As Scalable As Possible

Motivation

- Scale shift in distributed systems
- Key to scalability: Peer to peer communication paradigm
  - Ranging from unstructured to fully structured overlays.
  - Provide various functionalities (search)

- One physical peer may host several logical peers belonging to different overlays

Build one, get one free

- Leverage the existence of multiple overlays
- P2P structured overlay network
  - Constrained component (leafset)
  - Non-constrained component (Routing table)
- Gossip-based clustering protocol
  - Non-constrained component (Random sampling protocol)
  - Constrained component (Cluster sampling protocol)
- Build the constrained components and get for free the non-constrained ones
- Out of the scope of this talk
  - Network locality
  - Application switch between multiple overlays

Roadmap

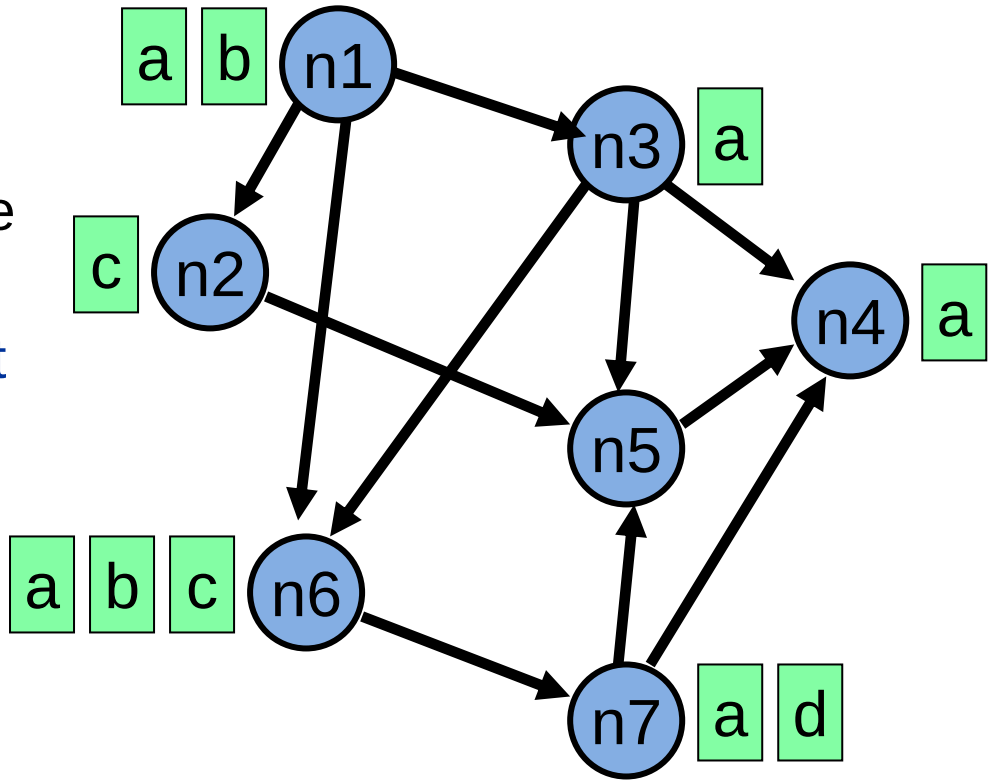
1. **Design rationale**
2. Pastry: a structured peer to peer overlay
3. Gossip-based clustering protocol
4. Construction of the joint overlay
5. Simulation results
6. Conclusion and discussion

Peer to peer overlay networks

- Logical network on top of a physical networking infrastructure
  - A peer may act both as a client and a server
  - Resource aggregation
  - Fully decentralized: Limited knowledge of the network
- Properties
  - Scalable
  - Robust
  - Self-organizing

Search in peer to peer overlays

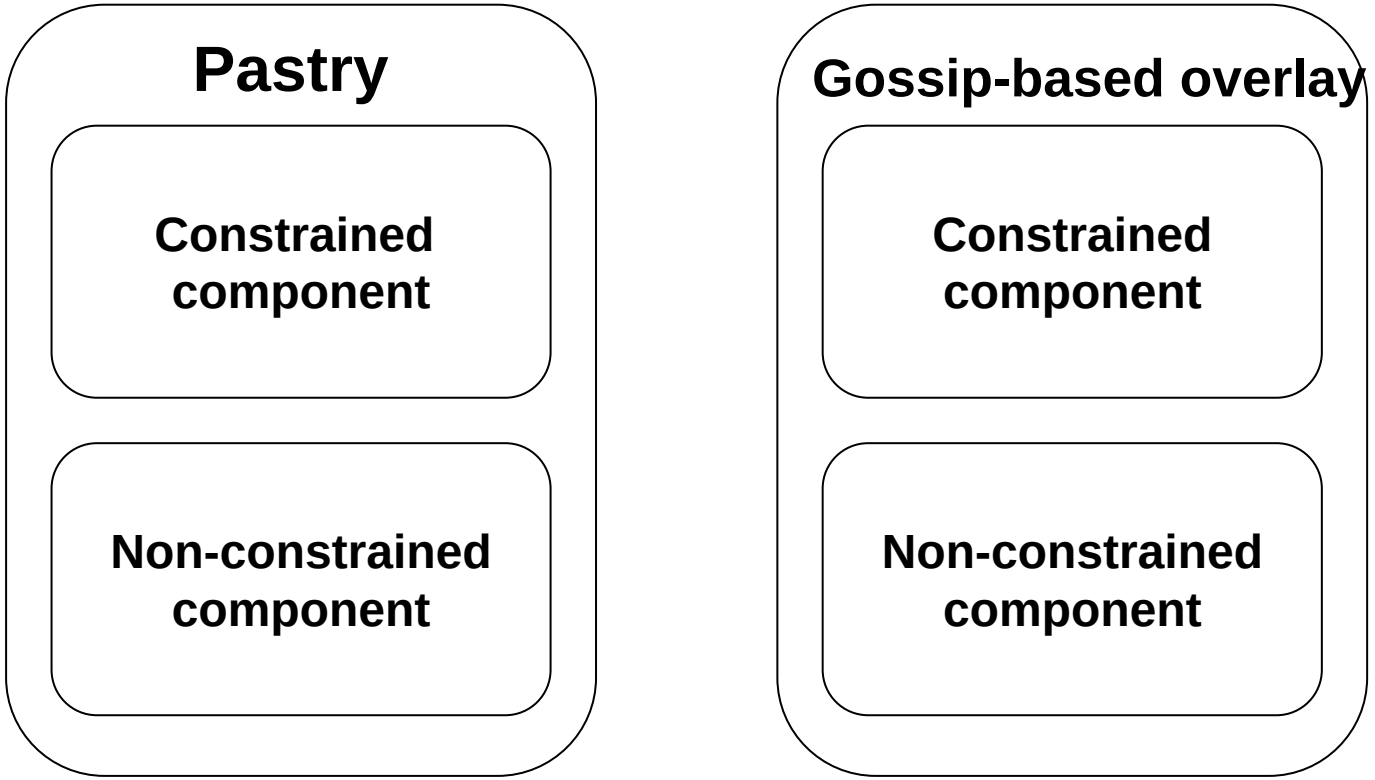
- Data distributed (and potentially replicated) between nodes
- Each node knows only the IP @ of its neighbours
- How to find a data without a central index?



Structure of peer to peer overlays

- Several ways of organizing a P2P overlay network
  - Search techniques
  - Expressiveness
    - A file uniquely identified as #4a56b23
    - All Britney Spears mp3 files
- Structured P2P overlay: DHT functionality
  - Support for exact search
- Unstructured gossip-based P2P overlays
  - Cope well with dynamics
  - Weakly structured overlay networks
  - Support for keyword-based search or range queries

Design rationale



Leveraging the presence of multiple overlays

- What metrics matter?
  - State to maintain
  - Routing performance

More maintenance for a better performance

Less maintenance for a similar performance

Roadmap

1. Design rationale
2. **Pastry: a structured peer to peer overlay**
3. Gossip-based clustering protocol
4. Construction of the joint overlay
5. Simulation results
6. Conclusion and discussion

Structured P2P overlays

- Rely on a predefined data structure: tree, ring, linked lists, skip lists etc…
- Peers are assigned a unique Id
- Data are identified by a key
- Map key to peers: Provide a support for a DHT functionality
- Existing P2P overlays: Pastry, Chord, CAN, Tapestry, etc.

Pastry (MSR/Rice)

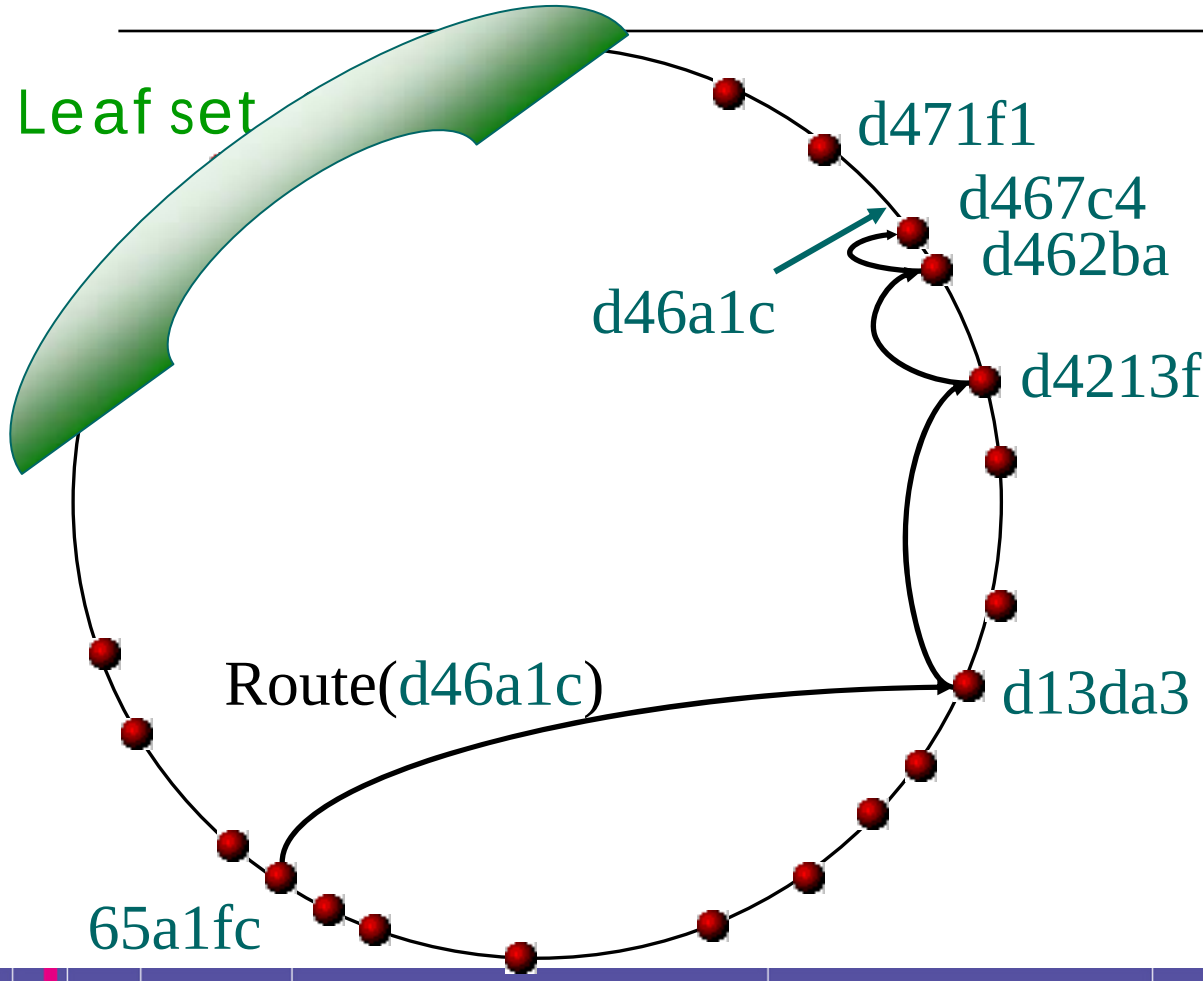
- Naming space :
  - Ring of 128 bit integers
  - *nodeIds* chosen at random
- Key/node mapping
  - key associated to the node with the numerically closest node id
- Data structures
  - Routing table
    - Identifiers are a set of digits in base 16
    - Matrix of 128/4 lines et 16 columns
    - routeTable(i,j): *nodeId* matching the current node identifier up to level I with the next digit is j
  - Leaf set*:* 8 or 16 closest numerical neighbours in the naming space

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

$\log_{16} N$ lines

Pastry: Routing



Properties

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

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

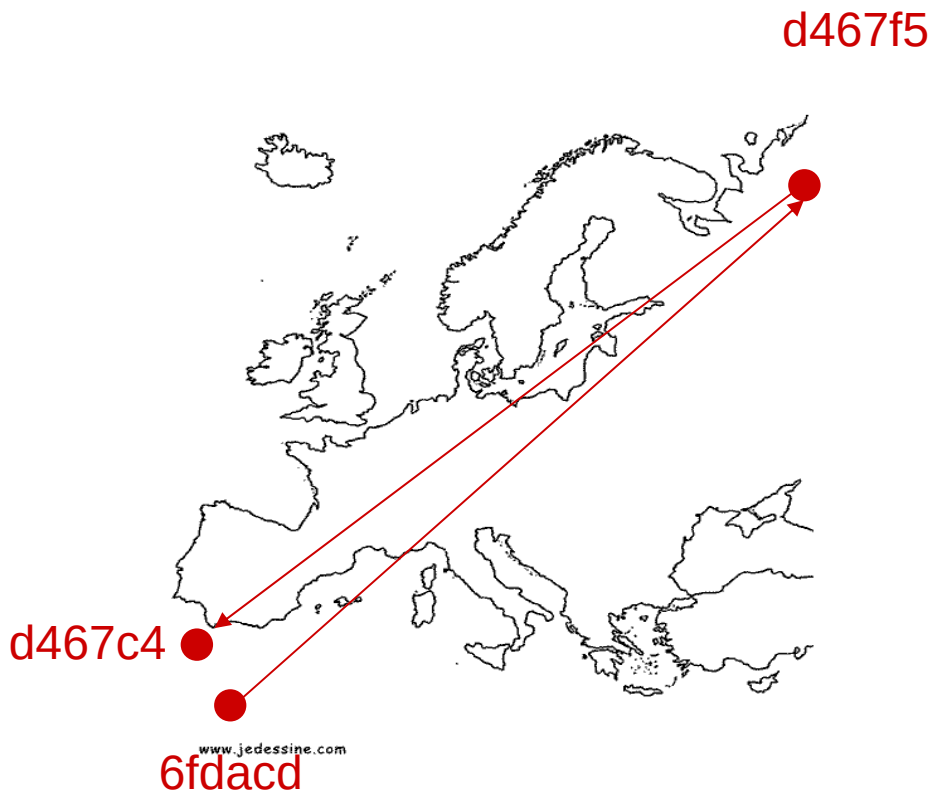
Failure detection

- 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
- ***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

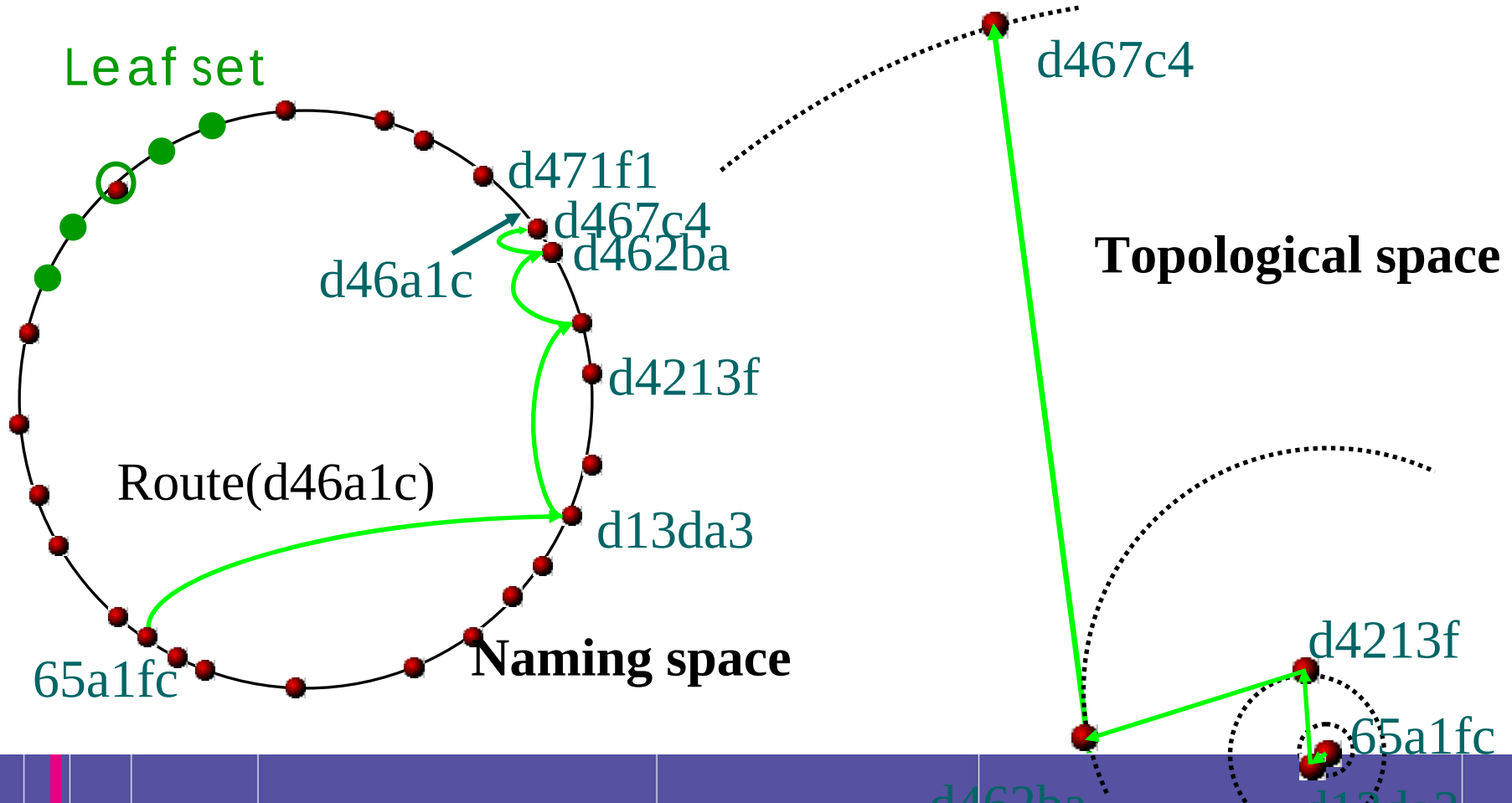
- **Random assignment of nodeId**: Nodes numerically close are geographically (topologically) distant
- **Objective**: fill the routing table with nodes so that routing hops are as short (latency wise) as possible
- **Topological Metric**: latency



Exploiting locality in Pastry

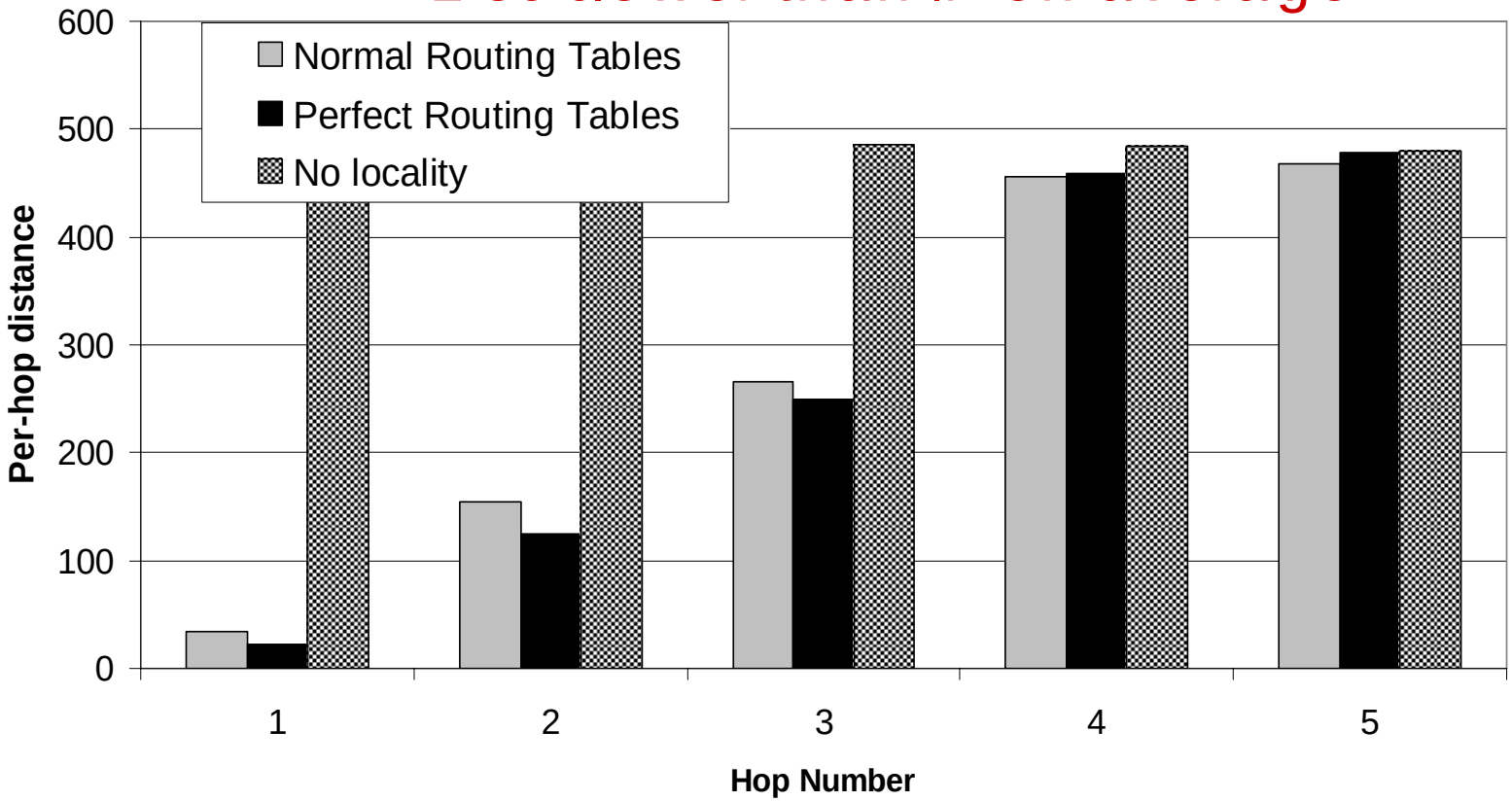
- Neighbour selected based of a network proximity metric:
  - Closest topological node
  - Satisfying the constraints of the routing table routeTable(i,j):
    - *nodeId* corresponding to the current *nodeId* courant up to level i
    - next digit = j
  - nodes are close at the top level of the routing table
  - random nodes at the bottom levels of the routing tables

Proximity routing in Pastry

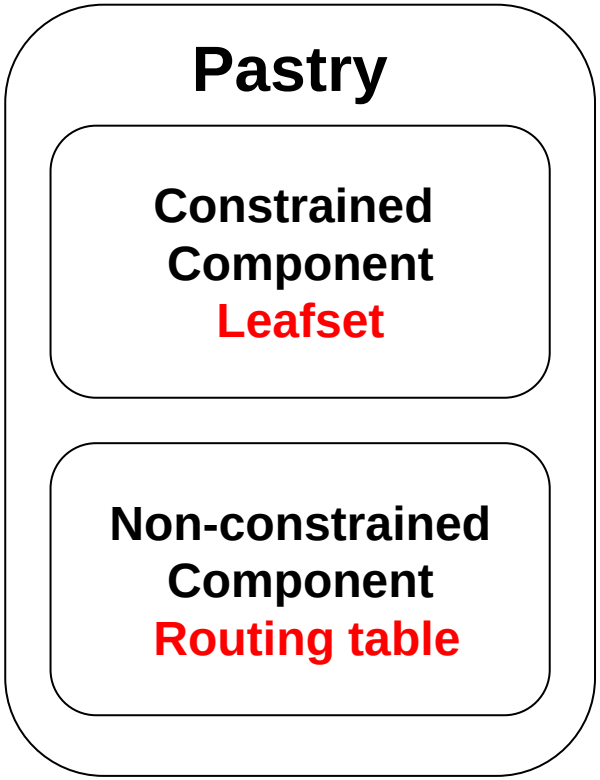


Performance

1.59 slower than IP on average



Summary


Pastry
Constrained
Component
Leafset
Non-constrained
Component
Routing table

Roadmap

1. Design rationale
2. Pastry: a structured peer to peer overlay
3. Gossip-based clustering protocol
4. Construction of the joint overlay
5. Simulation results
6. Conclusion and discussion

Unstructured P2P networks

- No (or few) constraints of the choice of neighbours
- Data are stored on any node (no index)
- Search by controlled flooding
- Gossip-based protocols
  - Extremely robusts
  - Properties close to those of random graphs

Gossip-based protocols

- Unstructured peer to peer networks
  - Highly resilient to failure and dynamics
- Gossip-based membership protocols
  - Periodic exchange of information between nodes
- Basic functionality: Peer sampling
  - Provide a sample of peers given a metric
  - Random sampling
- Applications
  - Event dissemination
  - Recovery protocols
  - Aggregation

Generic substrate

Dissemination

State = msg to broadcast

Topology management

State = membership information

Aggregation

State = data to aggregate

Active thread
Do once every T time units
P=selectPeer()
Send state to P
Receive state[P]
State= update(state[P])

Passive thread
Wait message(P)
Send state to P
State= update(state[P])

Design space

- Periodic peerwise communication
- **Peer selection**
- **View propagation**
  - How peers exchange their membership information?
  - What do they exchange?
- **View selection**: Select (c, buffer)
    - c: size of the resulting view
    - Buffer: information exchanged

Design space: view propagation

- Buffer (h)
  - initialized with the descriptor of the gossiper
  - contains c/2 elements
  - ignore h “oldest”
- Communication model
  - Push:buffer sent
  - Push/Pull: buffer sent both ways
  - (Pull: left out, the gossiper cannot inject information about itself, harms connectivity)

Design space: peer selection

- Selection
  - Rand: pick a peer uniformly at random
  - Head: pick the “youngest” peer
  - Tail: pick the “oldest” peer

Note that head leads to correlated views.

Design space: view selection

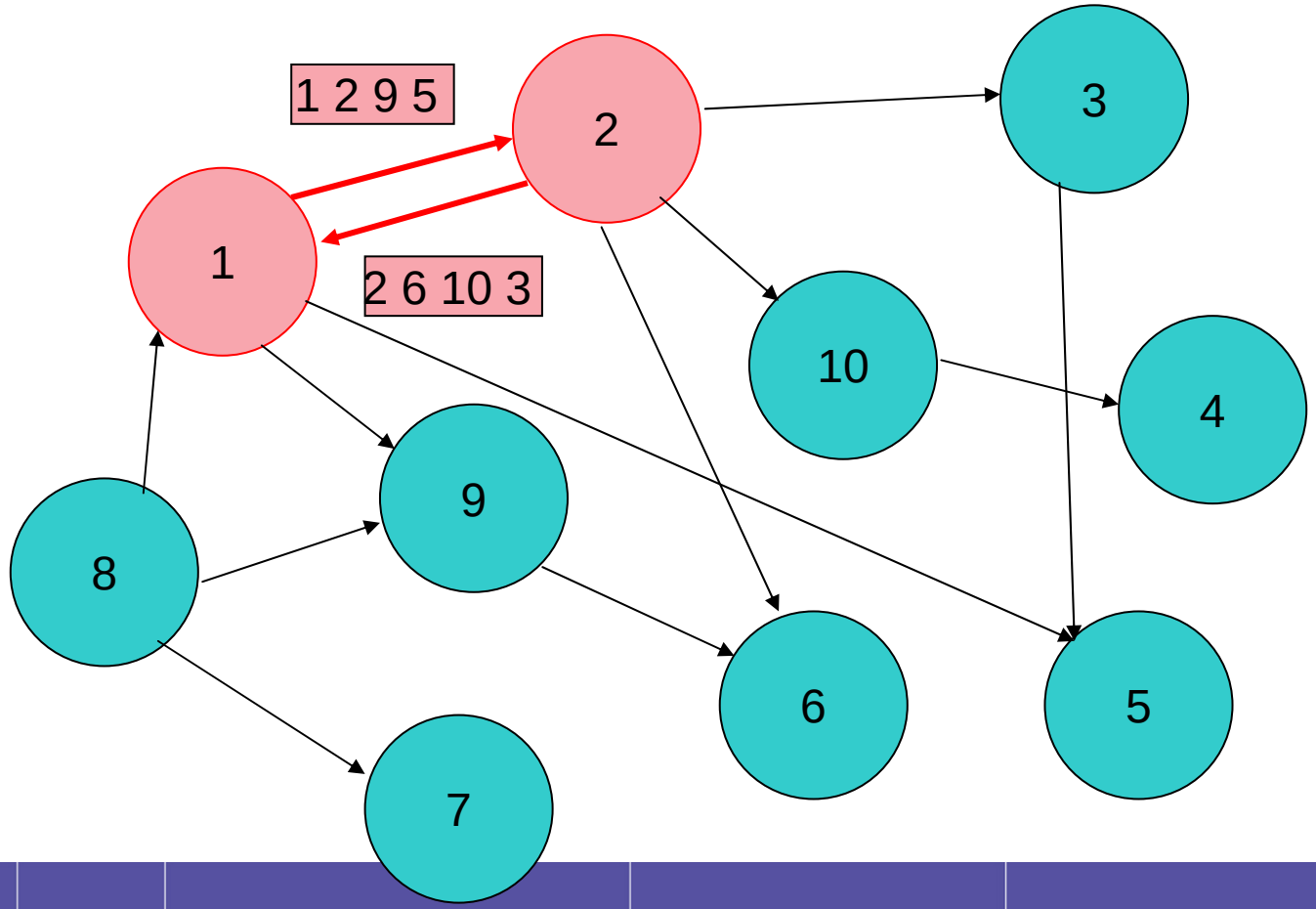
c: size of the resulting
H: self-healing parame
S: shuffle
Buffer: information exc

- Select(c,h,s,buffer)

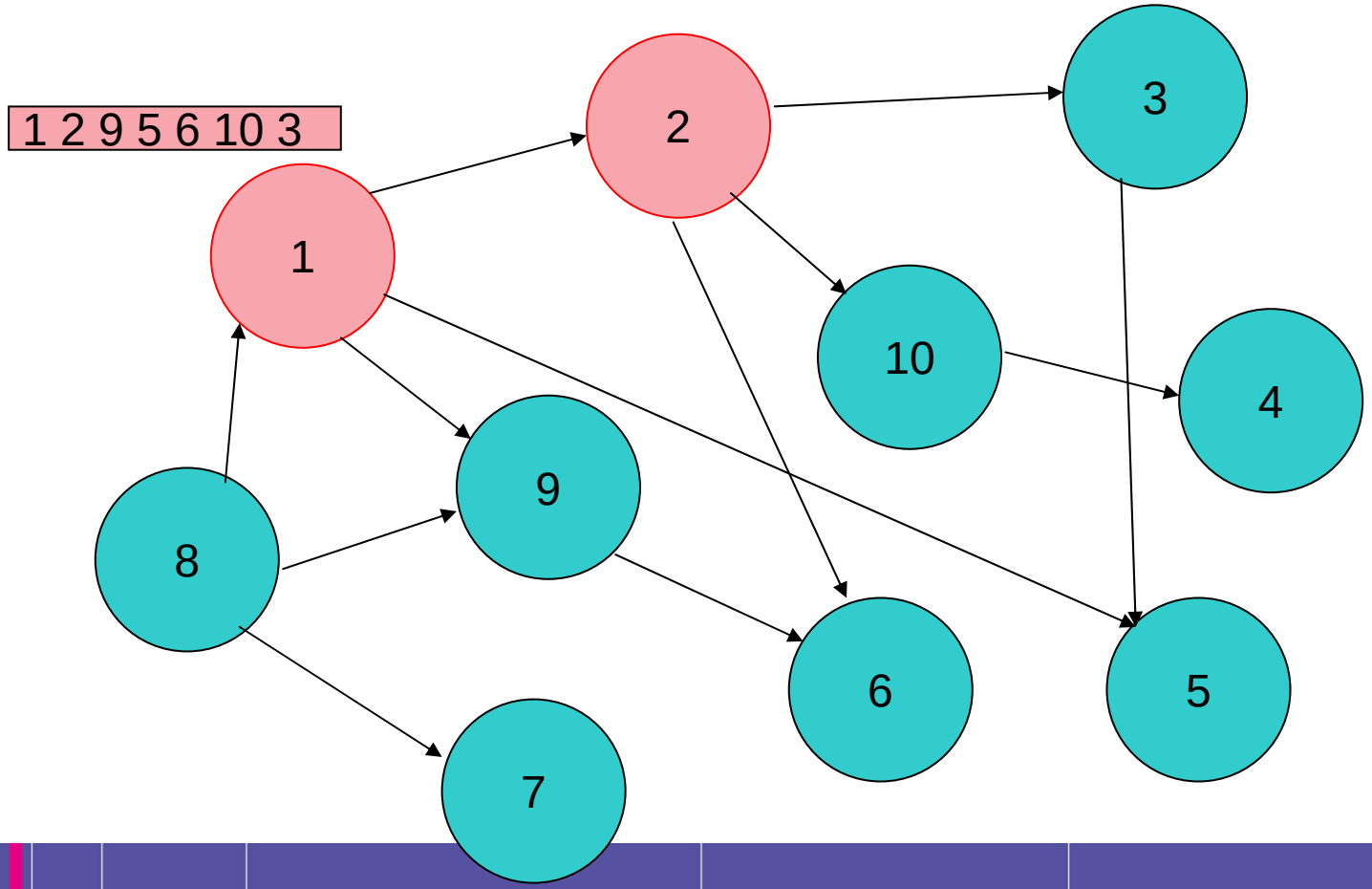
1. Buffer appended to view
2. Keep the freshest entry for each node
3. h oldest items removed
4. s first items removed (the one sent over)
5. Random nodes removed

- Merge strategies
  - Blind (h=0,s=0): select a random subset
  - Healer (h=c/2): select the “freshest” entries
  - Shuffler (h=0, s=c/2): minimize loss

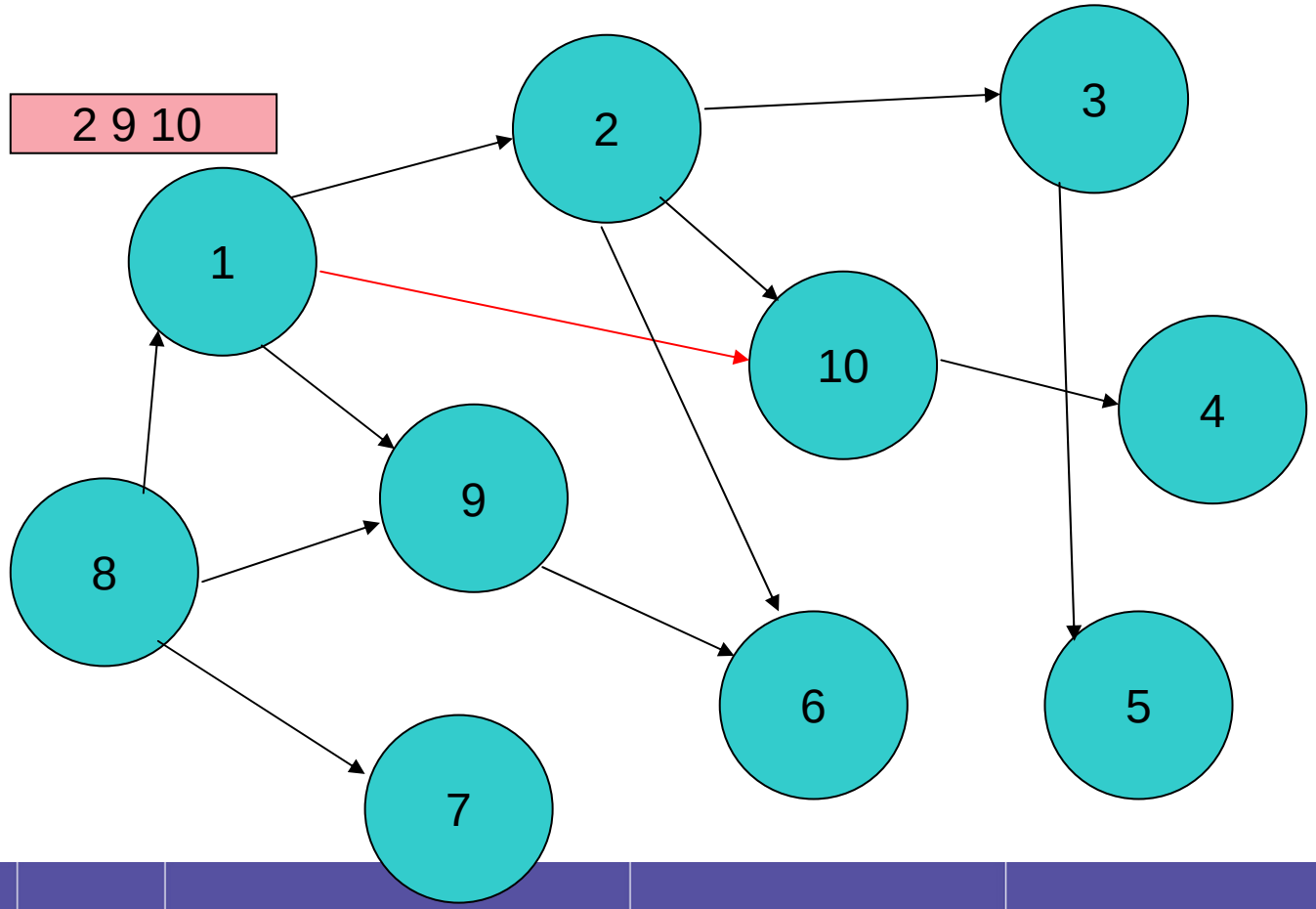
Gossip-based generic protocol



Gossip-based generic protocol



Gossip-based generic protocol



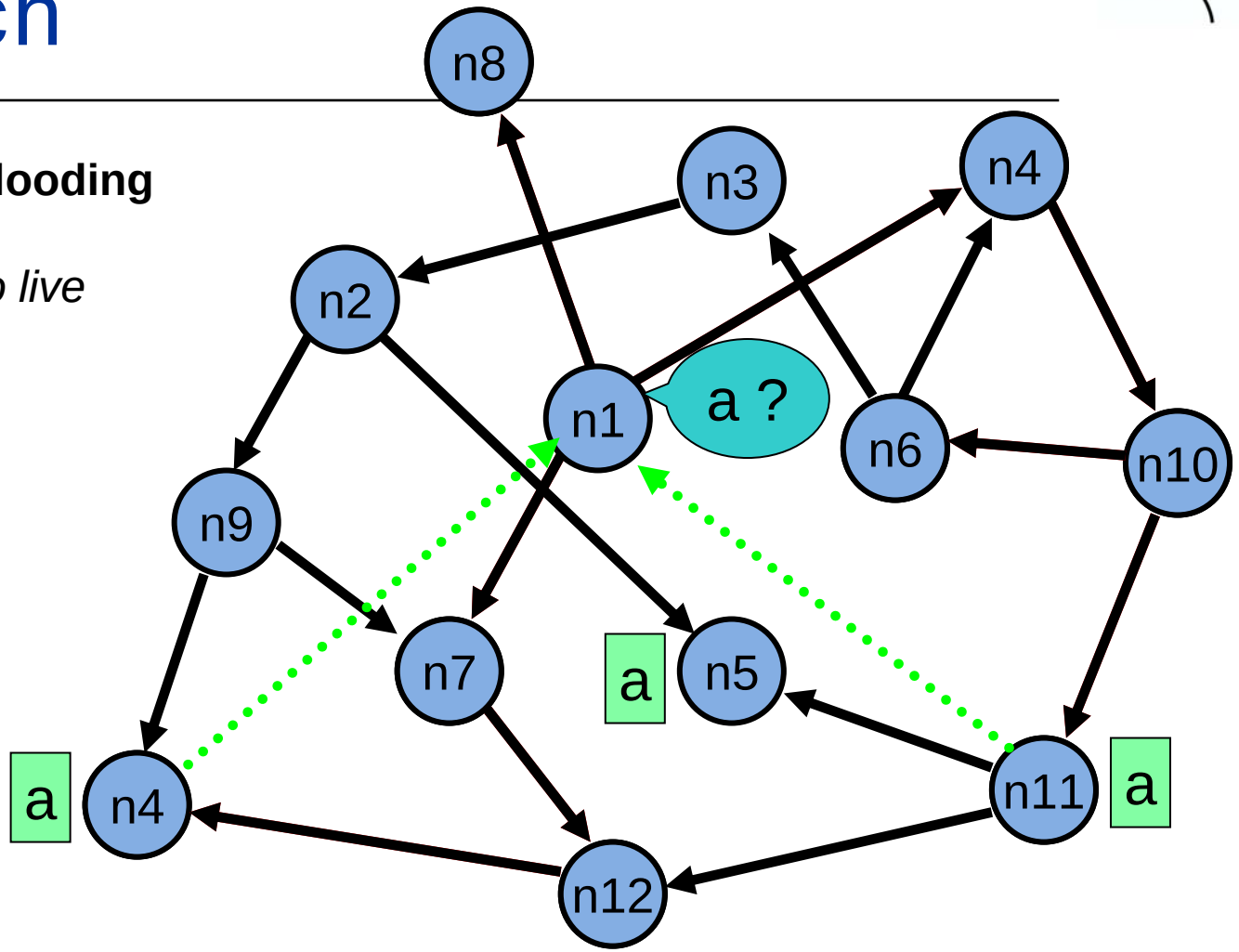
Peer sampling algorithm

- Common framework for existing gossip-based protocols
  - Lpbcast [Eugster & al, DSN 2001, ACM TOCS 2003]
  - Cyclon [Voulgaris & al JNSM 2005]
  - Newscast [Jelasity & van Steen, 2002]
- Provide random-graphs like properties
  - Average path length
  - Degree distribution
  - Clustering coefficient

Search

Controlled flooding

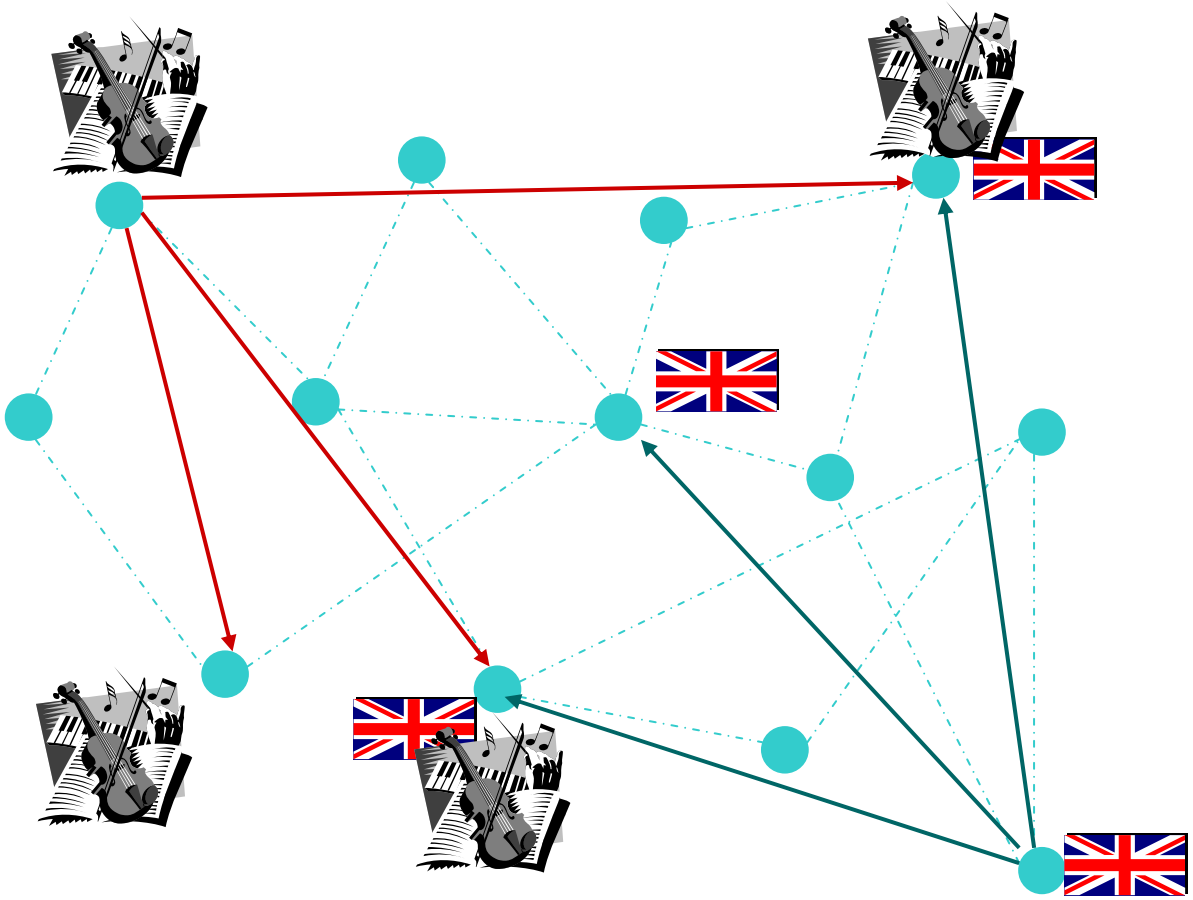
TTL = *time to live*
Ex TTL = 3



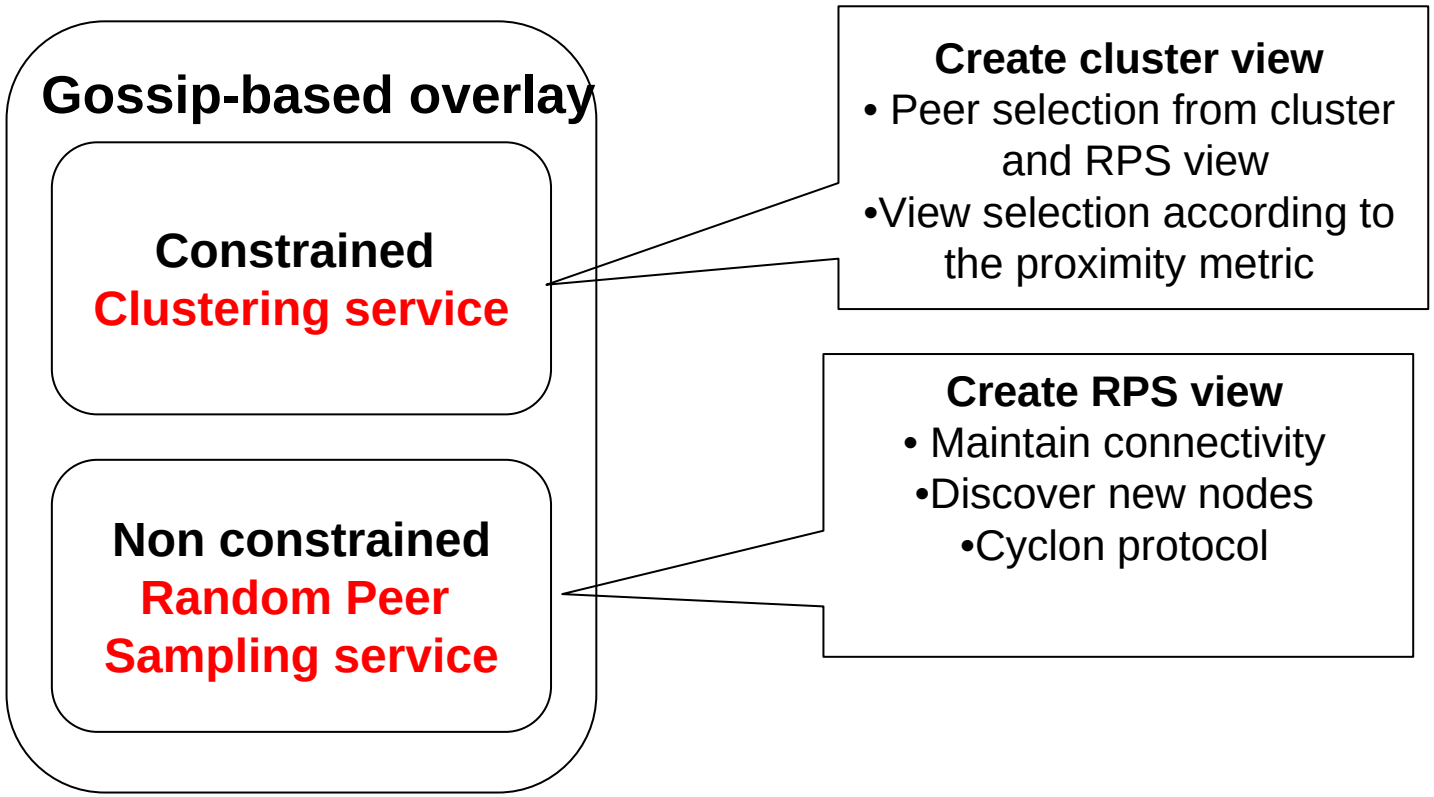
Clustering similar peers

- *Peers are not equal*
  - Geographical proximity
  - Social proximity
  - Interet-based proximity
- Leverage peers proximity to improve upon search performance
  - Application-dependent proximity metric
  - Use of gossip to discover « close » peers and let them form clusters
    - Peer selection and view selection: based on proximity metric

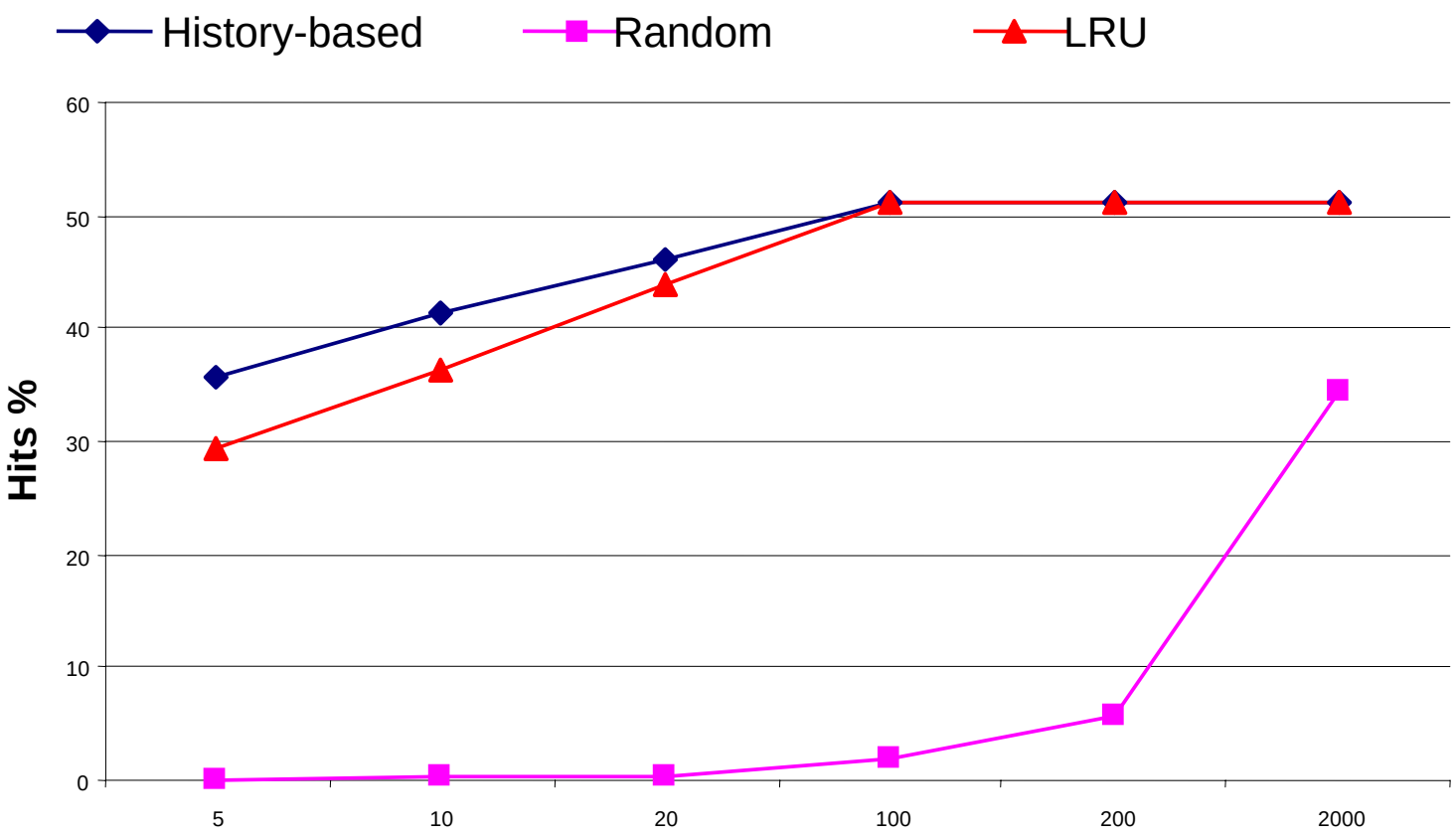
Creating proximity links



Gossip-based clustering protocol



Impact on hit rate



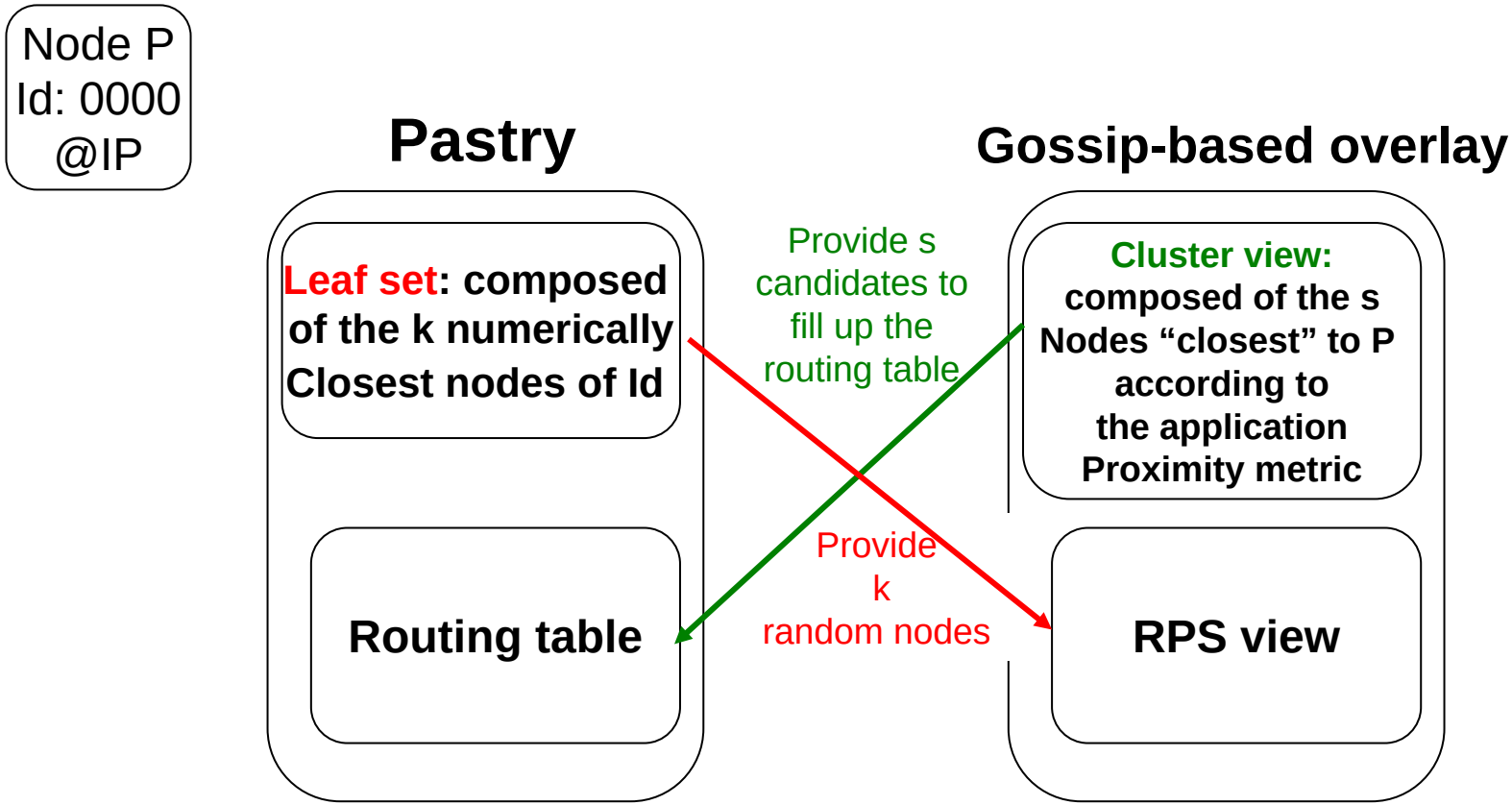
Roadmap

1. Design rationale
2. Pastry: a structured peer to peer overlay
3. Gossip-based clustering protocol
4. Construction of the joint overlay
5. Simulation results
6. Conclusion and discussion

Towards multiple overlays

- Easily deployable
- Relevant to have complementary overlays
  - DHT for exact search
  - Cluster-based (weakly structured) for keyword-based search
- Leverage the co-existence
- **Our contribution**: how to build ½ of Pastry and ½ gossip-based cluster-based overlay and get for free the other ½s.

Maintenance of a joint overlay



Routing structure

Pastry

Leafset

- Critical component
- Extremely constrained

Routing table

- Improve performance
- Lazily maintained
- Less constrained

Gossip-based

Cluster

- Extremely constrained (application-dependent)

Random peer sampling

- Random choice
- Ensure connectivity

Roadmap

1. Design rationale
2. Pastry: a structured peer to peer overlay
3. Gossip-based clustering protocol
4. Construction of the joint overlay
5. Simulation results
6. Conclusion and discussion

Simulation setup

- Peersim simulator
  - Pastry
  - C-Gossip: file sharing application
    - Interest-based proximity metric
  - 50,000 nodes
  - Growing network
  - Configurations
    - Static: nodes join, never leave
    - Failure scenario: 20% of the nodes fail

Simulation metrics

Compare the performance of the resulting overlay against the original ones and ideal cases

- Evaluation metrics
  - Pastry: number of empty cells in the routing tables
  - Gossip-based approach: size of the overlap between caches in a file sharing application

Interest score

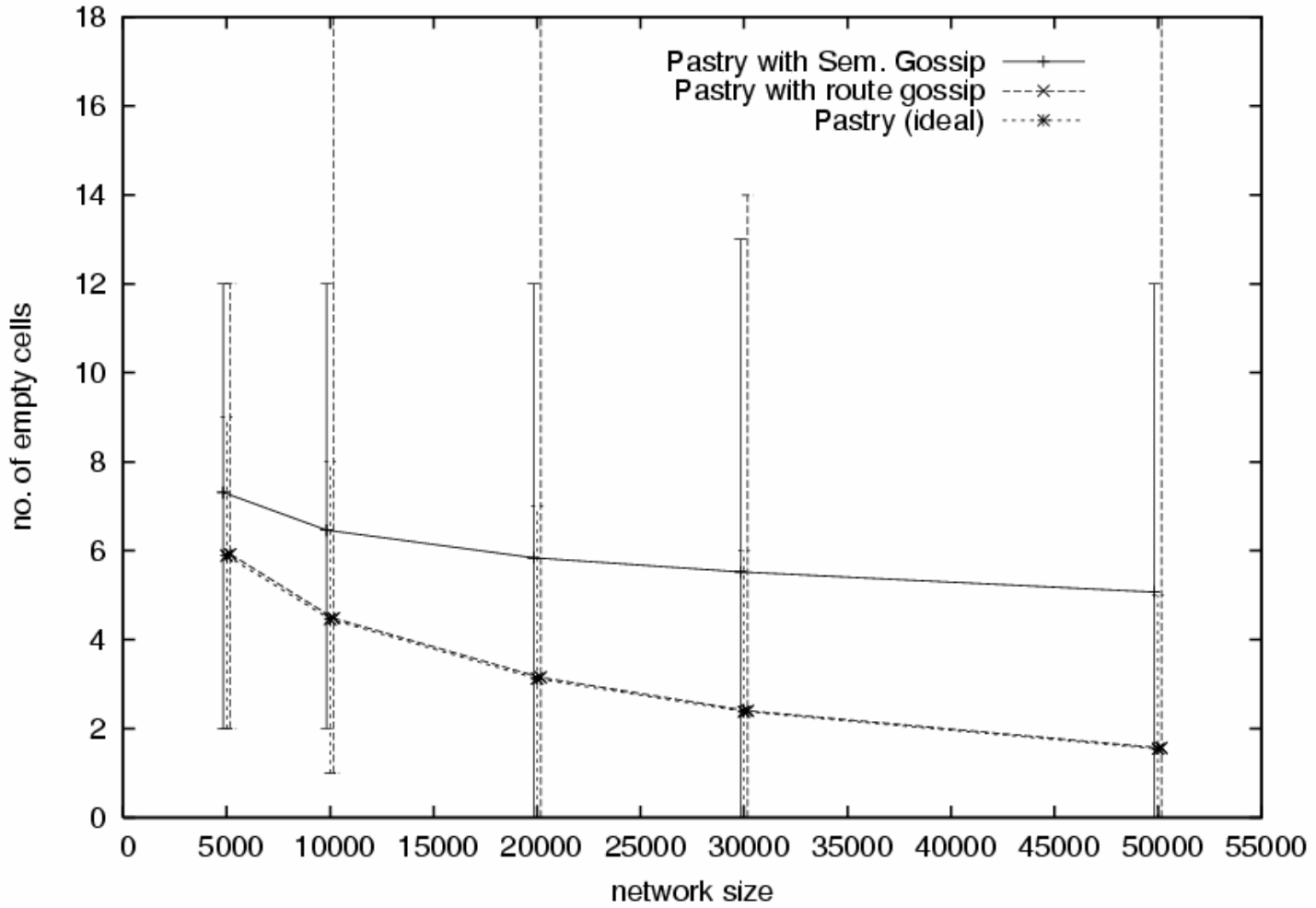
Node i, F_i being the content of node i

$$s_{i,j} = \frac{\left|F_i \cup F_j\right|}{\left|F_i\right|}$$

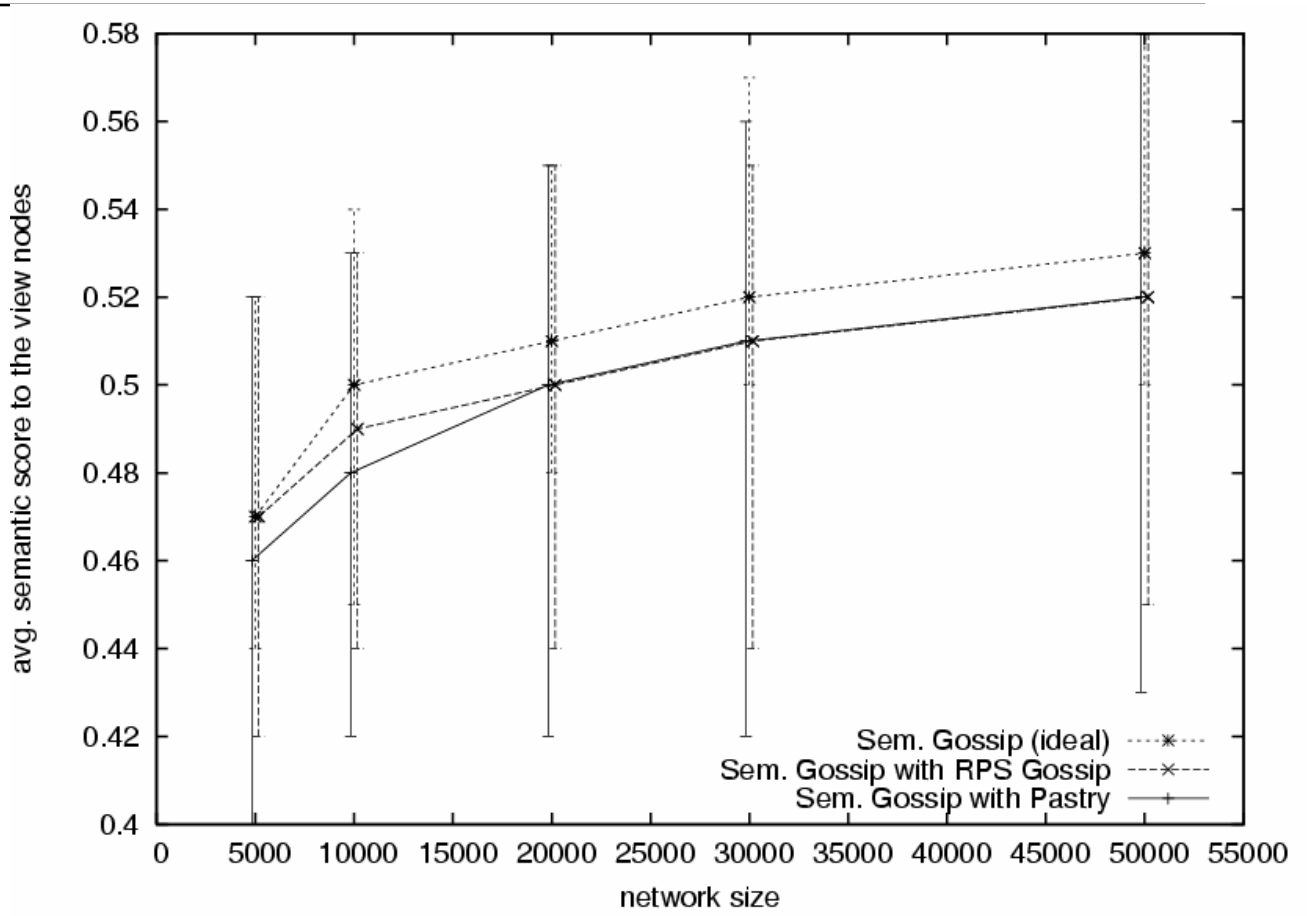
V_i is the view of node i

$$S_i = \frac{\sum_{j \in V_i} s_{i,j}}{\left|V_i\right|}$$

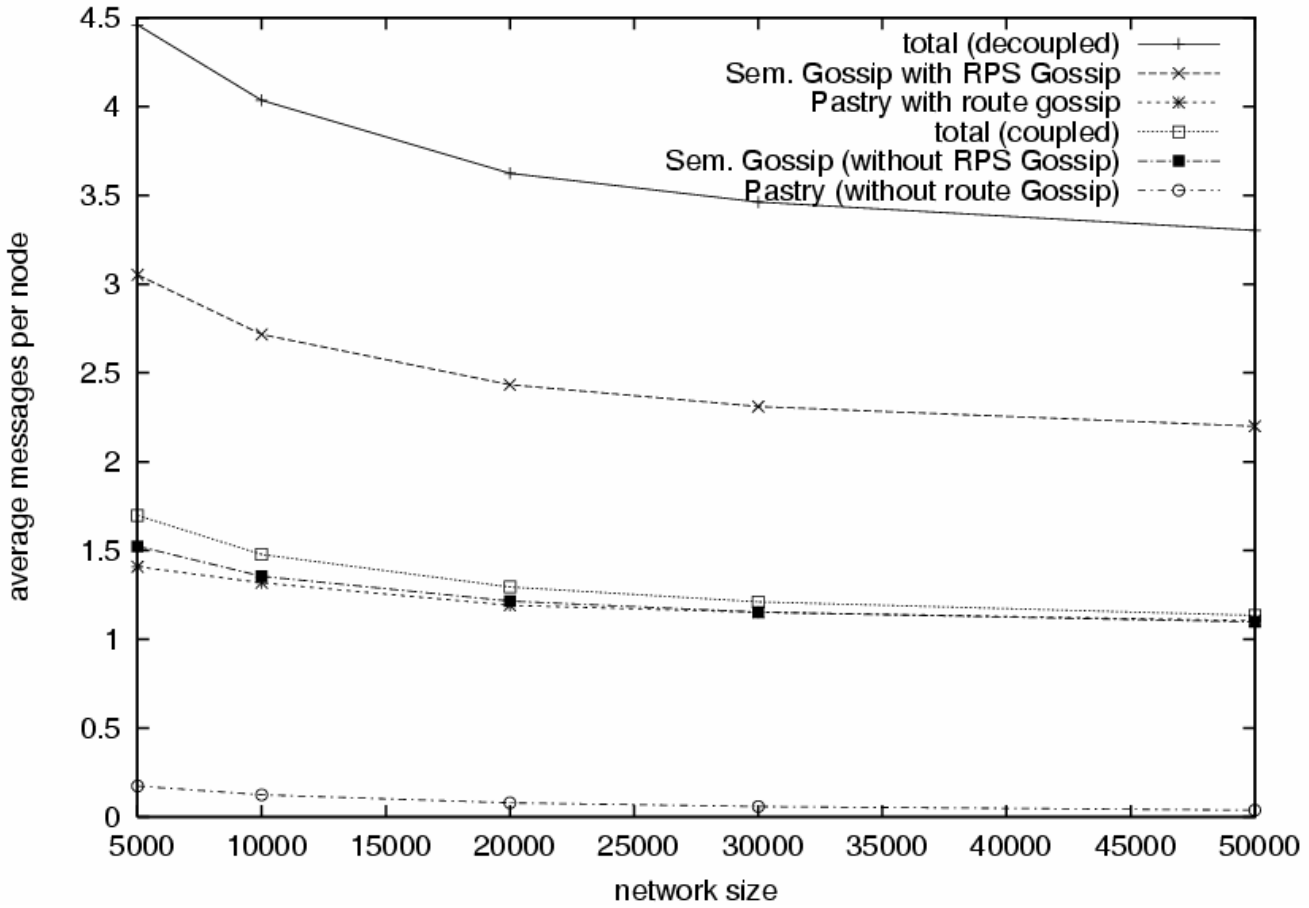
Static network



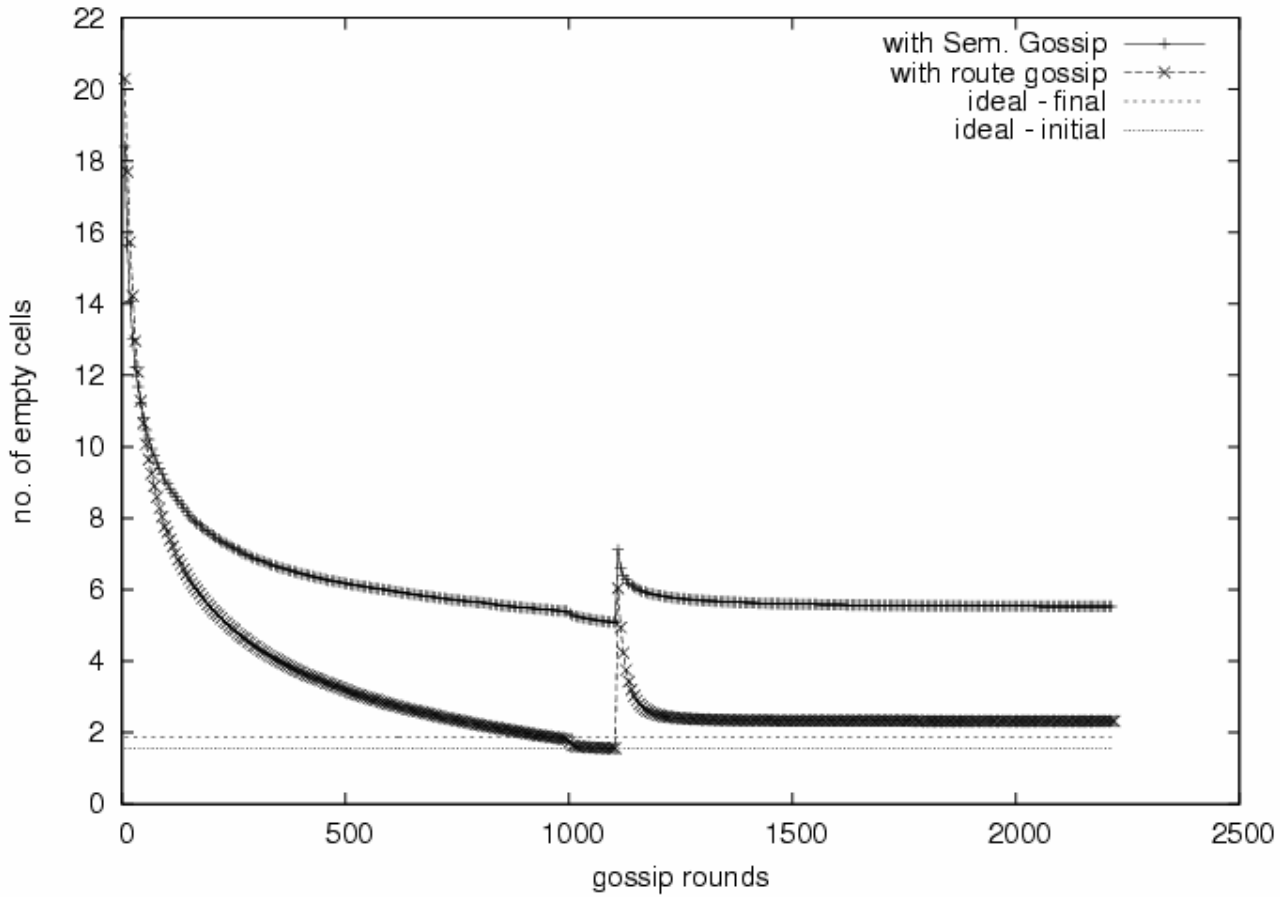
Static configuration



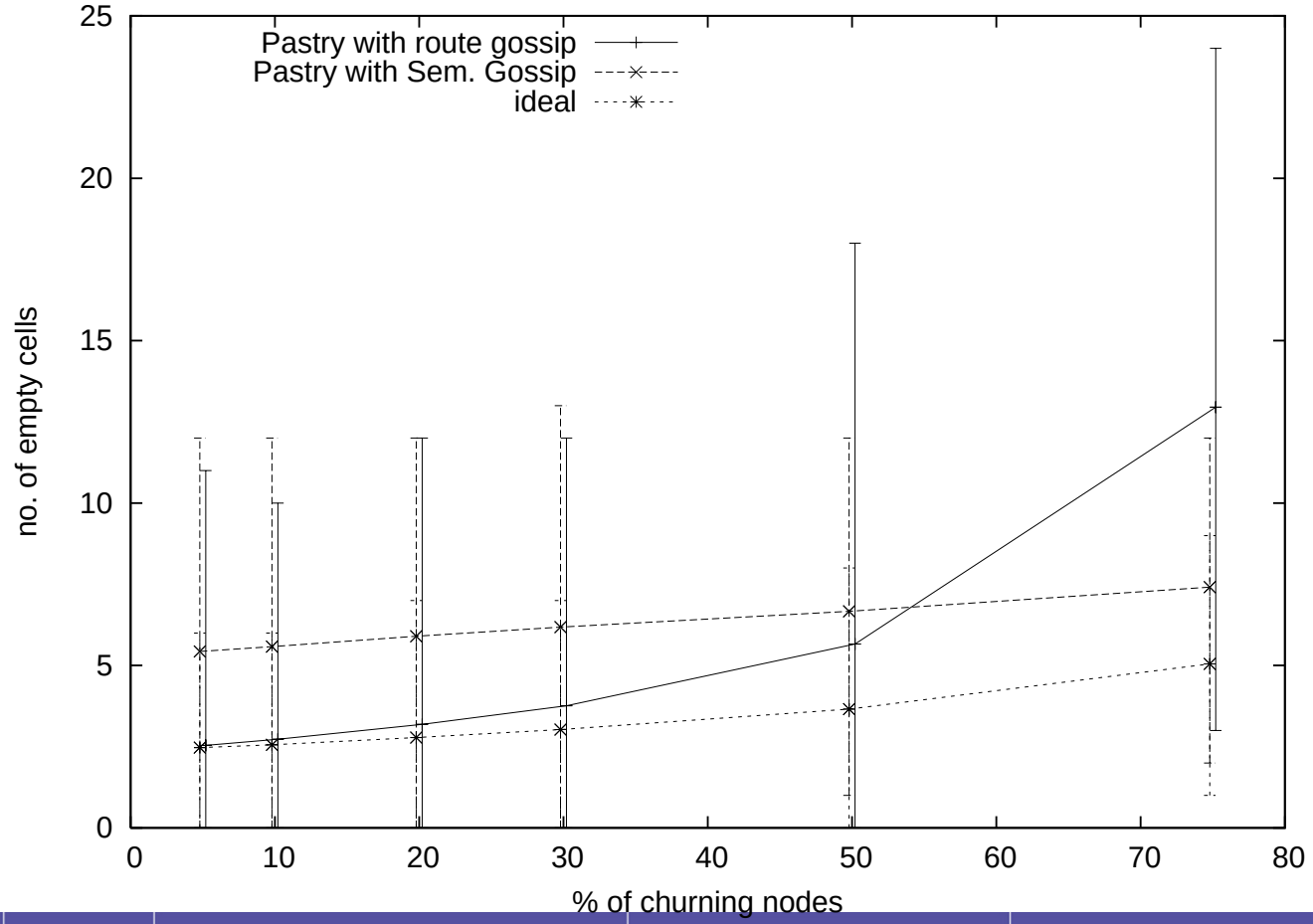
Static configuration: overhead



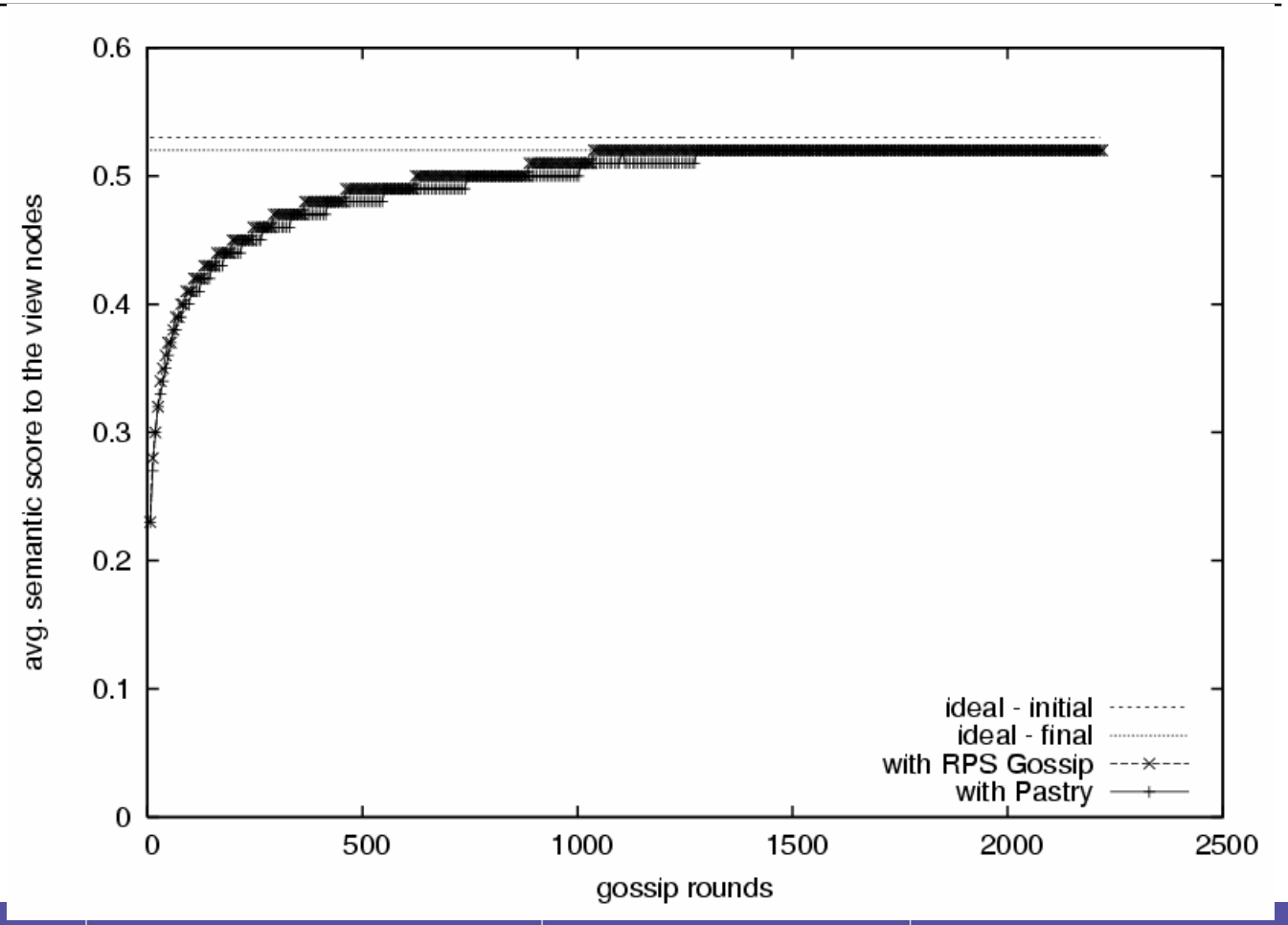
Dynamic scenario: failures



Dynamic configuration



Dynamic scenario: failures



Conclusion

- Many P2P overlays providing various functionalities
- Relevant to have them cohabiting on the same physical network: how to leverage this
  - Better performance at the price of an increased maintenance
  - Similar performance for a lower overhead.
- Open issues
  - Application adaptation
  - Are the resulting overlays exhibit the same properties wrt failures, dynamics, functionalities?
    - Ex: resilience to churn of RPS? (correlated views)
    - Ex: proximity neighbour selection in Pastry
  - To what extent this can be generalized?