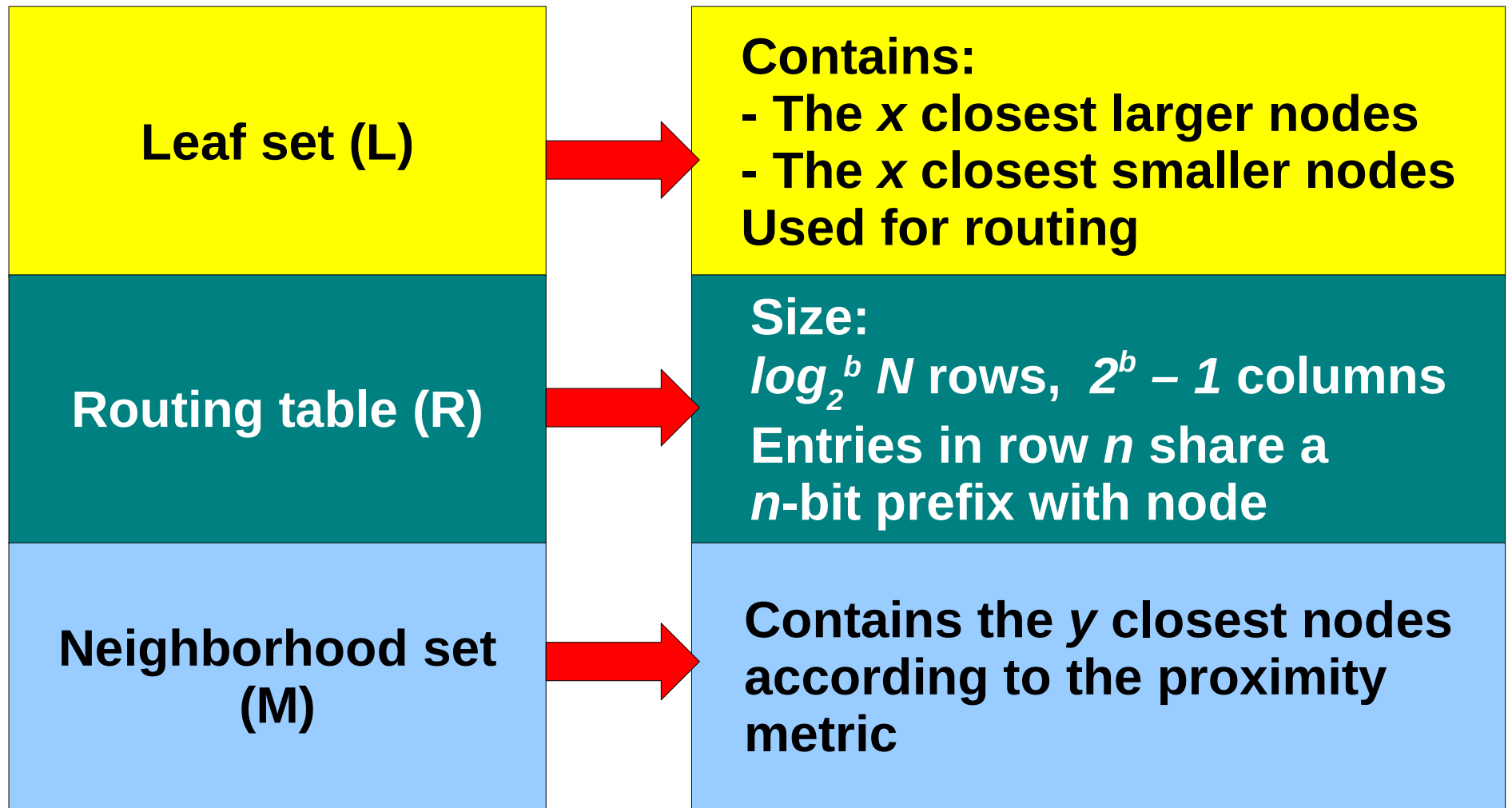
SCRIBE – PAST – SQUIRREL – SplitStream – POST - Scrivener

Design of Pastry

- NodeId = 128-bit (position in a circular space).
- Adjacent nodes are likely diverse.
- Route to closest NodeId in less than $\log_{2^b} N$ steps ($b=4$).
- Prefix-based search.



Node State



Example

(8-bit Ids, b=2, N=64, base 4)

ID=3112

KEY=3211

Leaf set

<i>smaller</i>	<i>smaller</i>	<i>larger</i>	<i>larger</i>
3010	3100	3121	3133

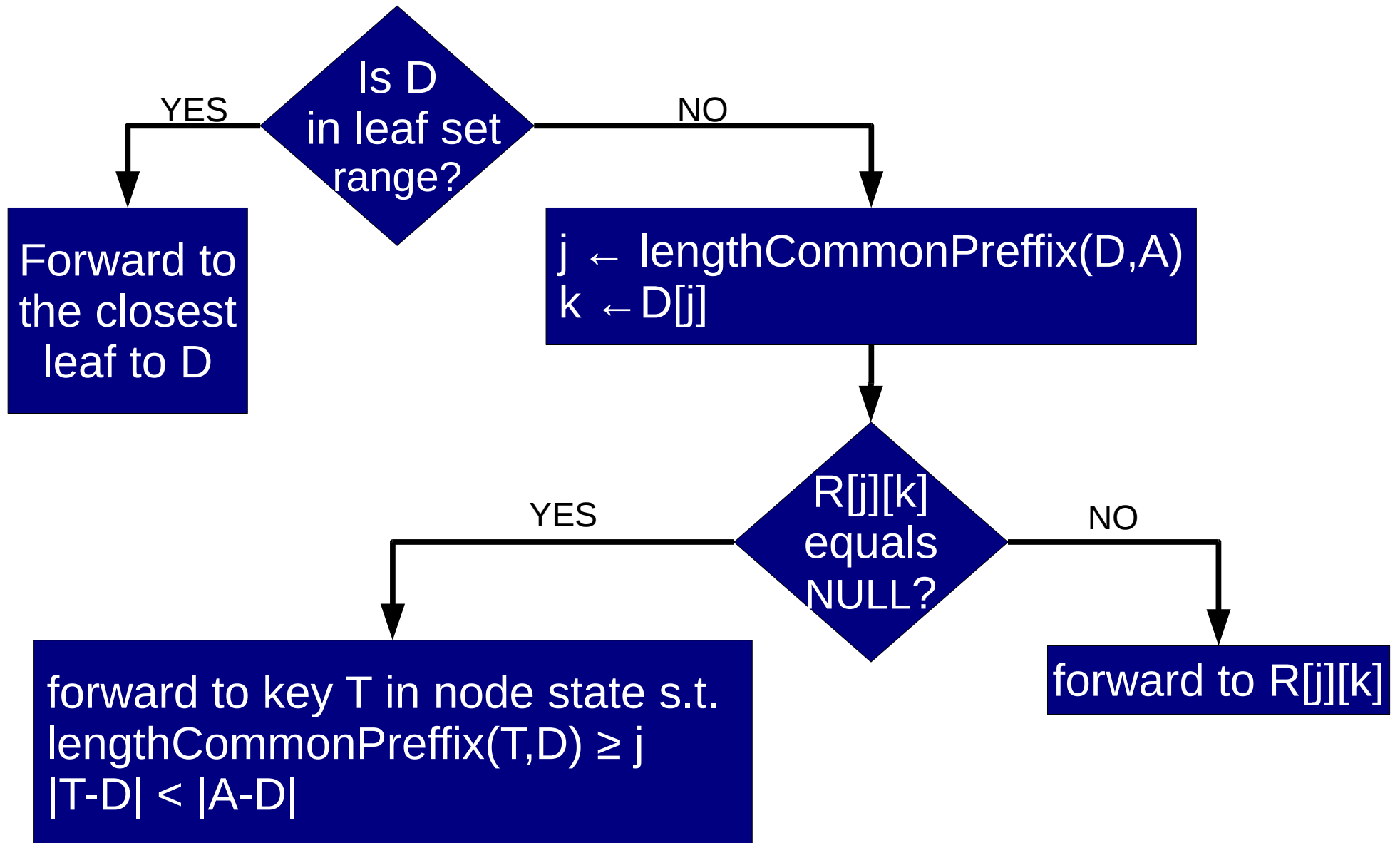
Routing table

0213	1221	2312	
3001		3212	3322
3100		3121	3133

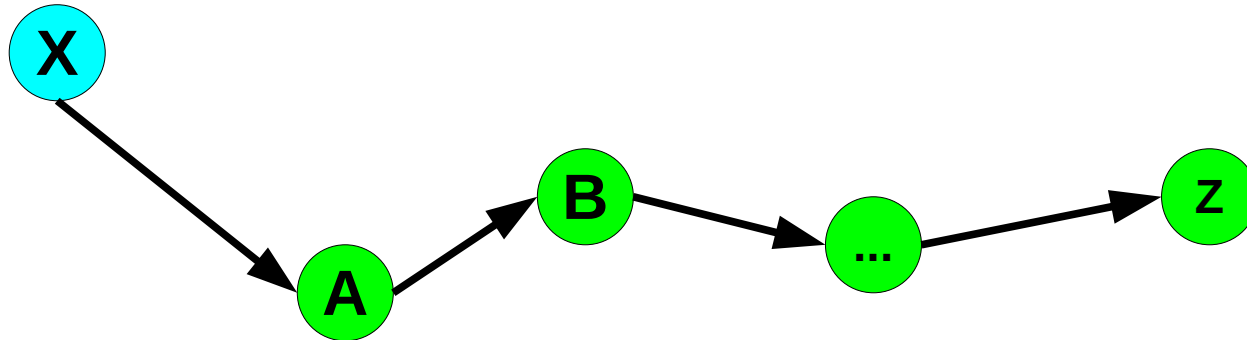
Neighborhood set

1221	2312	3100	3333
-------------	-------------	-------------	-------------

Routing key D in node A



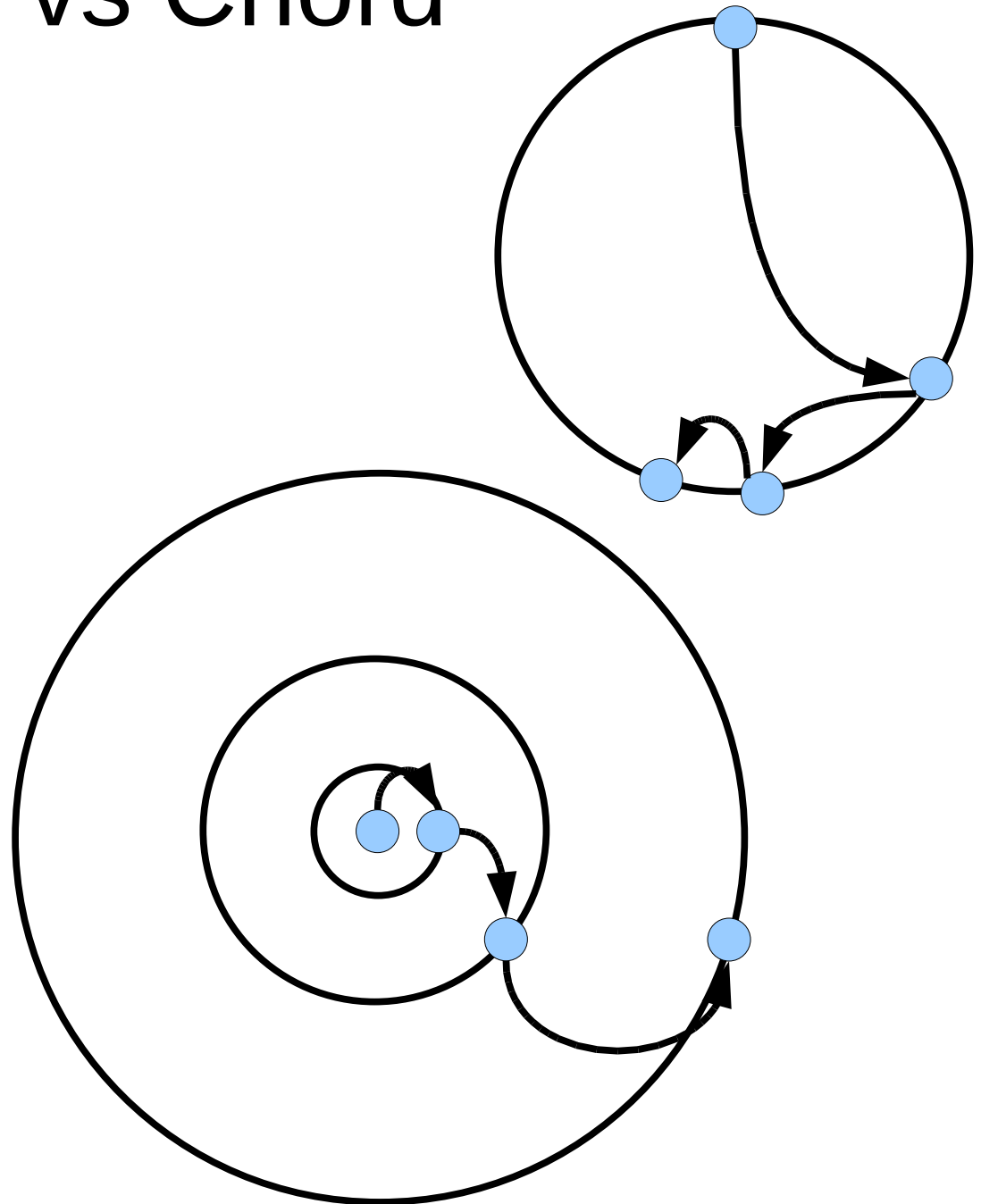
Keeping Locality (Arrival of Node X)



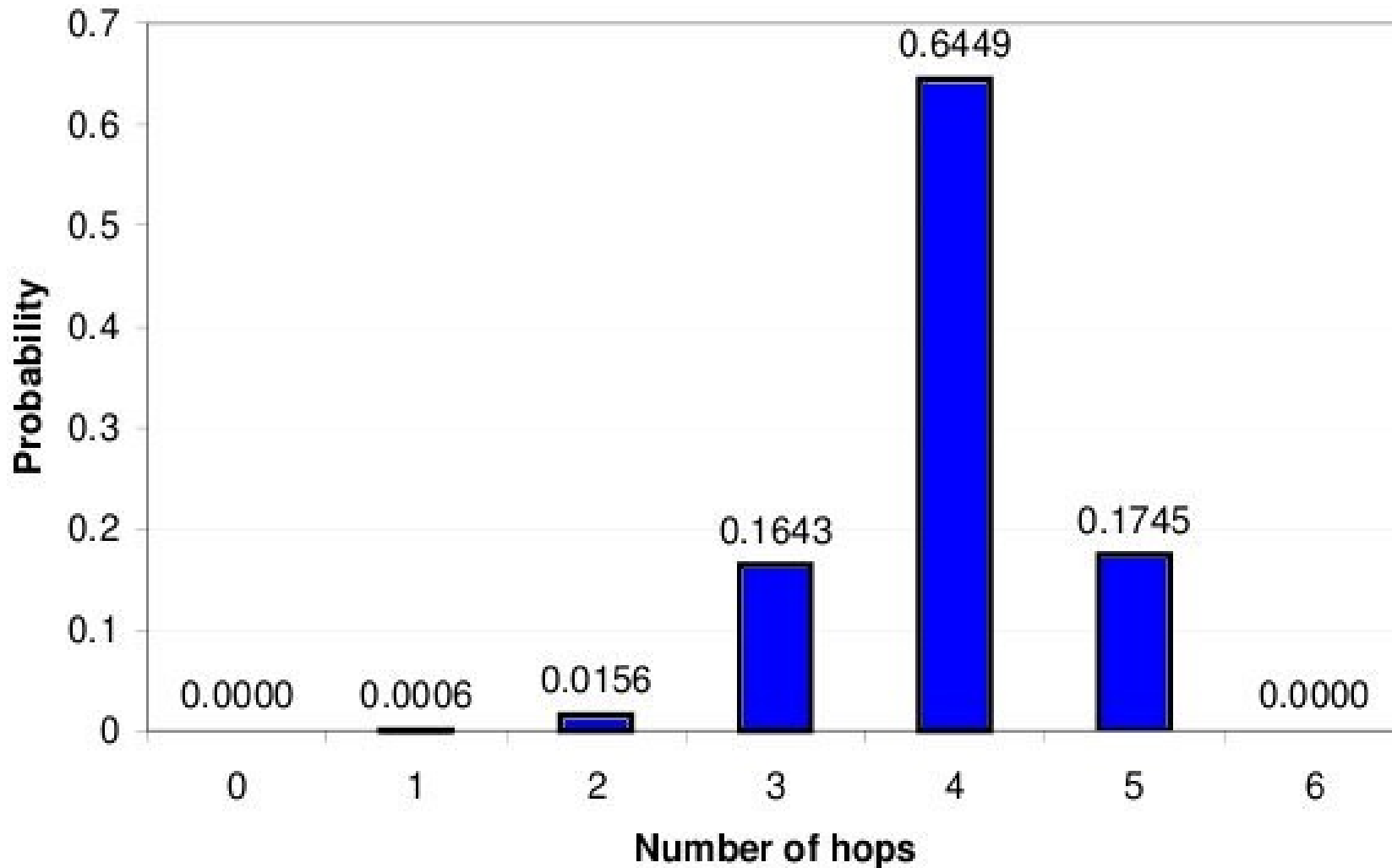
- Routing table for X is built using routing tables from nodes in route from A to Z .
 - i -th row of X 's routing table is i -th row of i -th node's routing table in the route from A to Z .
- Assumption: routing tables entries refer to near nodes.

Pastry vs Chord

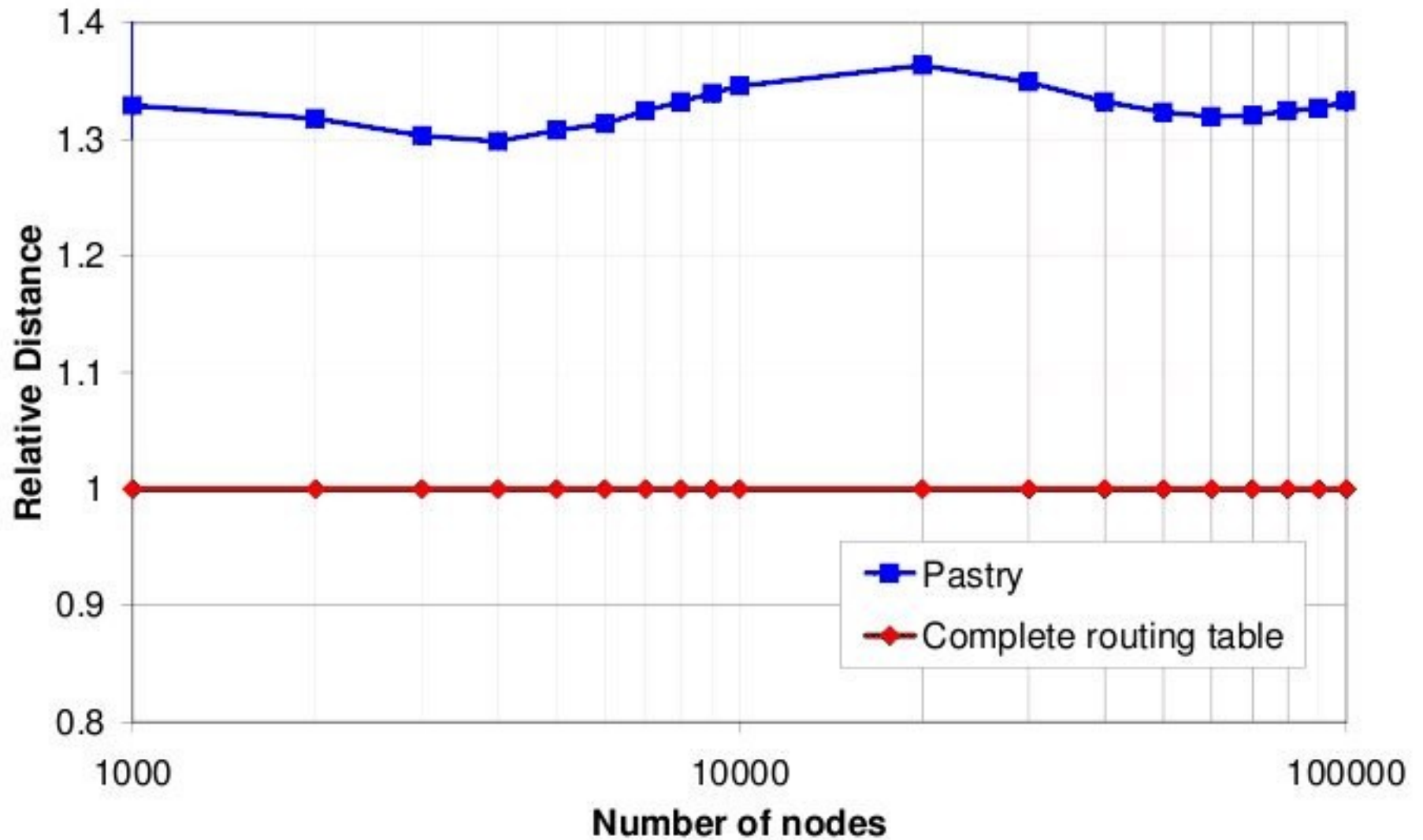
- Both are based on a virtual circular space.
- Both require $O(\log N)$ node space and lookup cost.
- But, Chord doesn't take locality into account.
- But, Pastry doesn't have any stabilization protocol (reactive).



Experiment (Hop Count)



Experiment (Distance)

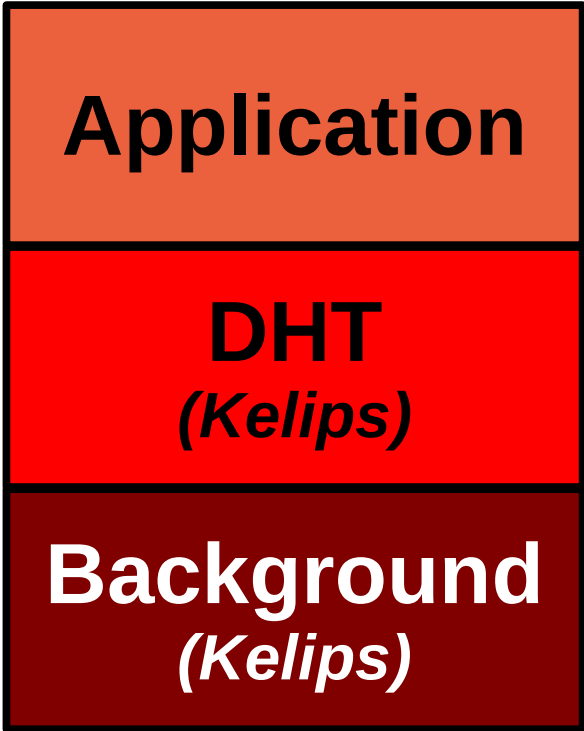
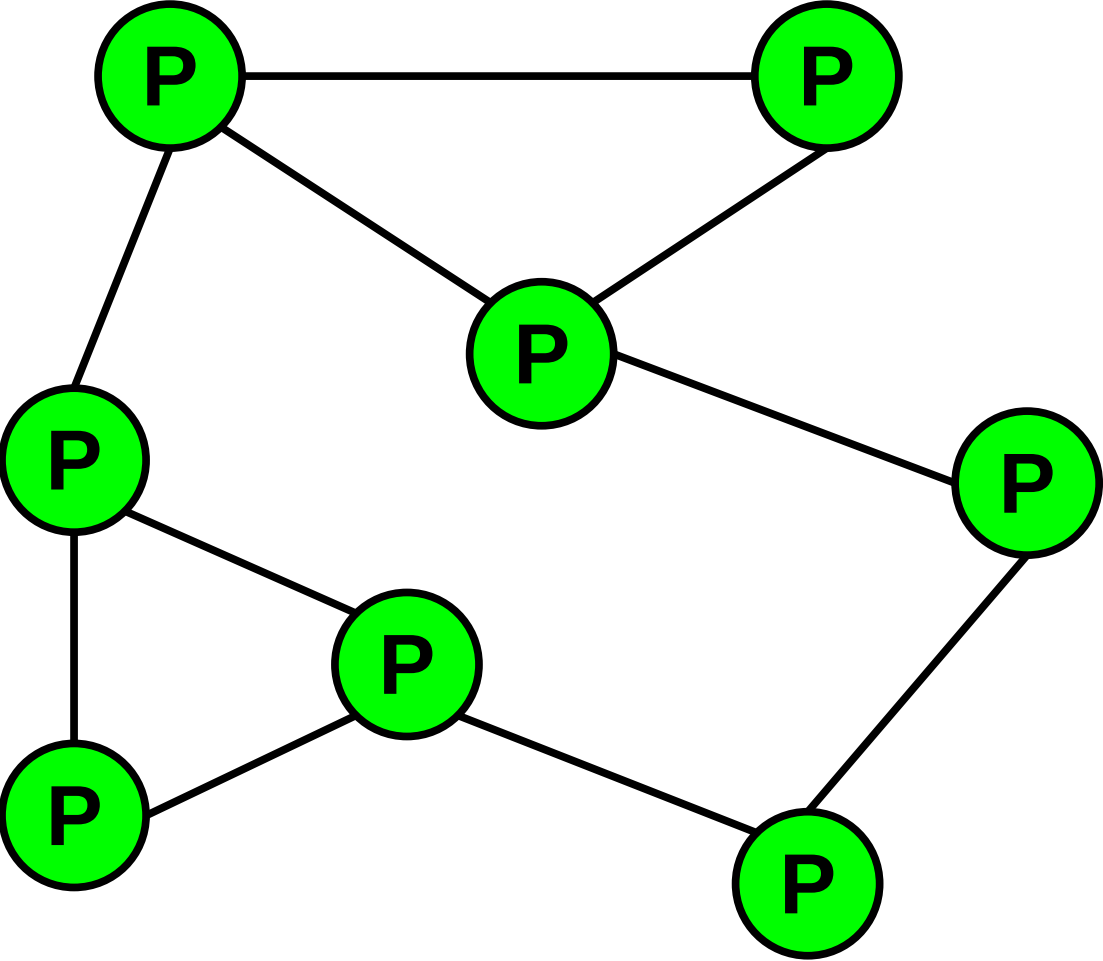


Kelips: Building an Efficient and Stable P2P DHT Through Increased Memory and Background Overhead

Indranil Gupta, Ken Birman, Prakash Linga, Al Demers, Robbert van Renesse (Cornell)

IPTPS 2003

Motivation



Can we have $O(1)$ lookup cost?

Increment memory usage

Design

- Each node is in one of k affinity groups.
- Affinity Group View: partial set of nodes in the same group (RTT, heartbeat count).
- Contacts: set of nodes for other affinity groups in the system.
- Filetuples: partial set of tuples for filenames and their host IP (*homenode*). Only holds tuples for files in the same affinity group.
- Soft state: all the queries will be satisfied, likely (One hop lookup, Beehive).

Soft State

- It uses structured replication to provide $O(1)$ lookup latencies on overlays.
- More than $O(\log N)$ memory space to keep track of membership changes.
- Less than 2MB memory usage in a 10^5 node system with 10^7 files.
- Invariants of Chord and Pastry: costly to maintain.

Memory Usage at Node

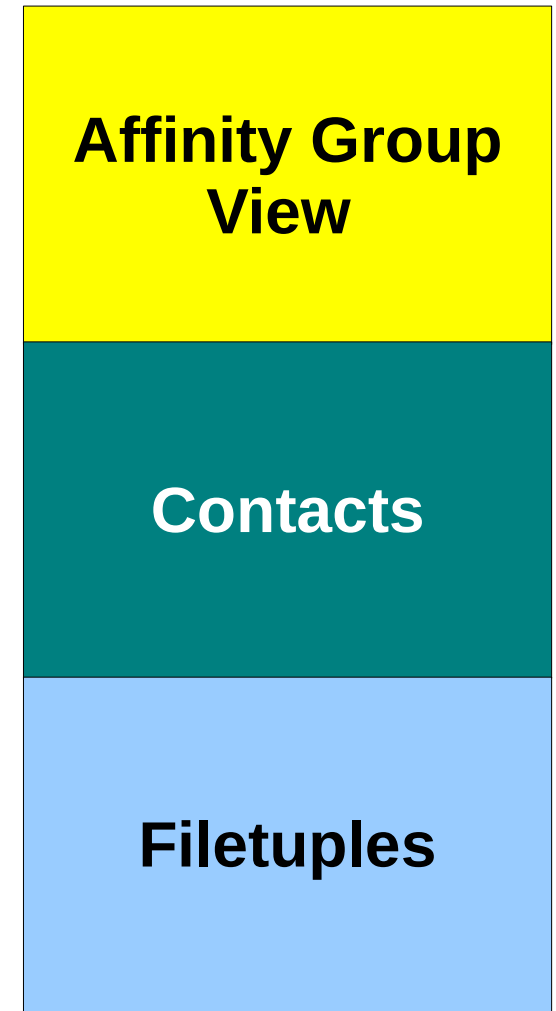
- Total storage requirements:

$$S(n, k) = \frac{n}{k} + c \times (k - 1) + \frac{F}{k}$$

- Minimized at:

$$k = \sqrt{\frac{n + F}{c}}$$

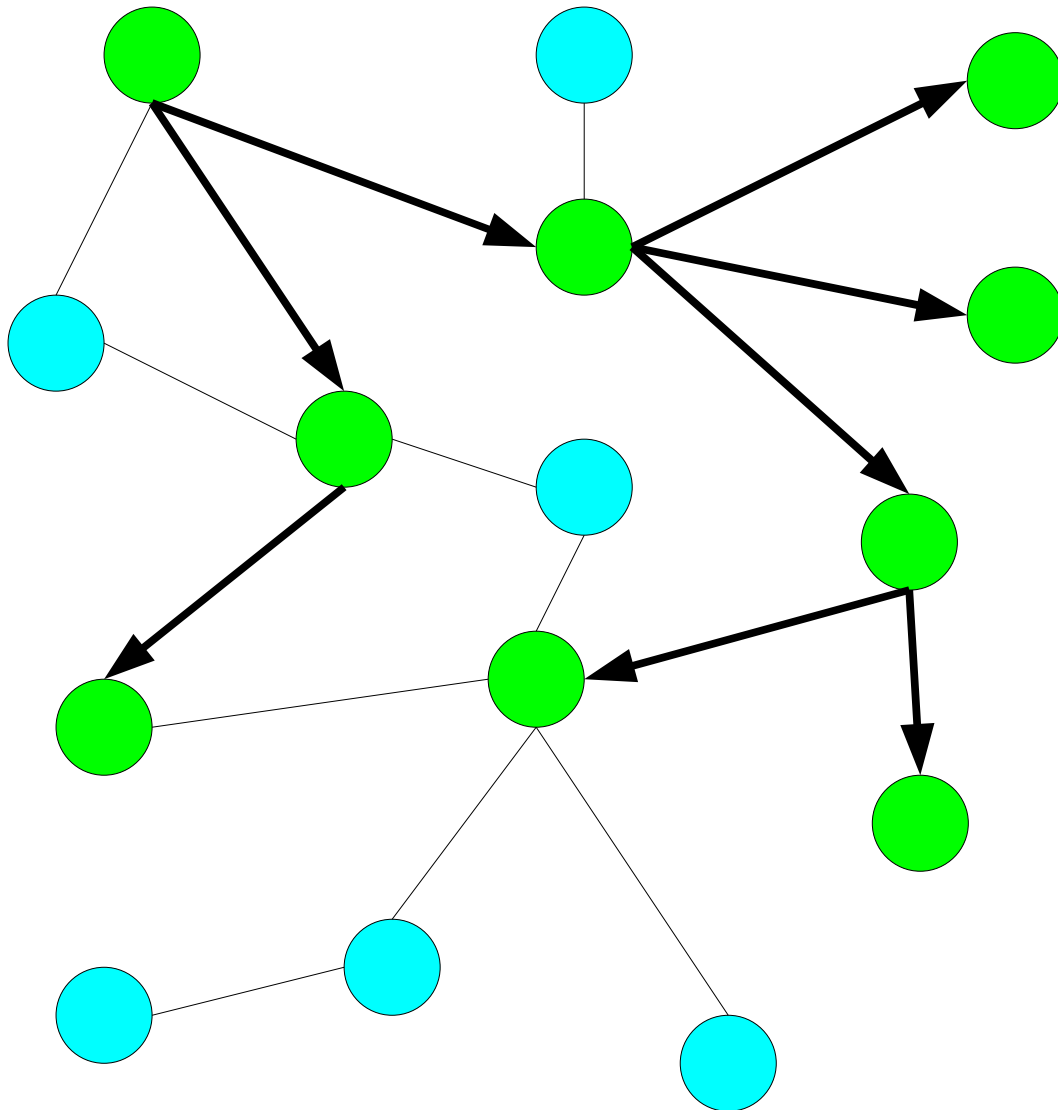
- n is the total number of nodes.
- c is the number of contacts.
- F is the total number of files.



Background Communication

- Group view, contacts and file tuples are constantly refreshed within and across groups.
- Heartbeat mechanism: every item is associated with a heartbeat count.
- Originated at the responsible node and disseminated by a gossip-style protocol.
- Each node periodically selects a few nodes as gossip targets (same group & contacts) to send them partial soft state information (constant gossip message size).
- Gossip target selection: topologically aware (RTT).

Gossiping Protocol



**Works in rounds
of fixed time**

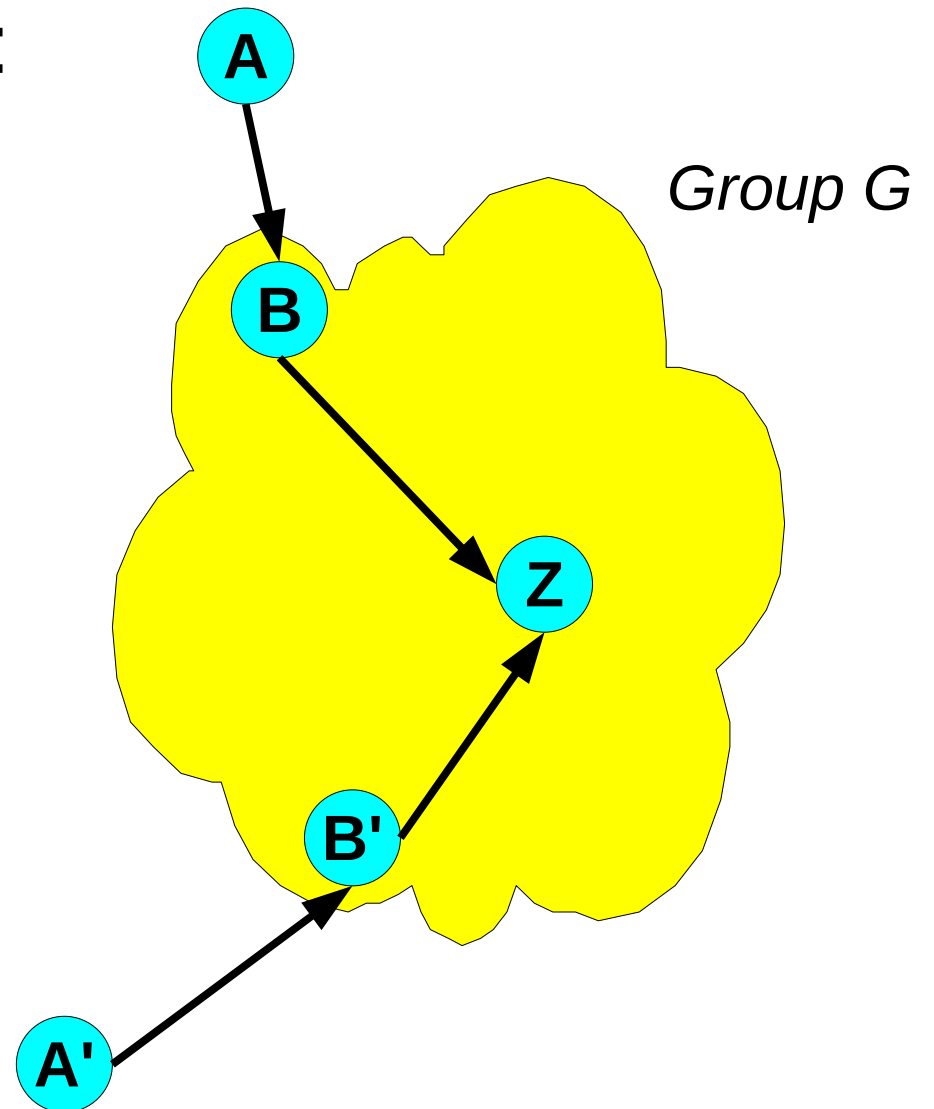
**Uses constant
bandwidth**

**Spatially weighted
gossip**

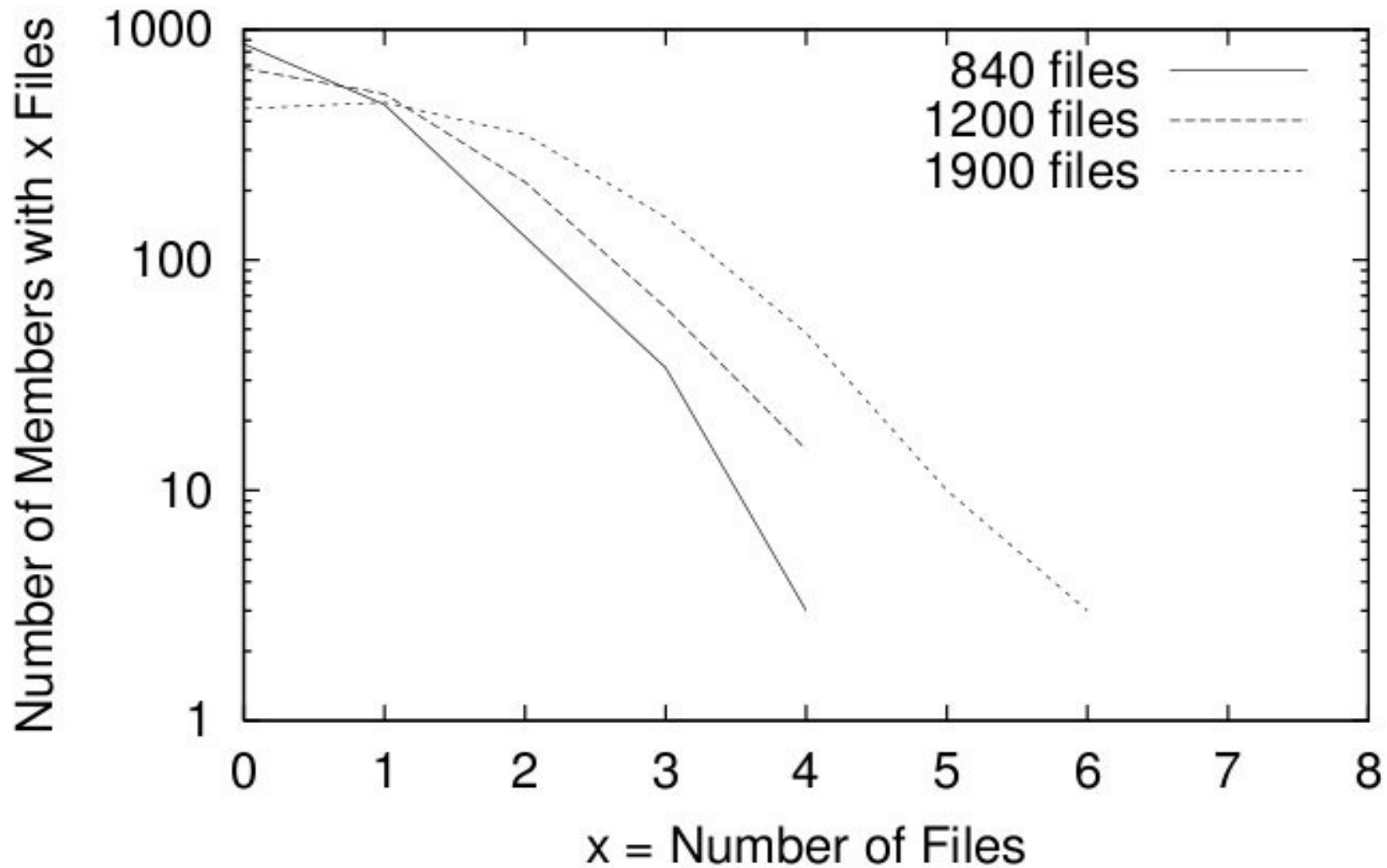
**Robust to network
packet loss and
node failures**

File Lookup and Insertion

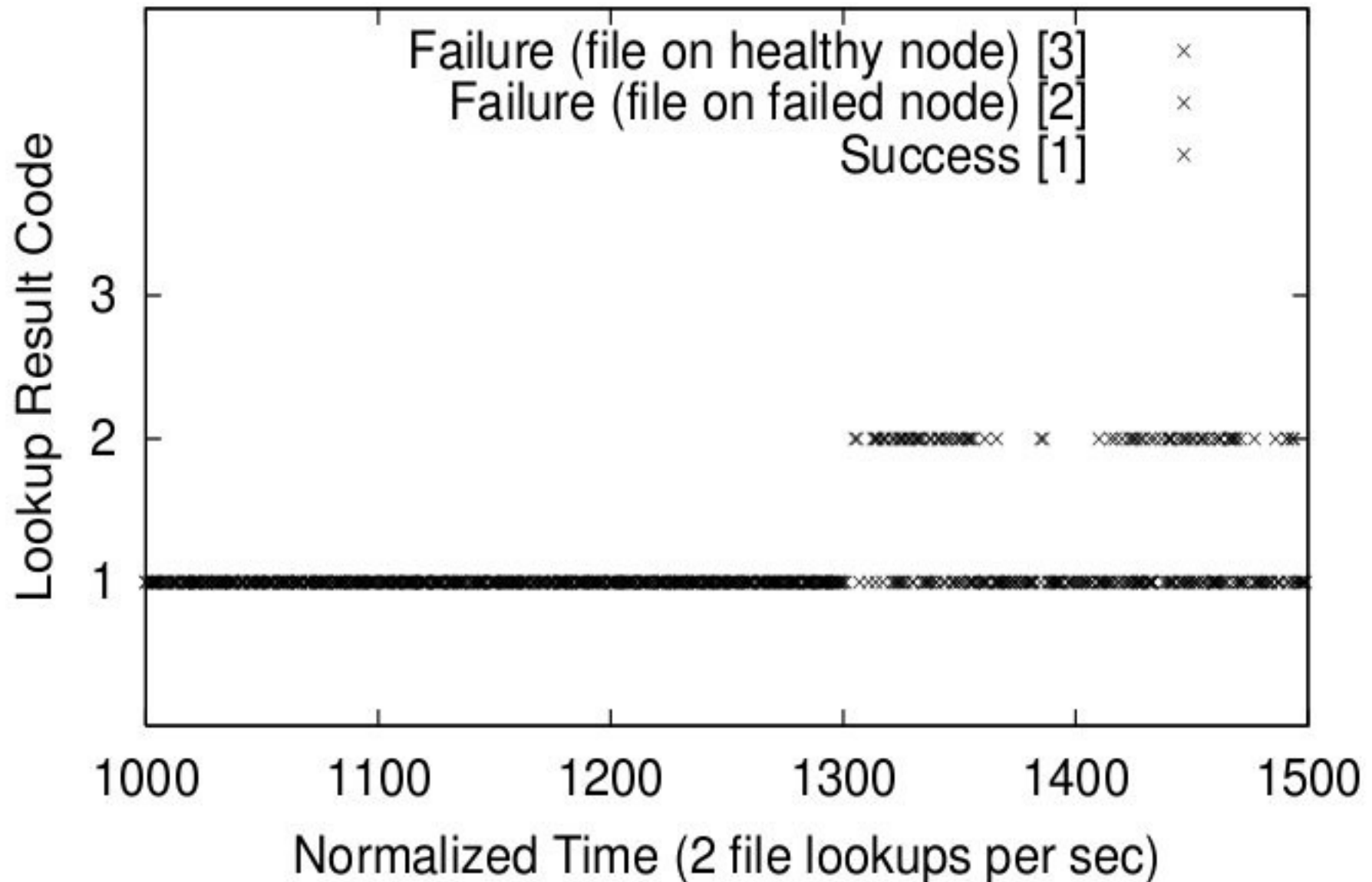
- Lookup (key D at node A):
 - $\text{hash}(D) \rightarrow G$
 - Sends a message to the topologically closest contact in G . Let's call it B.
 - B searches in its file tuples for D, whose *homenode* is Z.
- $O(1)$ complexity.
- Insertion: equivalent to lookup.



Experiment (Load Balancing)



Experiment (Fault Tolerance)



Discussion (III): Pastry vs Kelips

Criteria	Pastry	Kelips
<i>Memory</i>	$O(\log N)$	$O(\sqrt{N})$
<i>Lookup Latency</i>	$O(\log N)$	$O(1)$
<i># messages for lookup</i>	$O(\log N)$	$O(1)$
<i>Routing</i>	<i>Prefix-based</i>	<i>Affinity groups</i>

Discussion (IV)

- Which would you use, Chord or Pastry?
- Node partition (Kelips) and nodeIDs.
 - Two different mappings.
 - CAN and virtual spaces.
- Node failure detection: *proactive* (Kelips) vs *reactive* (Pastry).
- Pastry: resilient to high churn?
- Is locality so important?

Discussion (V)

- Kelips: multiple partitions for file replication?
- Kelips: designed for medium-sized p2p systems. The memory requirement is larger than Chord or Pastry.
- Kelips: churn resistant?
- Kelips: does it make any sense to form groups according locality/density?

Thank you!