Gossip-based networking

Anne-Marie Kermarrec

ASAP (As Scalable As Possible) Research Group

INRIA Rennes, France

Gossip (Wikipedia)

- **Gossip** consists of casual or idle talk of any sort, sometimes (but not always) slanderous and/or devoted to discussing others.

  While gossip forms one of the oldest and (still) the mos
  and sharing
  reputation fo
  other variatio
  transmitted...

Reliable way of spreading information

Epidemic (Wikipedia)

- In epidemiology, an **epidemic** is a disease that appears as new cases in a given human population, during a given period, at a rate that substantially exceeds what is "expected".
- Non-biologica

The term is ... o

refer to wide ... s

Efficient way of spreading something

Gossip/epidemic in distributed computing

Replace people by computers (nodes or peers), words with data

We retain from

- Gossip: peerwise exchange of information
- Epidemic: wide and exponential spread

Refer to gossip in the reminder of the talk

The gossip revival

- Dramatic shift in scale (size, data, spread)
- Dynamic nature (mobility, versatility, ...) leads to near continuous changes

Lead to a fair amount of uncertainty

Gossip-based networking

- Peer to peer communication paradigm
- Probabilistic nature
- Eventual convergence

Gossip-based protocols

- Some form of randomization
- Periodic exchange of information
- Bounded messages
- Strengths
  - Simplicity
  - Emergent structure
  - Convergence
  - Robustness
- Weaknesses
  - Overhead
  - Hard to cope with malicious behavior

1001 ways of leveraging gossiping

Content-based search
Vicinity[Voulgaris & Steen,Euro-Par 05]
VoroNet [Beaumont & al, IPDPS 07]
RayNet[Beaumont & al, OPODIS 07]

Consistency Management
[Demers &al, PODC

Epidemic dissemina
Bimodal Multicast [Birman&al, A
[Kermarrec&al, IEEETPD
Lpbcast [Eugster&al DSN01, AC
ream[Patel & al, NCA 2006]

Aggregation
[Jelasity&al, ACM TOCS 05]
Astolabe [Birman & al, 2003]

Slicing
[Jelasity, Kermarrec, P2P06]
S07]

Overlay maintenance
bcast [Eugster & al,ACM TOCS 03]
Cyclon[Voulgaris& al, 2005]
Newscats[Jelasity & al, 2003]

Secure Sampling
Brahms [Bortnikov & al, 08]

Publish-
Sub-2-Sub [Voula
Tera[Bald

Clustering
cinity, Jstream, Tman

Secure streaming
BAR Gossip [Li & al, OSDI06]

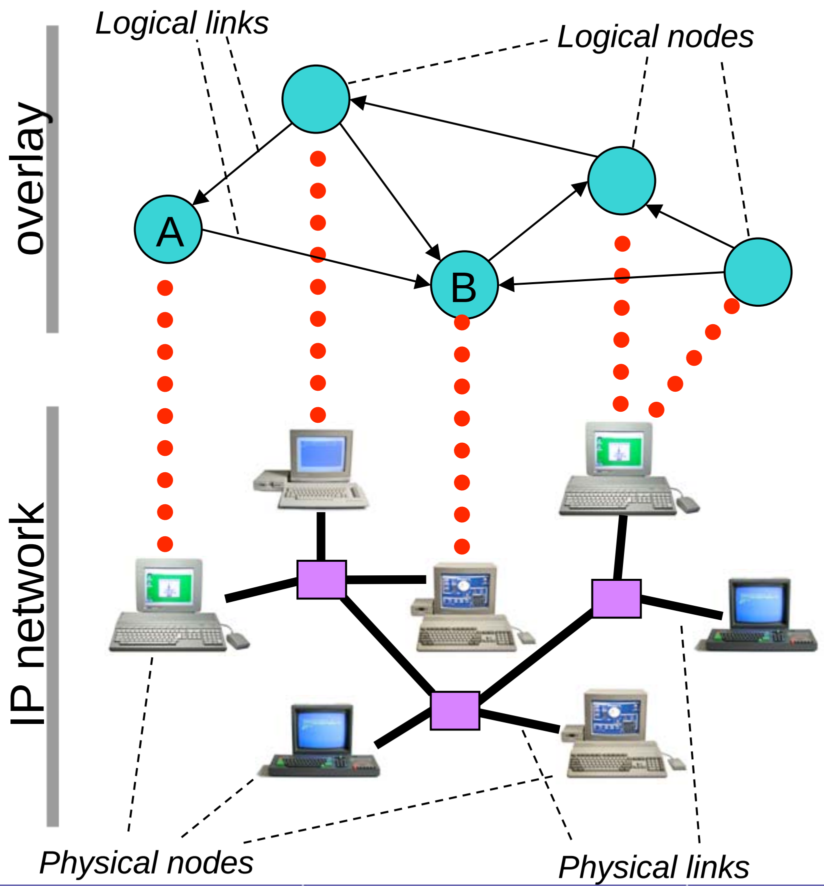
Agenda

1. Overlay maintenance: Unstructured networks
   Random Peer Sampling
2. Loose structuring: clustering
   Biased Peer Sampling
3. Enabling efficient routing
   Kleinberg-like Peer Sampling
4. Gossip-based structured networks: for which applications?
   1. Distributed Slicing
   2. Content-based pub-sub systems (Sub-2-sub)
   3. Range queries in multidimentional spaces (Voronet/Raynet)

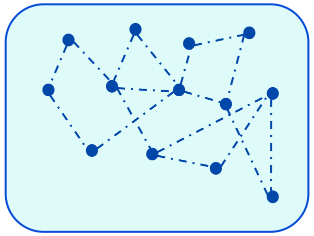
P2P overlay: which structure?

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE
INRIA
RENNES

Peer to peer overlay networks

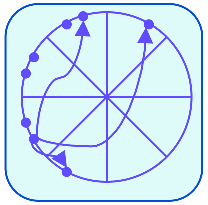


Peer to peer overlay networks



Routing capabilities

Flexibility



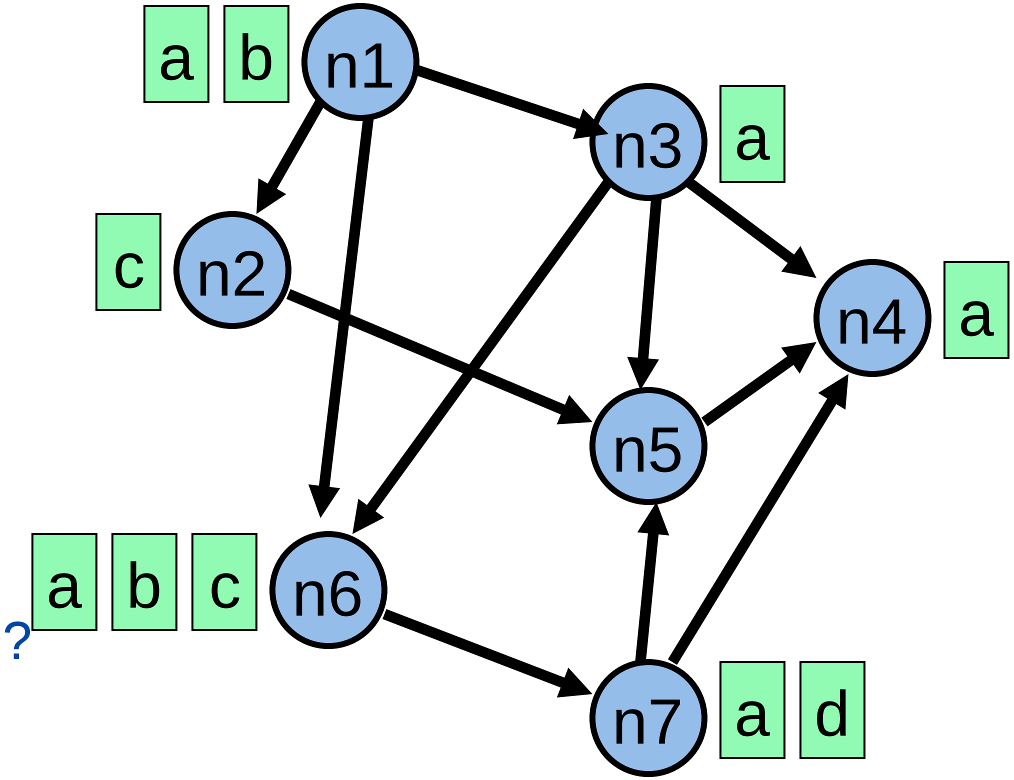
Unstructured networks

Fully structured networks

- Provide various functionalities/performance: search, dissemination, etc
- Common characteristics
  - Self-organizing
  - Local knowledge
  - Resource aggregation
- Resulting properties
  - Scalability
  - Resilience to churn

Example: Search in peer to peer overlays

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



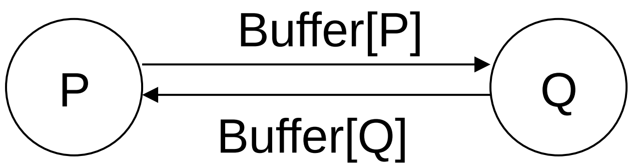
Impact of the structure on search

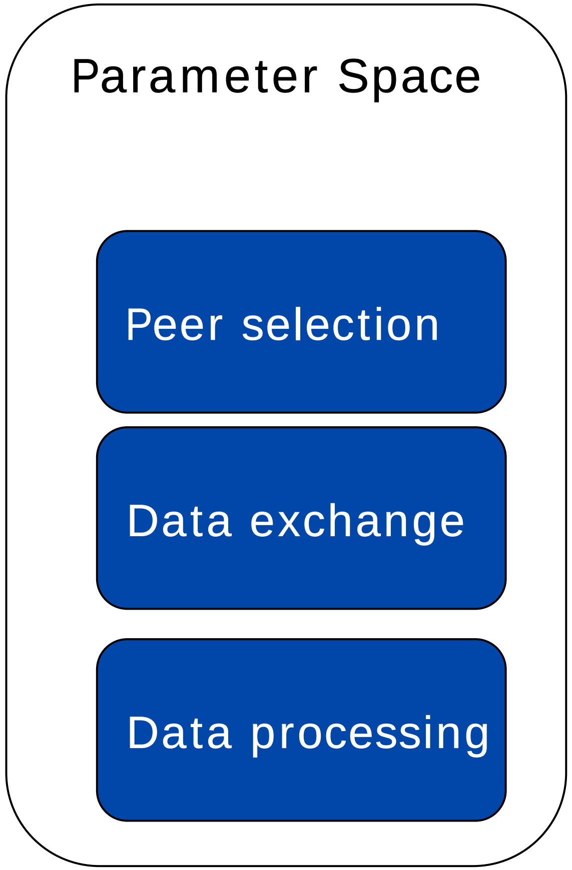
- Several ways of organizing a P2P overlay network
  - Search techniques: flooding versus routing
  - Expressiveness
  - Completeness
- Structured P2P overlay: DHT functionality
  - Support for exact search
- Unstructured gossip-based P2P overlays
  - Support for keyword-based search or range queries
- Weakly structured gossip-based overlays
  - Improve search efficiency upon fully unstructured overlays

A generic gossip-based substrate

Gossip-based generic substrate

- Each node maintains a set of neighbours (c entries)
- Periodic peerwise exchange of information
- Each process runs an active and passive threads





A generic gossip-based substrate

Active thread (peer P)		Passive thread (peer Q)
(1) selectPeer (&Q);		(1)
(2) selectToSend(&bufs);		(2)
(3) sendTo(Q,bufs);	→	(3) receiveFrom(&P,&bufr);
(4) -		(4) selectToSend(&bufs);
(5) receiveFrom(Q,&bufr);	←	(5) sendTo(P,bufs);
(6) selectToKeep(cache,bufr);		(6) selectToKeep(cache,bufr);
(7) processData(cache)		(7) processData(cache)

Dissemination

Peer selection

Data exchange

Data processing

K random

Message

Broadcast protocol (Lpbcast)

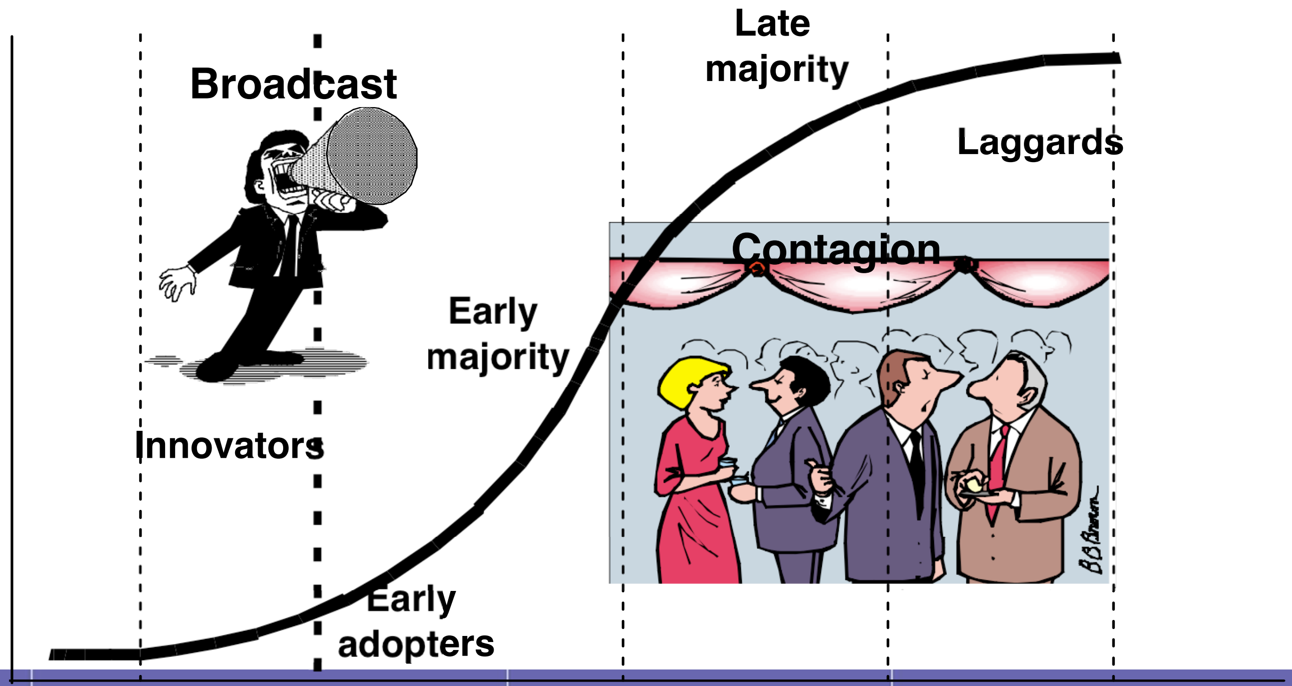
Overlay maintenance

	LpbCast	Cyclon	T-man
Peer selection	Random	Oldest	Closest
Data exchange	List of neighbours	½ List of neighbours	List of neighbours
Data processing	Random merging	Age-based merging	Proximity Based merging

Decentralized computations

	Aggregation	System size estimation	Slicing
Peer selection	Random	Random	Random
Data exchange	value	value	Attribute value Random value
Data processing	Aggregation Average	Aggregation	Attribute/random matching

Why are we interested in building random graphs?
Illustration through dissemination



Epidemic-based dissemination

- **Goal**:
  - Broadcast reliably a msg to a large number of peers in a decentralized way
  - Proactive technique to tolerate failures
- **System model**
  - *n* processes
  - Each process forwards the message once to *f* (fanout) neighbors, picked up uniformly at random.
  - Alternatively f times to 1 neighbour.
- **Metrics of the success of an epidemic process**
  - Proportion of infected processes

    $$Y_r = Z_r / n$$

    Z_r is the number of infected processes prior to round r
  - Probability of atomic "infection"

    $$P(Z_r = n)$$

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE
INRIA
RENNES

Proportion of infected processes

Large system of size n

Probabibility that the epidemic catches (1-p_{ext})

Proportion of processes eventually contaminated

$\pi = 1 - e^{-\pi f}$ where f is the fanout

Independent of n, a fixed average of descendants will lead to the same proportion of infected processes

Probability of atomic infection

Erdos/Renyi examine final system state, the system is represented as a graph where each node is a process, there is an edge from n_1 to n_2 if n_1 is infected and chooses n_2 .

An epidemic starting at n_0 is successful if there is a path from n_0 to all members. If the fanout is $\log(n) + c$, the probability that a random graph is connected is

$$p(connect) = e^{-e^{-c}}$$

Other measures

- Latency of infection

[Bollobas, Random Graphs, Cambridge University Press, 2001]

Logarithmic number of rounds

$$R = \frac{\log(n)}{\log(\log(n))} + O(1)$$

- Resilience to failure

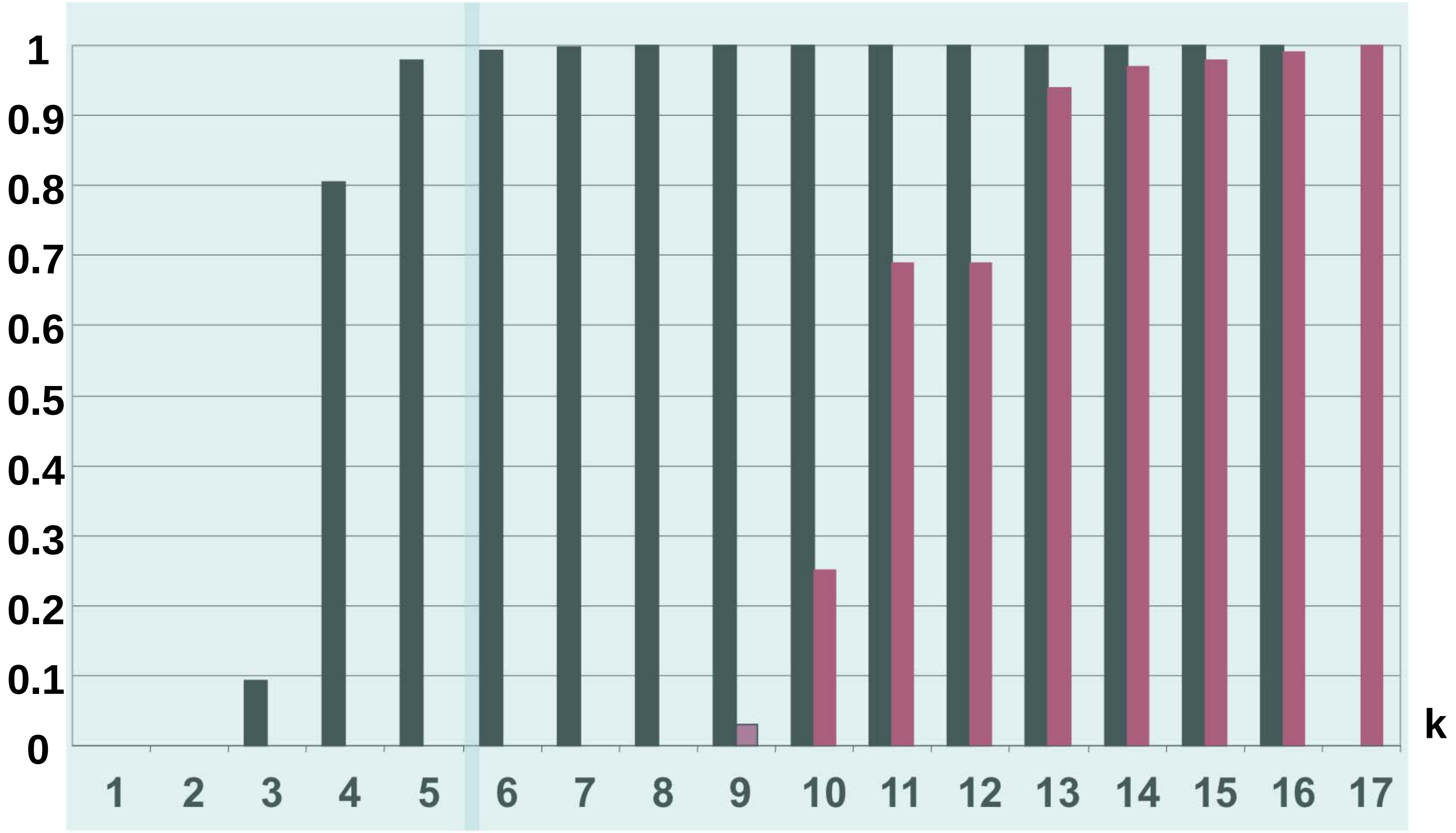
[KMG, IEEE Tpds 14(3), Probabilistic reliable dissemination in Large-scale systems, 2003]

$$k = (n / n')[\log(n') + c + O(1)]$$

The log(n) magic

- Simple dissemination algorithm
- Probabilistic guarantees of delivery
- Each node forwards the message to *f* nodes chosen uniformly at random
  - If *f=O(log(n)),* “atomic” broadcast whp
  - Result is valid if the fanout for each peer is on average *log(N)* + *c*, whatever the degree distribution.
- Relate probability of reliable dissemination and proportion of failure
  - Set parameters

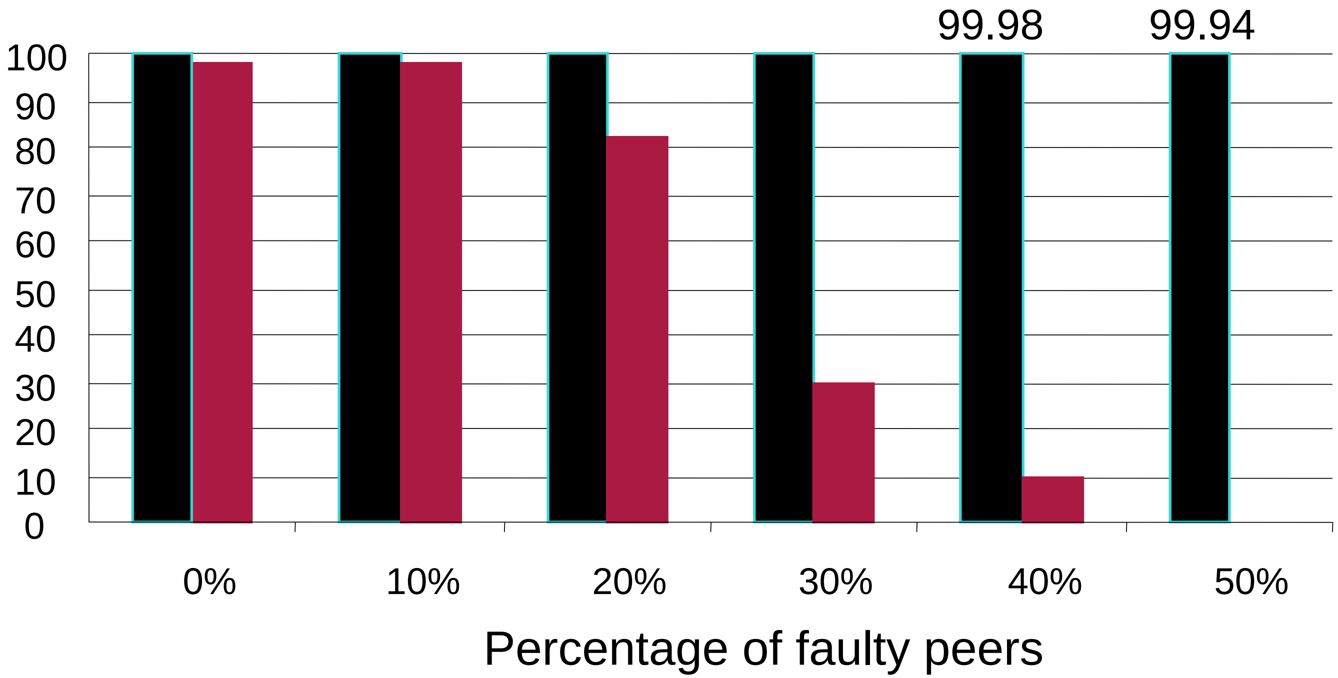
Performance (100,000 peers)



Proportion of “atomic”broadcast

Proportion of connected peers in non “atomic” broadcast

Failure resilience (100,000 peers)



Proportion of "atomic" broadcast

Proportion of connected peers in non "atomic" broadcast

The relevance of gossip

- Introduces implicit redundancy
- Flexible and simple protocols
- Overhead
  - Small messages
  - Application to maintenance, monitoring, etc...

Differ in the choice of gossip targets and information exchanged

Gossip-based dissemination

Peer selection — K random

Data exchanged — Message

Data processing

Dissemination
Data = msg to broadcast

Each process gossips one message once

How can we achieve Random sampling?

Achieving random topologies through gossiping

- Epidemic dissemination
- Distributed computations (average)
- System size estimation

The peer sampling service

- How to create a graph upon which applying gossip-based dissemination?... **By gossiping around**
- **Goal:**
  - Create an overlay network
  - Provide each peer with a **random** sample of the network in a decentralized way
- **Means:** gossip-based protocols
  - What data should be gossiped?
  - To whom?
  - How to process the exchanged data?
- Resulting "who knows who" graphs: overlay
  - Properties (degree, clustering, diameter, etc.)
  - Resilience to network dynamics
  - Closeness to random graphs

The peer sampling service

- Creates unstructured overlay network topologies
- Interface
  - *Init()*: service initialization
  - *GetPeer():* returns a peer address, ideally drawn uniformly at random

Properties

- View: local knowledge of the system
  - Continuously updated to reflect the dynamics of the system
  - Provides a sample of the network

- Generic framework [GJKvSV, ACM TOCS 2007]
  - Covers existing gossip-based membership protocols: Lpbcast [EGKK01], Newscast[JKvS03], Cyclon[VDvS03]
  - Explore the design space
    - Evaluation of the "randomness" of the sampling
  - Interestingly enough: generic enough for many other protocols

System model

- System of n peers
- Peers join and leave (and fail) the system dynamically and are identified uniquely (IP @)
- Epidemic interaction model:
  - Peers exchange some membership information periodically to update their own membership information
  - Reflect the dynamics of the system
  - Ensures connectivity
- Each peer maintains a view (membership table) of c entries
  - Network @ (IP@)
  - Age (freshness of the descriptor)
  - Each entry is unique
  - Ordered list
- Active and passive threads on each node

Protocol

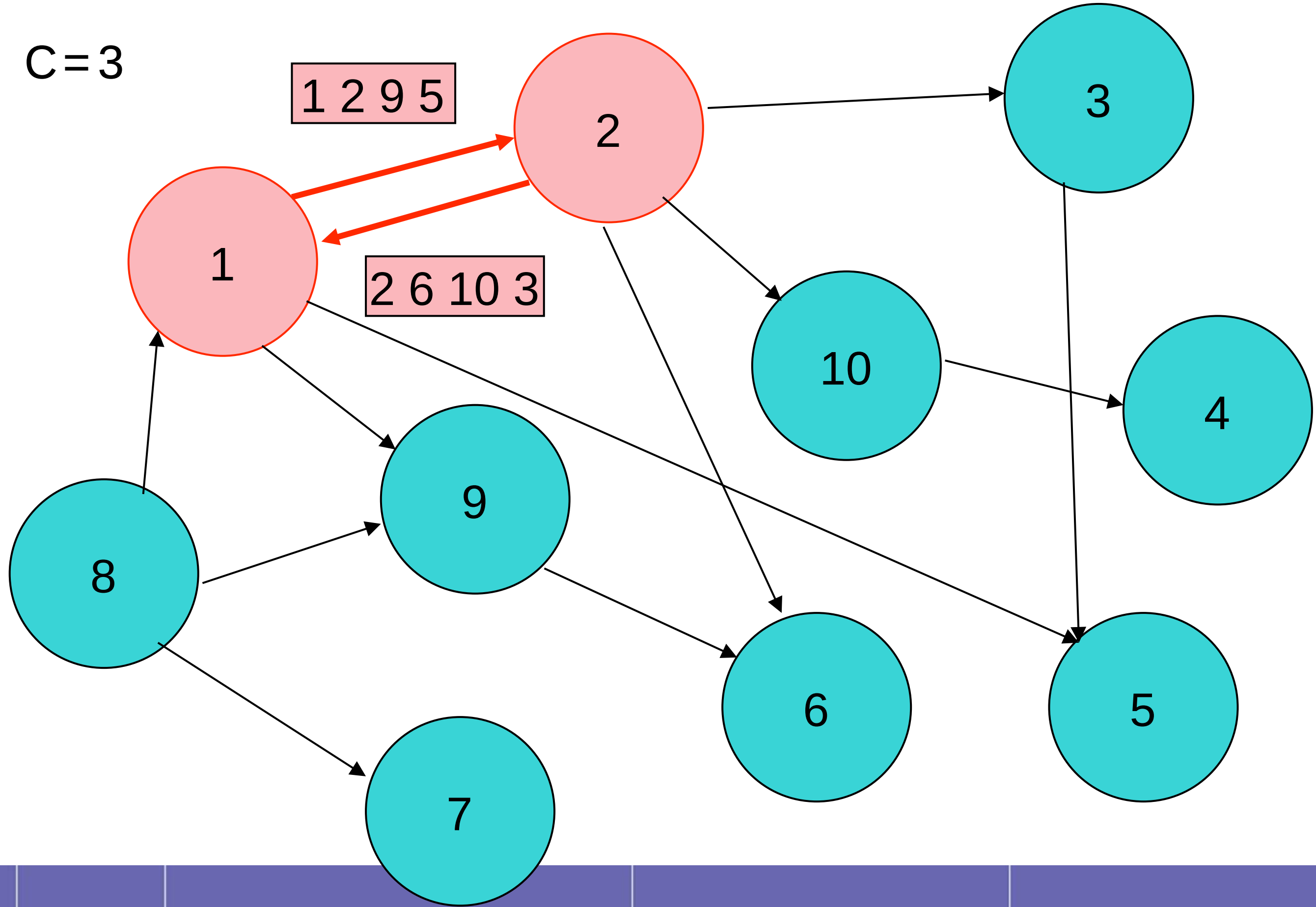
Active thread

```
Wait (T time units)
P <- selectPeer()
if push then
    myDescriptor <- (my@,0)
    buffer <- merge (view,
    {myDescriptor})
    send buffer to p
else send{} to p //triggers response
if pull then
    receive view from p
    buffer <- merge(view_p, view)
    view <- selectView(buffer)
view_p<-increaseage(view_p)
```

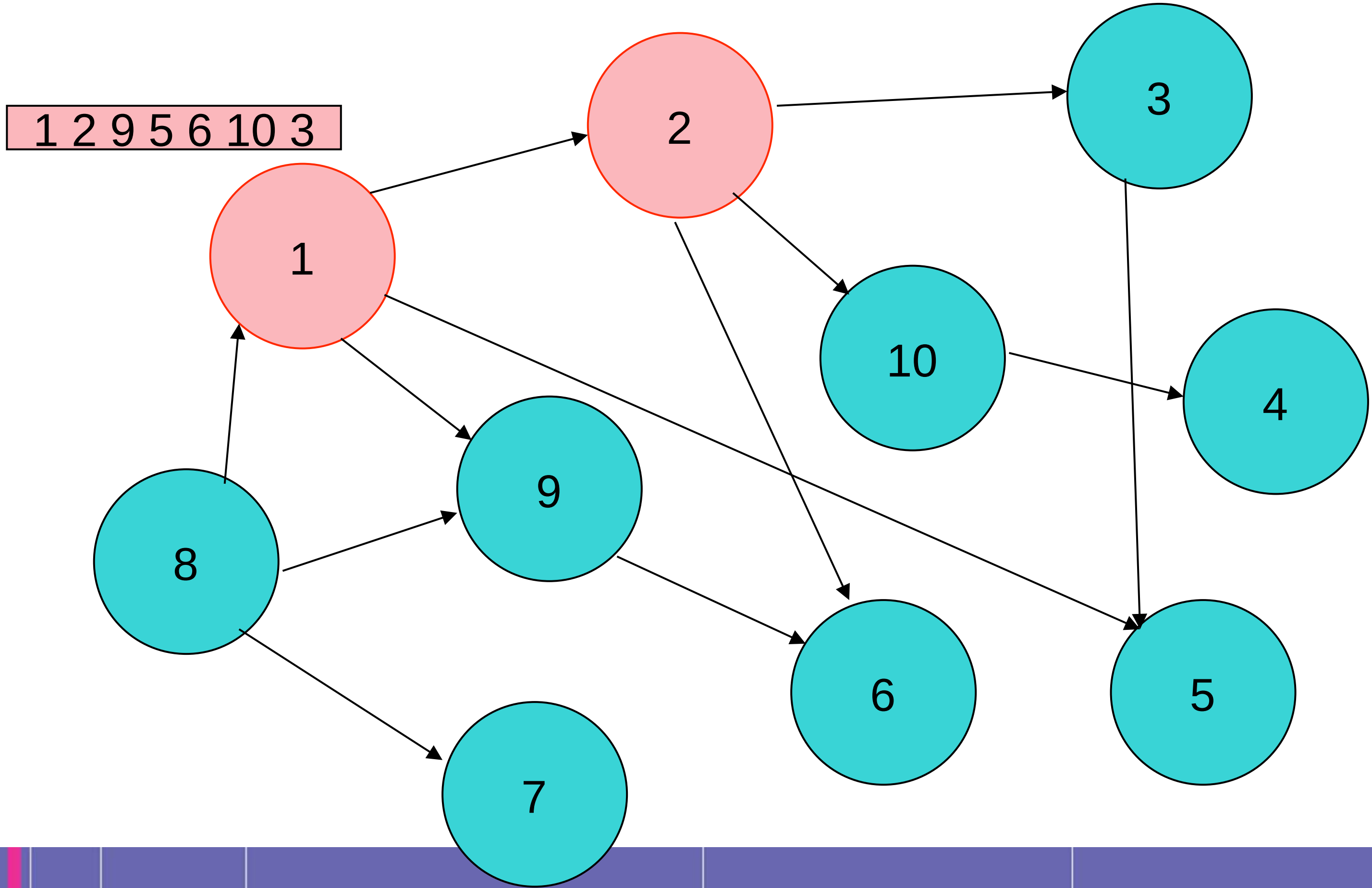
Passive Thread

```
(p,view_p) <- waitMessage()

if pull then
    myDescriptor <-(my@,0)
    buffer <-merge(view,
    {myDescriptor})
    send buffer to p

View_p <-increaseage(view_p

buffer <- merge(view_p, view)
View <-selectView(buffer)
```

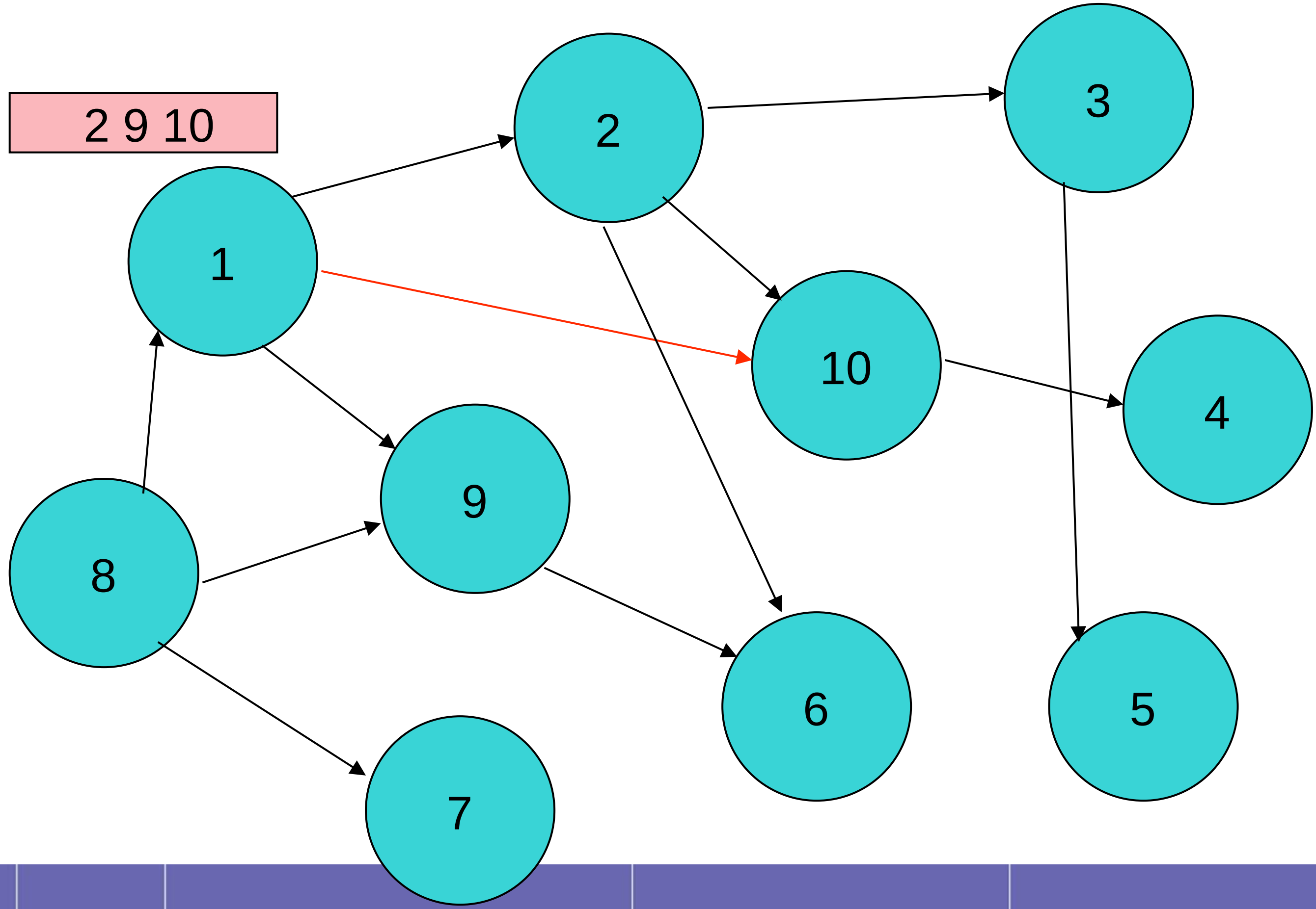
Example: Gossip-based generic protocol



Example: Gossip-based generic protocol



Example: Gossip-based generic protocol



Design space

- Periodically each peer initiates communication with another peer
- **Peer selection**
- **Data exchange (View propagation)**
  - How peers exchange their membership information?
  - What do they exchange?
- **Data processing (View selection)**: Select (c, buffer)
    - c: size of the resulting view
    - Buffer: information exchanged

Design space: data exchange

- **Buffer (h)**
  - initialized with the descriptor of the gossiper
  - contains c/2 elements
  - ignore h “oldest”
- **Communication model**
  - Push:buffer sent
  - Push/Pull: buffers 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: Data processing

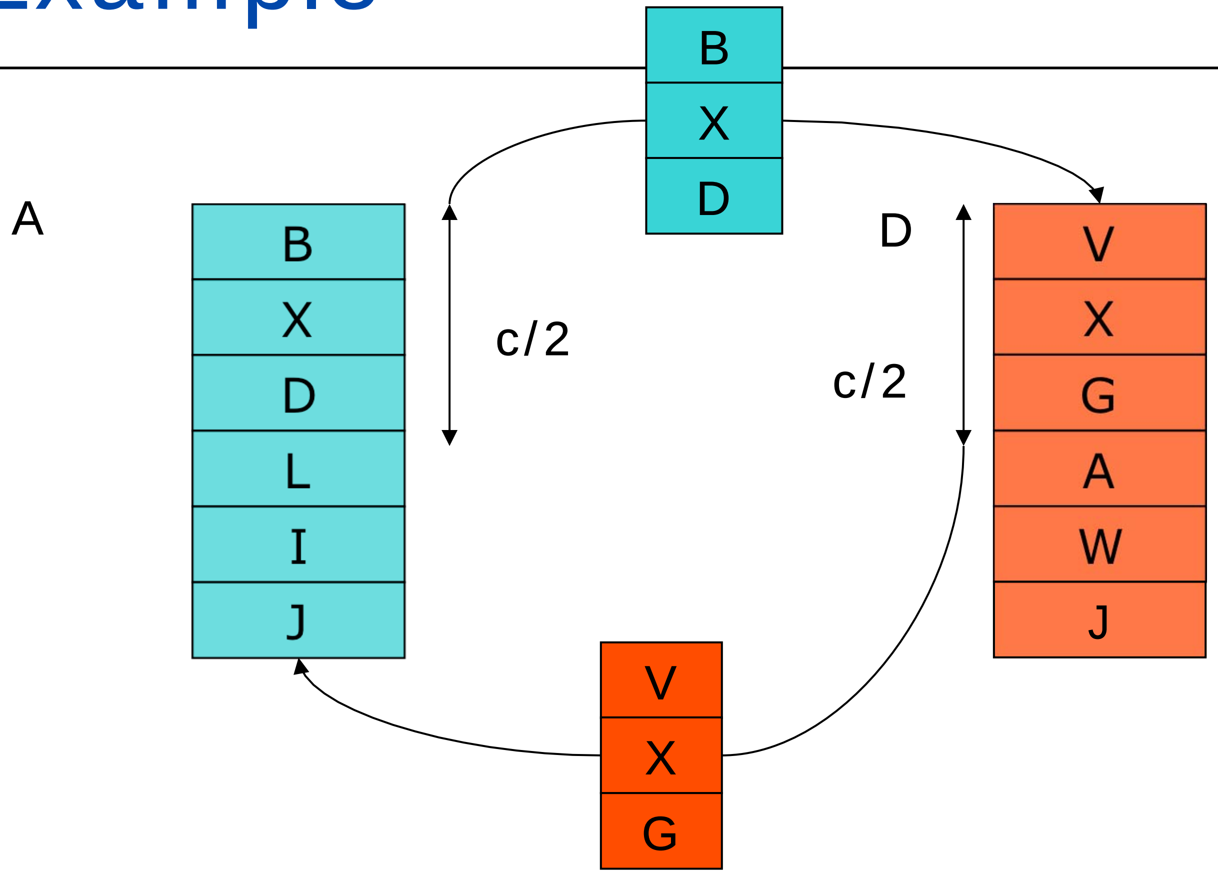
c: size of the resulting view
H: self-healing parameter

- Select(c,h,s,buffer)

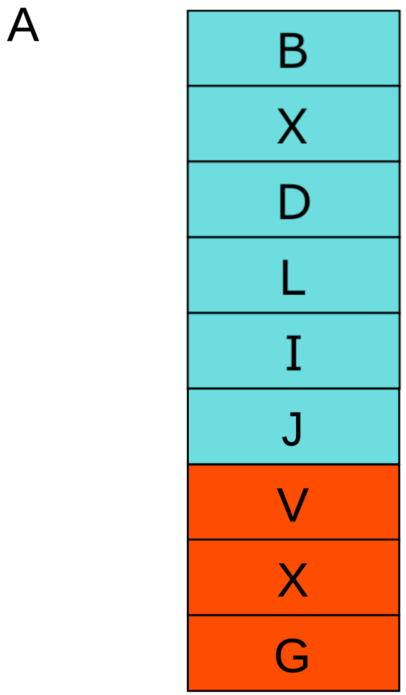
1. Buffer appended to view
2. Keep the freshest entry for each node
3. h oldest items removed
4. s first items removed (the one sent over)
5. Random nodes removed

- 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

Example



Example



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

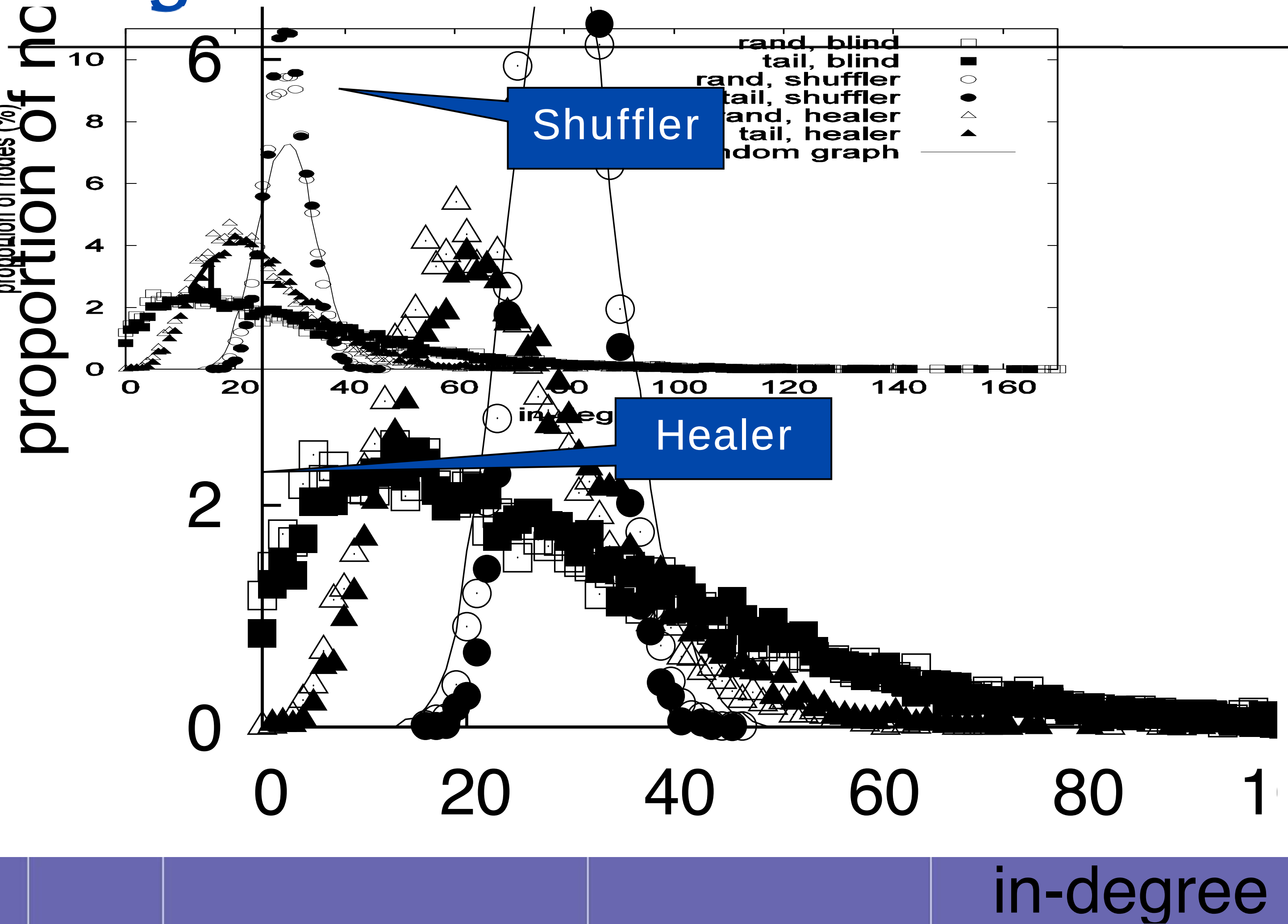
Resulting graphs properties

- Relationship « who knows who »
  - Highly dynamic
  - Capture quickly changes in the overlay networks
- Experimental study= lattice, random, growing networks
- Metrics
  - Degree distribution
  - Average path length
  - Clustering coefficient
- Healer (h=c/2, s=0)
- Shuffler (h=0, s=c/2)

Degree distribution

- Out degree = c (30) in 10.000 node system
- Distribution of in-degree
  - Detect hotspot and bottleneck
  - Load balancing properties
- Convergence
  - Self-organization ability irrespective of the initial topology

Degree distribution



Degree distribution

- Convergence, even in growing scenario
- View selection parameter matters
- Shuffler and healer result in lower standard deviation for opposite reasons
  - Shuffler
    - Controlled degree distribution
    - New links to a node are created only when the node itself injects its own fresh node descriptor during communication.
  - Healer
    - Short life time of links
    - When a node injects a new descriptor about itself, this descriptor is copied to other nodes for a few cycles.
    - Later all copies are removed because they are pushed out by new links injected in the meantime

Average path length

- Shortest path length between a and b
  - minimal number of edges required to traverse in the graph to reach b from a
- Defines a lower bound on the time and costs of reaching a peer.
- Small average path length essential for scalability

Average path length

- Results
  - all protocols result in a very low path length.
  - large S values are the closest to the random graph.

Clustering coefficient

- Results
  - clustering coefficient also converges
  - controlled mainly by H.
    - Large value of H result in significant clustering, where the deviation from the random graph is large.

      large part of the views of any two communicating nodes overlap right after communication (freshest entries).
    - Large values of S, clustering is close to random

Peer sampling service: Summary

- Experimental study
  - How random are the resulting graphs?
  - What properties may affect the applications
- Global randomness
  - Best configuration is the shuffler irrespective of the peer selection
- Load balancing
  - Blind performs poorly
  - Best configuration is shuffler while healer performs well
- Fault-tolerance
  - More important parameter is H: the higher the better
  - Shuffler is slow to remove dead links

Overlay maintenance

	LpbCast [Eugster & al, DSN 2001, ACM TOCS 2003]	Newscast [Jelasity & van Steen, 2002]	Cyclon [Voulgaris & al JNSM 2005]
Peer selection	Random	Head	Oldest
Data exchange **Membership data**	List of Neighbours Push	½ List of Neighbours PushPull	½ List of neighbours
Data processing	Random merging	Age-based Merging (Head)	Shuffle

Imposing more structure: biasing the peer sampling

Structuring the network

- T-Man[Jelasity&Babaoglu, 2004]
- Peers optimize their view using the view of their close neighbours
- Ranking function

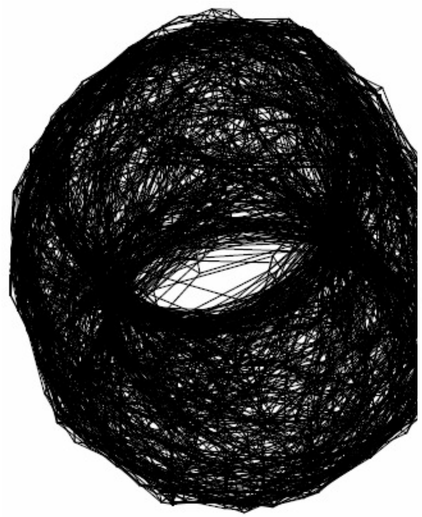
$$R(x, \{y_1, \ldots, y_m\}) \text{ ranks } y_j \text{ strictly lower than } y_i \text{ if } y_i \text{ precedes strictly } y_j \text{ in all possible rankings}$$

- Peer selection
  - Rank nodes in the view according to *R*
  - Returns a random sample from the first half
- Data exchange
  - Rank the elements in the (view+buffer) according to *R*
  - Returns the first *c* elements
- Data processing
  - Keep the c closest

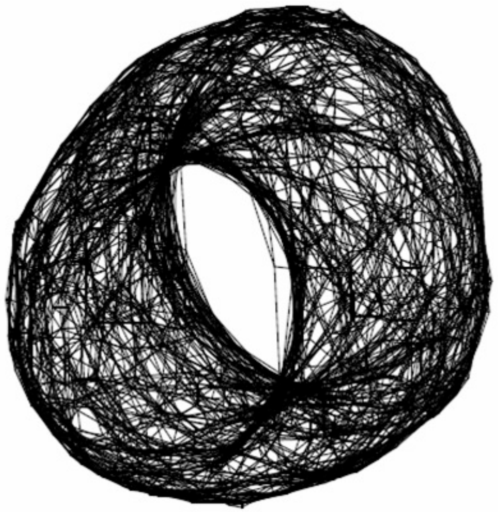
Gossip-based topology management

- Line: d(a,b) =|a-b|
- Ring: interval[0,N], d(a,b)=min(N-|a-b|,|a-b|)
- Mesh and torus: d=Manhattan distance
- Sorting problems: any other application dependent metric

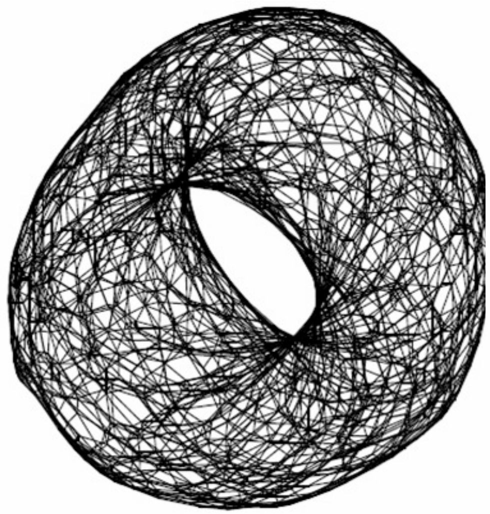
T-man: torus



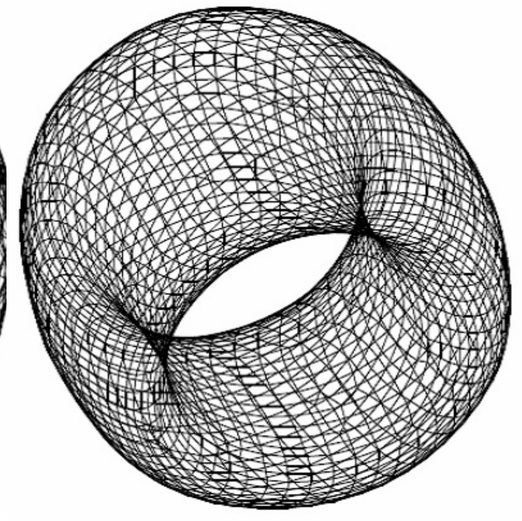
Cycle 3



Cycle 5



Cycle 8



Cycle 15

T-man wrap up

- Generate a large number of structured topologies
- Exponential convergence (logarithmic in the number of nodes)
- Irrespective of the initial topology
- Exact structure

Clustering similar peers

- Vicinity: Introducing application-dependent proximity metric [VvS, EuroPar 2005]
- Two-layered approach
  1. Biased gossip reflecting some application semantic
  2. Unbiased peer sampling service

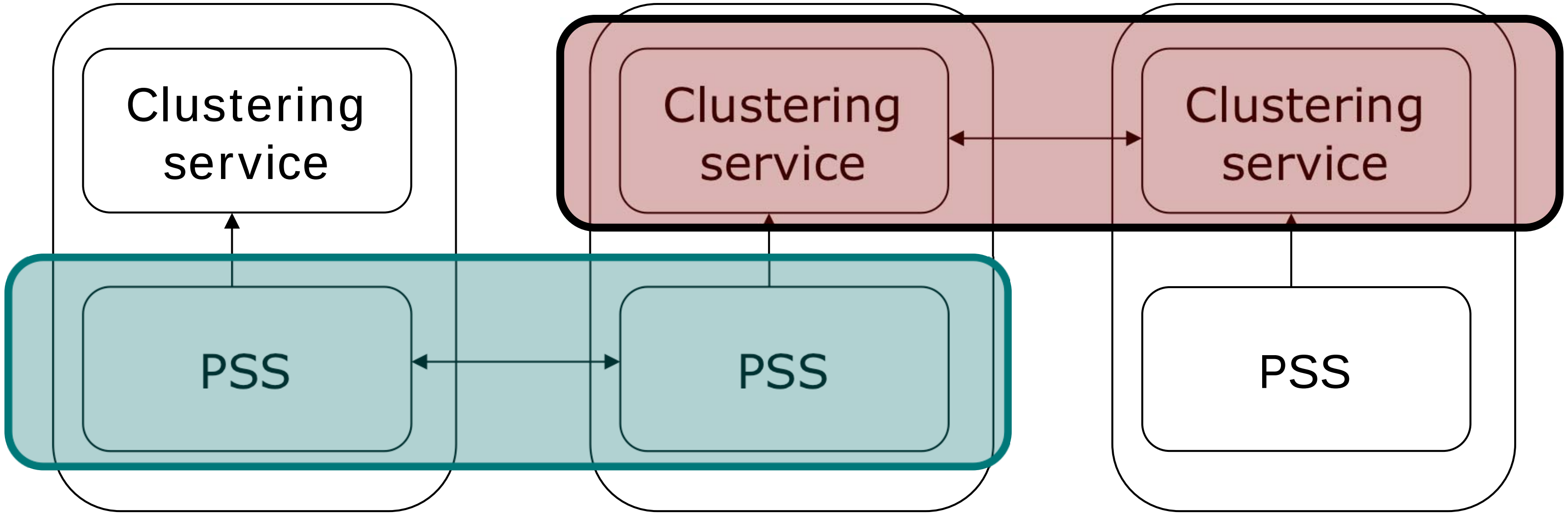
System model

- Semantic view of *l* semantic neighbours
- Semantic proximity function *S(P,Q).*
  - The higher the value of *S(P,Q),* the “closer” the nodes.
  - The objective is to fill P’s semantic view to optimize

$$\sum_{i=1}^{l} S(P,Q_i)$$

Gossiping framework

- Target selection
  - Close peers
  - All nodes are examined: create a “small-world” like structure so that new nodes are discovered.



Gossip parameter setting

- Clustering protocol

  Peer selection
  - tail "oldest timestamp"

  Data exchange
  - aggressively biased,
  - select the g items the closest from semantic and random views

  Data processing
  - select the l closest peers (buffer, semantic and random views)
- Peer sampling service

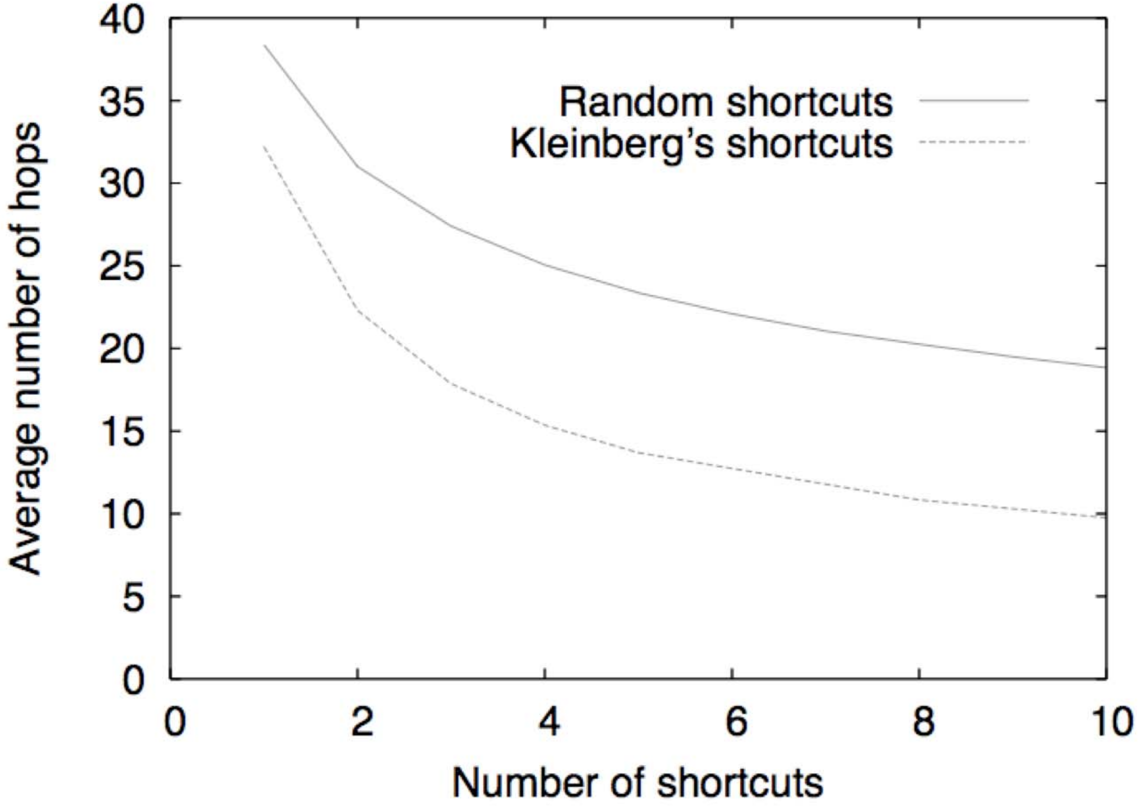
Improving routing: Kleinberg-like peer sampling

Motivation

- Small-world overlay networks
  - Neighbour set: Close + shortcuts
  - Theoretical analysis: Asymptotic bounds on routing performance (random versus Kleinberg's shortcuts)
- Epidemic-based overlay networks
  - Decentralized overlay building and maintenance using gossip-based protocols
  - Practical systems: efficient routing

Epidemic-based small-world networks
Clustering protocols: close neighbours
Peer sampling service: shortcuts

Motivation



Random shortcuts
Kleinberg's shortcuts
Average number of hops
Number of shortcuts
0
5
10
15
20
25
30
35
40
0
2
4
6
8
10

Objective

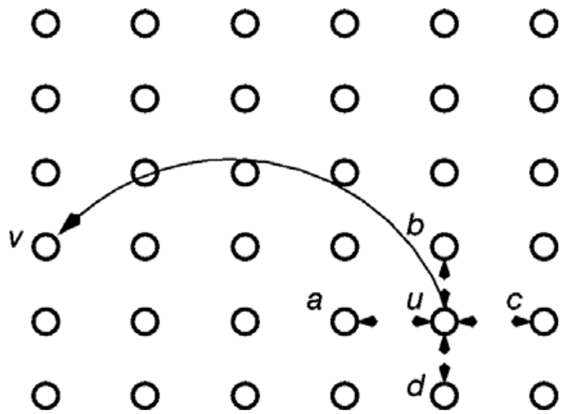
Leveraging theory: how to apply Kleinberg's results to improve upon current epidemic protocols?

Epidemic-based small world networks

Small world overlay network

- Neighbour set
  - Local contacts
  - Shortcuts
- Shortcut selection
  - Random [Watts &Strogatz1998]
  - ` Greedy routing $O(n^{1/3})$
  - Harmonic distribution [Kleinberg 2000]

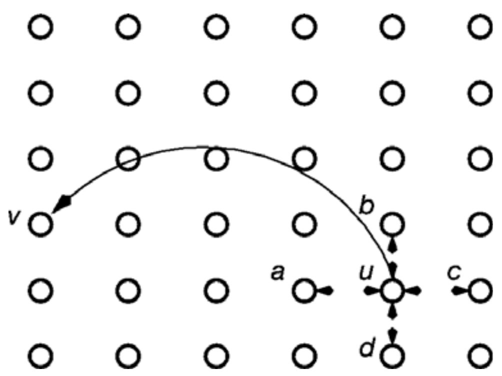
    Greedy routing $O(\log^2(n))$
- Results
  - Asymptotic bounds : Magnitude order of routing performance



Shortcut selection and routing performance

- <u>Random selection</u>
  - Shortcuts picked uniformly at random
  - Greedy routing performance

$$O(n^{1/3})$$



- <u>Kleinberg selection</u>

Selection with probability proportional to distance

$$\delta(B) = \frac{1}{d(A,B)^2}$$

B chosen by A with

$$P = \frac{\delta(B)}{\sum_{B' \in S} \delta(B)}$$

S = Set of peers not neighbour of A

- Greedy routing performance

$$O(\log^2(n))$$

Small-world gossip-based networks

Assume each node has some coordinates in a d-dimensional space

Clustering service
Peer selection: "closest"
Data exchange: c entries
Data Processing: "closest" kept

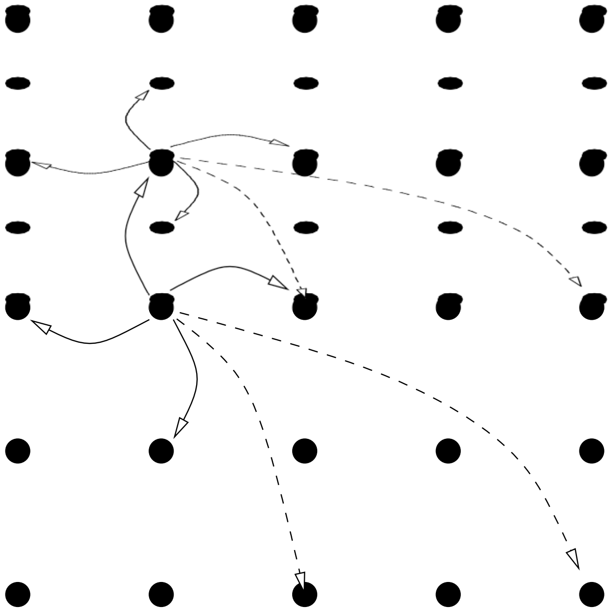
Close links

Peer sampling service
Peer selection: random
Data exchange: c/2 entries
Data Processing: random

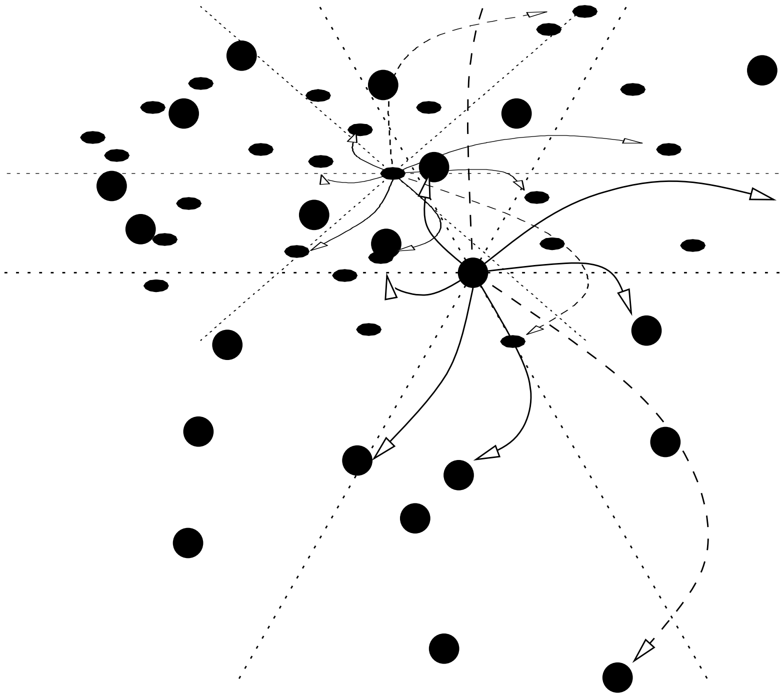
Shortcuts

[Watts &Strogatz1998]

Topologies



Grid, Manhattan distance
Close neighbours: neighbours on the Grid



Grid, Euclidian distance
Close neighbours: one in each wedge

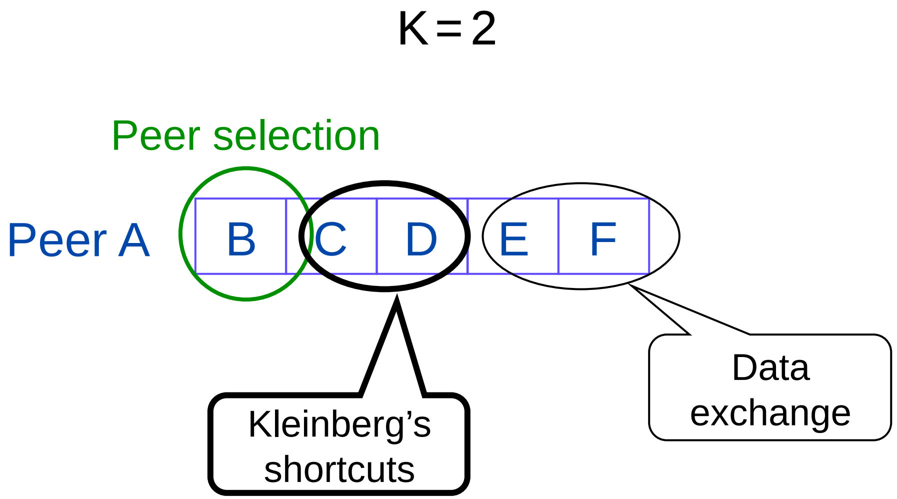
Gossip-based small-world networks

- Leverage theory
- Decentralized selection of neighbours
  - Clustering protocol: local neighbours
  - Peer sampling: shortcuts
- Shortcut selection: peer sampling service
  - Random selection: random peer sampling
  - Kleinberg selection: tune the view so that it matches the Kleinberg's distribution
- What are we interested in?
  - Impact on the routing efficiency
  - Impact on the graph properties

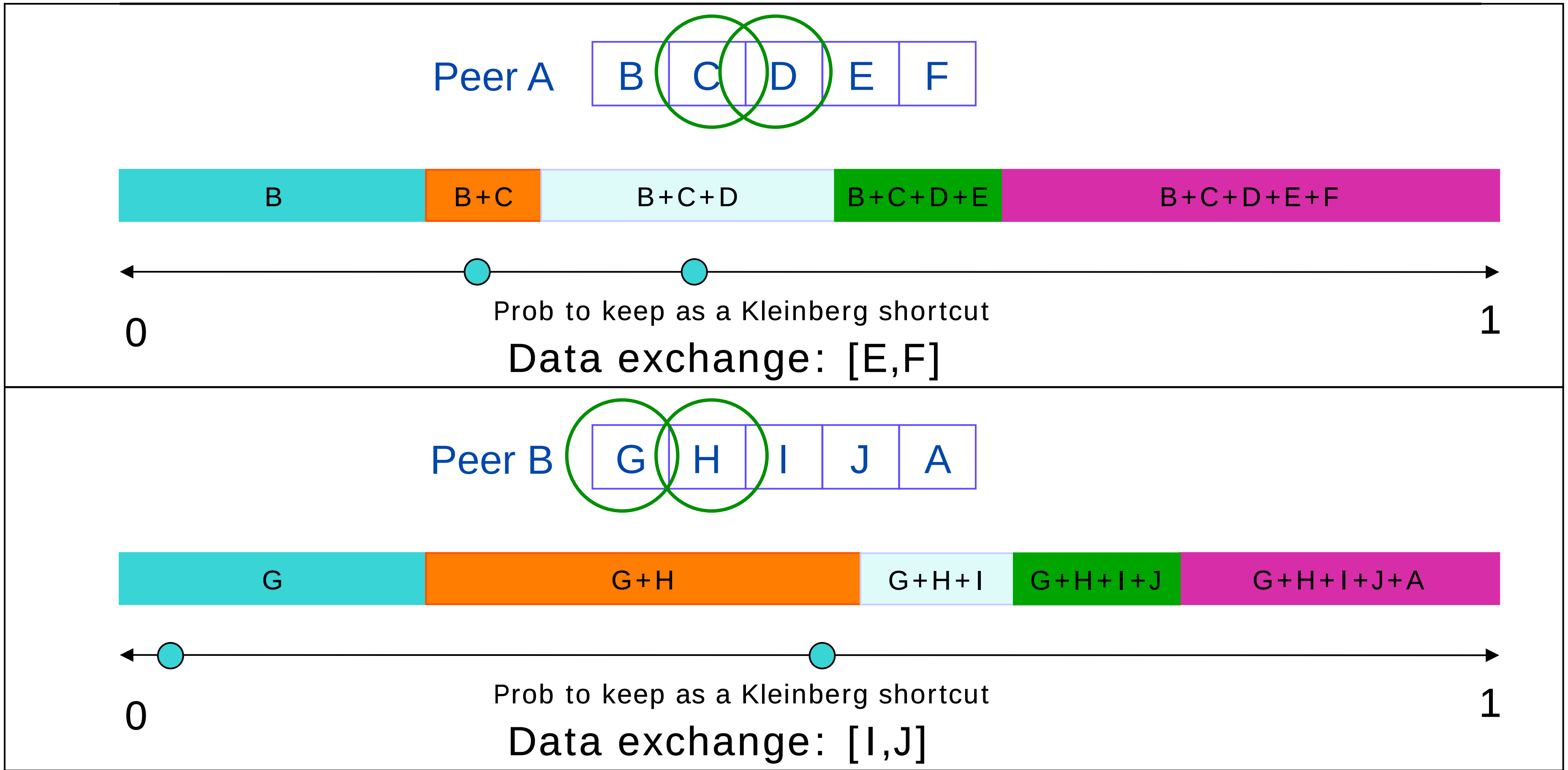
Kleinberg's peer sampling

- Use standard clustering protocol for local neighbours
- Shortcuts: bias Cyclon protocol to approximate Kleinberg's distribution (probability of being kept is $1/d^2$

Peer sampling service
Peer selection: random
Data exchange: k entries,
c-k kept bias by Kleinberg's distribution
Data Processing: c-k entries exchanged

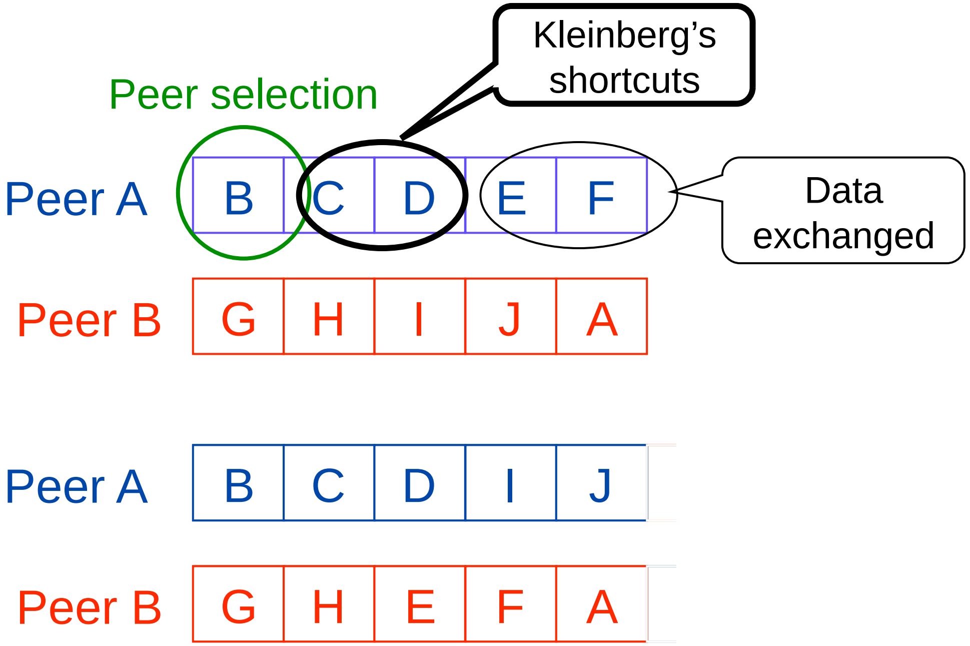


Implementation

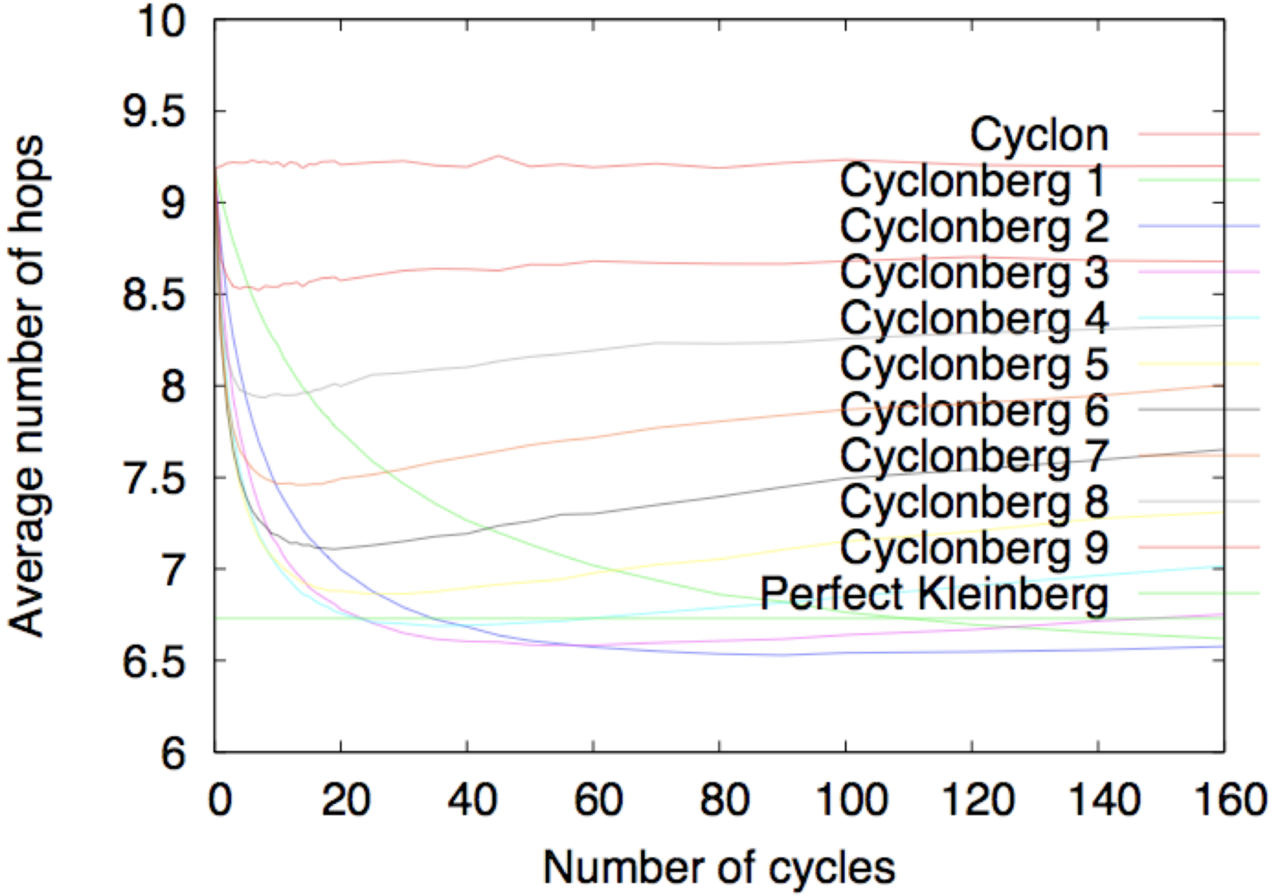


Kleinberg's peer sampling

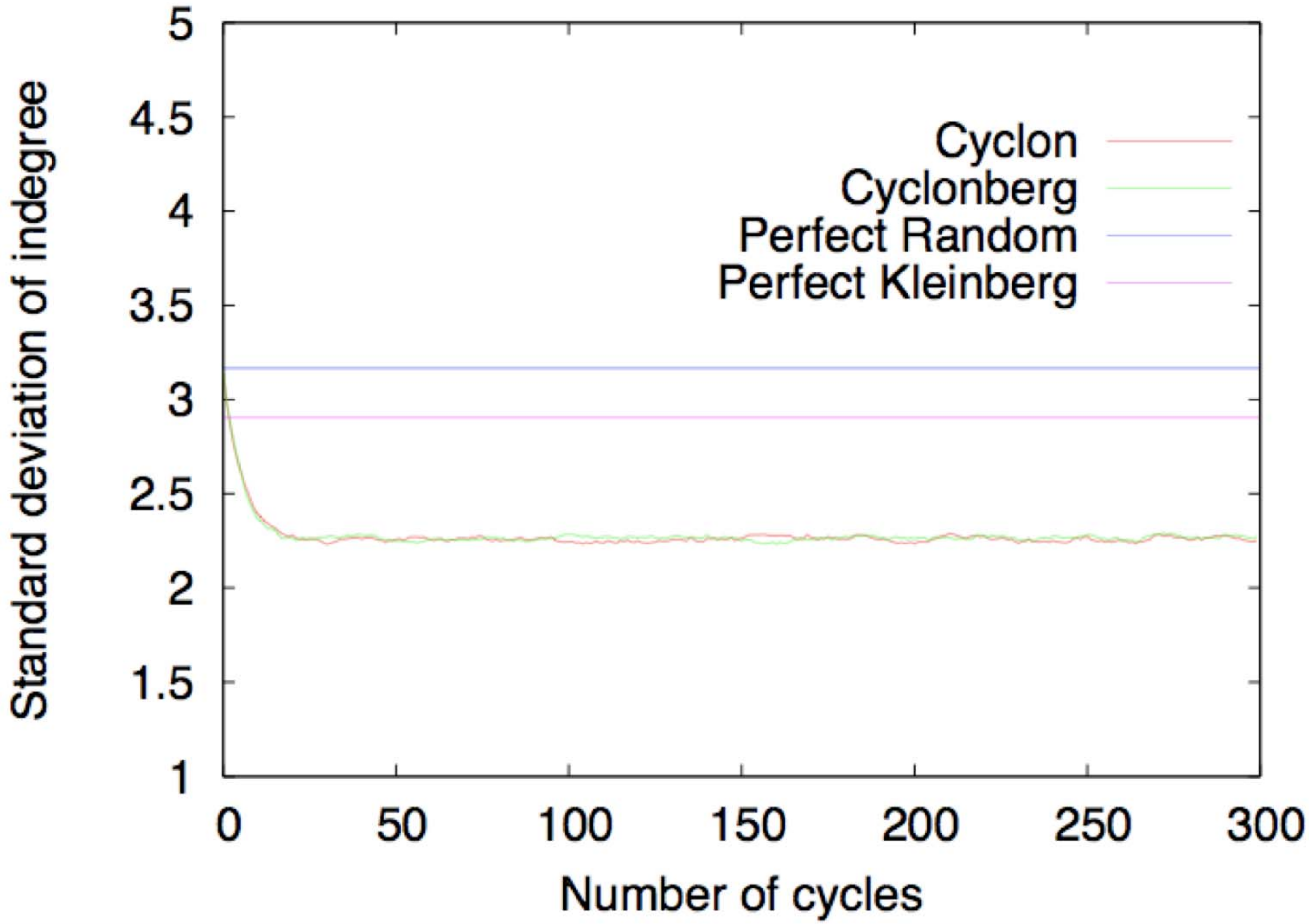
- Example



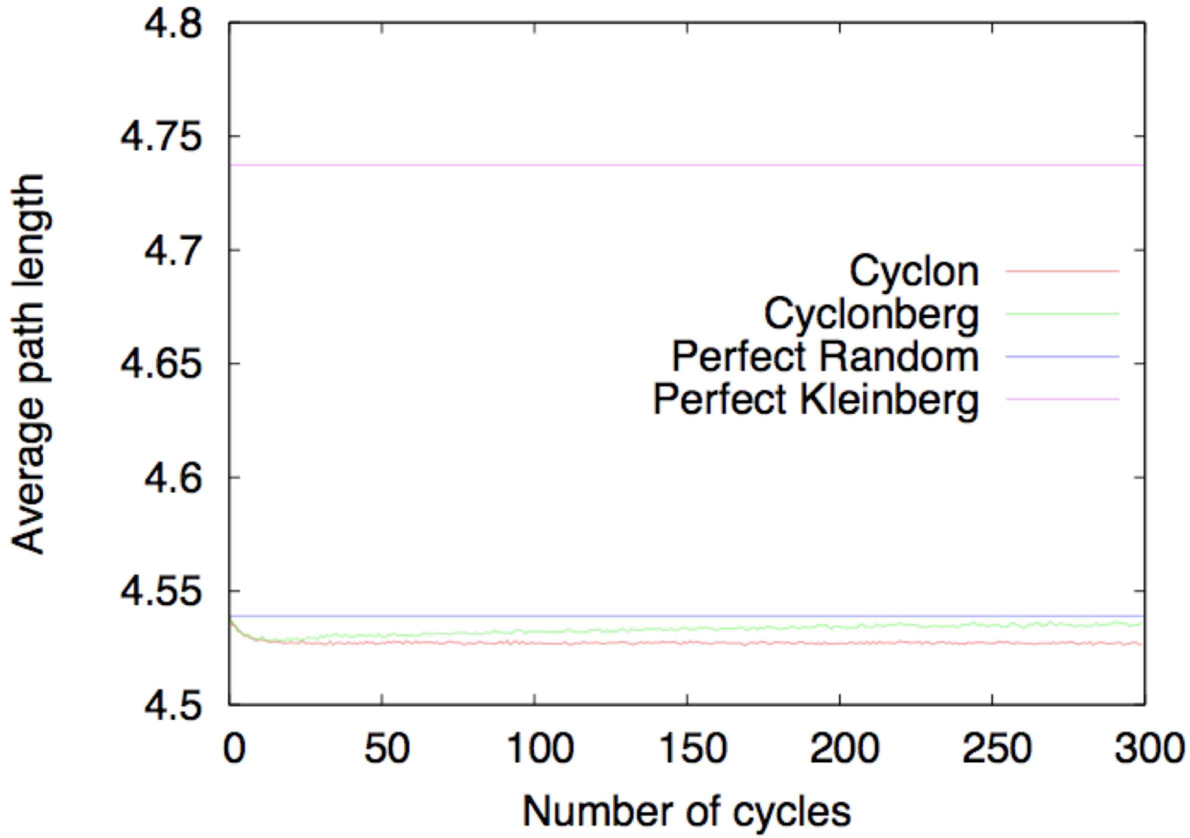
Routing performance



Impact on the degree

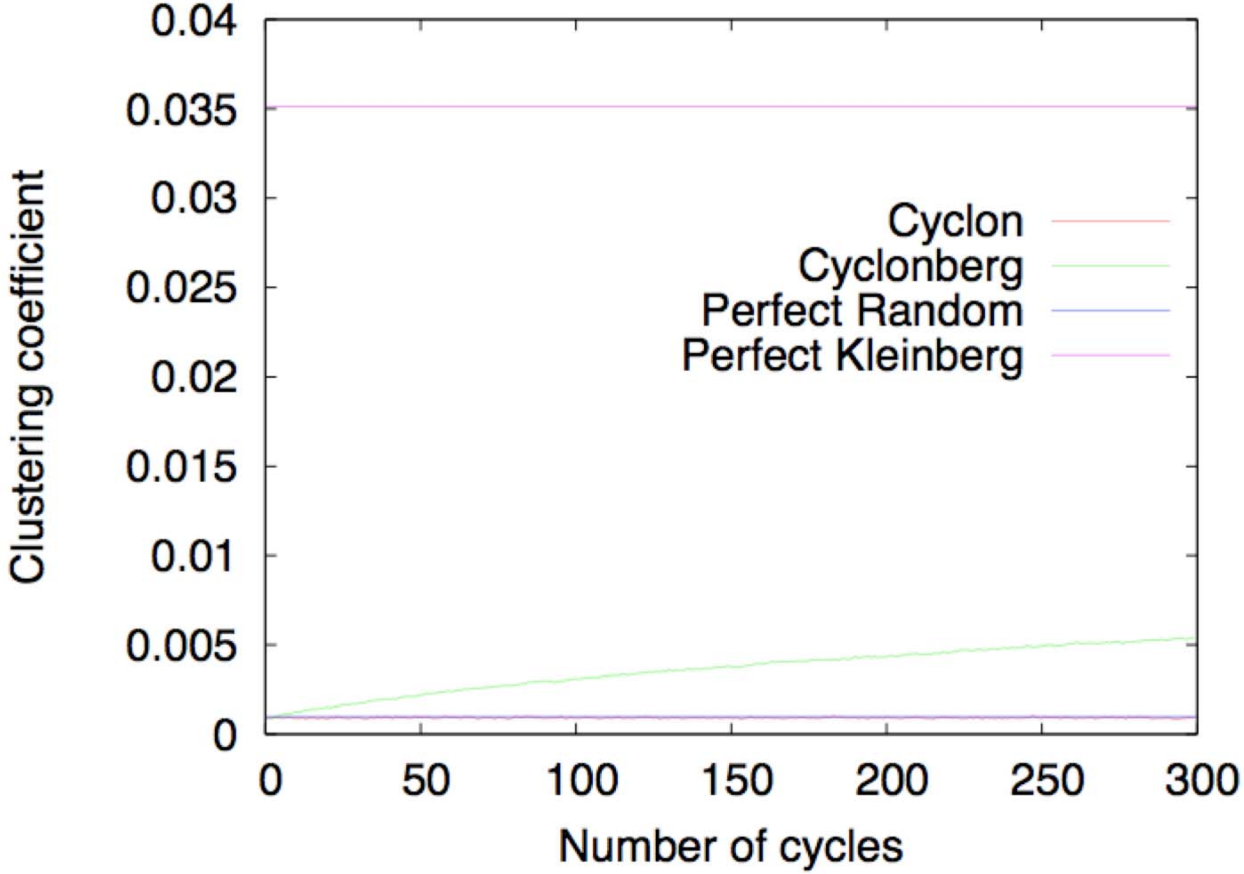


Path length



Average path length
4.8
4.75
4.7
4.65
4.6
4.55
4.5
0
50
100
150
200
250
300
Number of cycles
Cyclon
Cyclonberg
Perfect Random
Perfect Kleinberg

Clustering coefficient



Outcomes

- Possible to tune the peer sampling to achieve a routing similar to the one obtained with a Kleinberg's shortcut selection
  - Driven by the shuffle length
- Resulting graph properties
  - Degree distribution and average path length similar to a random peer sampling
  - Clustering coefficient: slightly higher
  - Harmless to most distributed applications
- Improves the clustering algorithm

Structuring the network: ordering nodes

Gossip-based distributed slicing

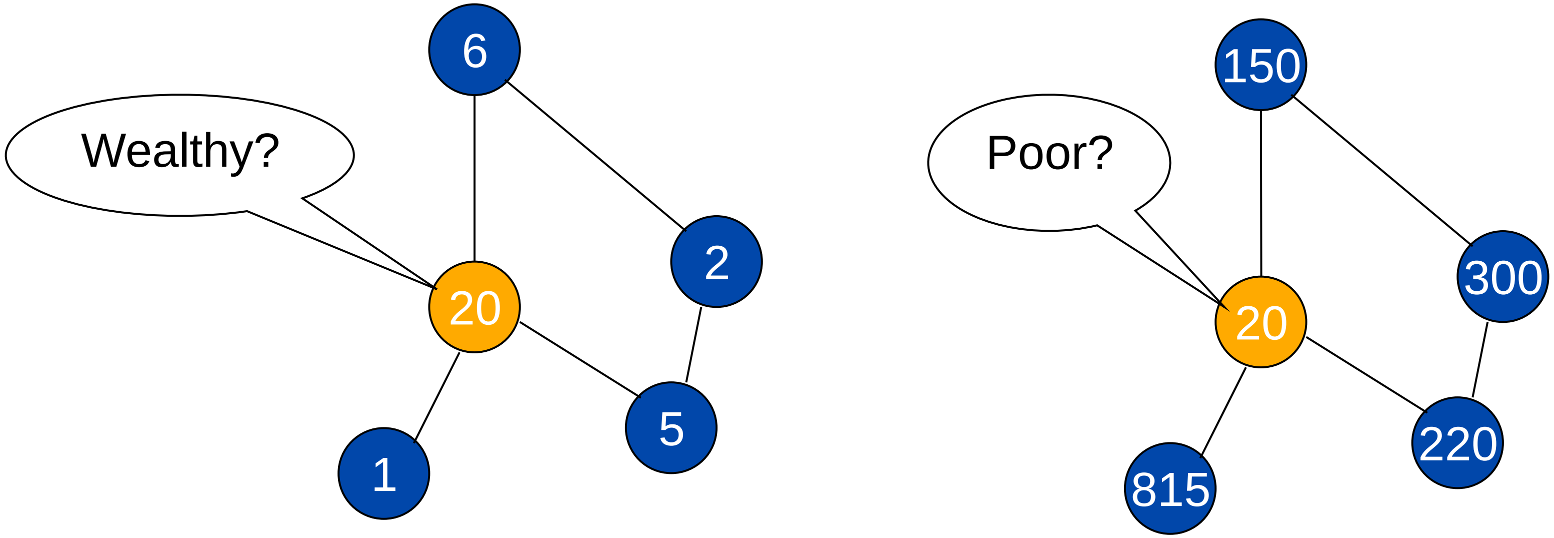
[JK,P2P 2006] [FGJKR,ICDCS 2007]

Why slicing a P2P network?

- Slices: sets of size proportional to the size of the network
- Heterogeneous environment: Identify sets of specific nodes
  - Live streaming applications (upload)
  - Load balancers in datacenters (CPU,availability)
  - File sharing systems (number of files, storage)

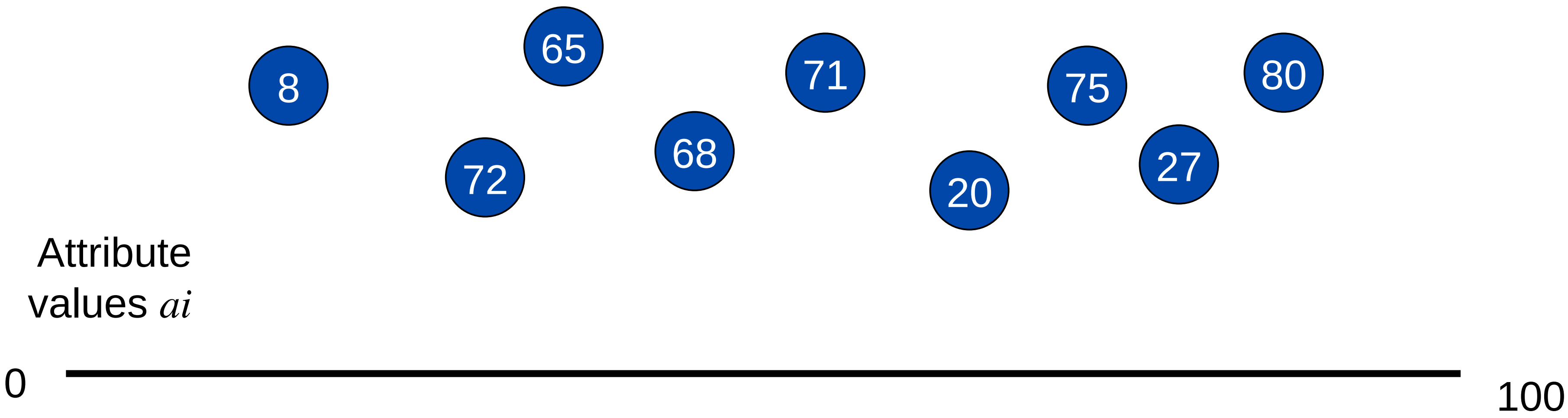
Basic structure: slice

Why slicing is not trivial?

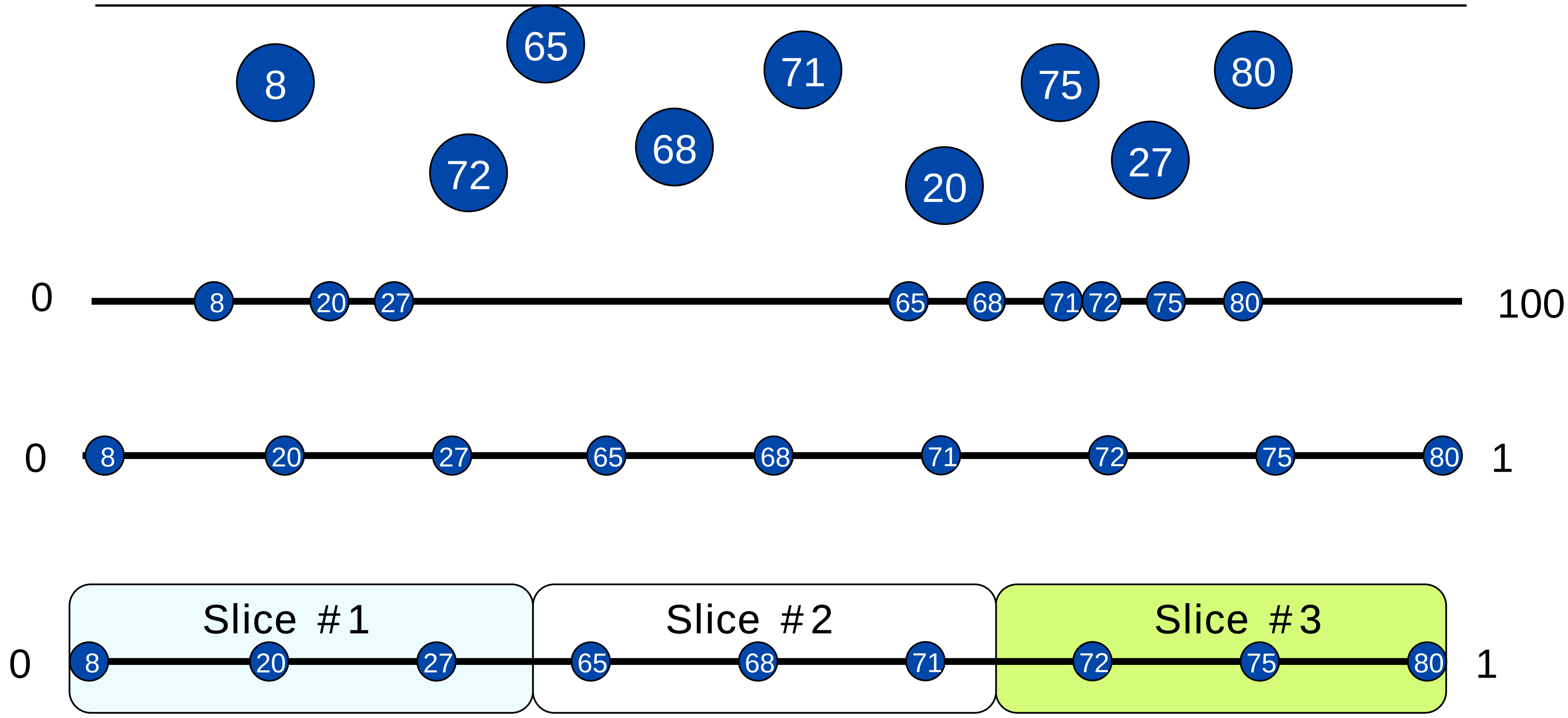


- Presence of churn
- Dynamic heterogeneity
- No global information

Classifying nodes



Slicing the network



Objective

Create and maintain equally balanced slices of the network in a fully decentralized manner

Upon termination: each node knows the slice it belongs to

Gossip-based approach

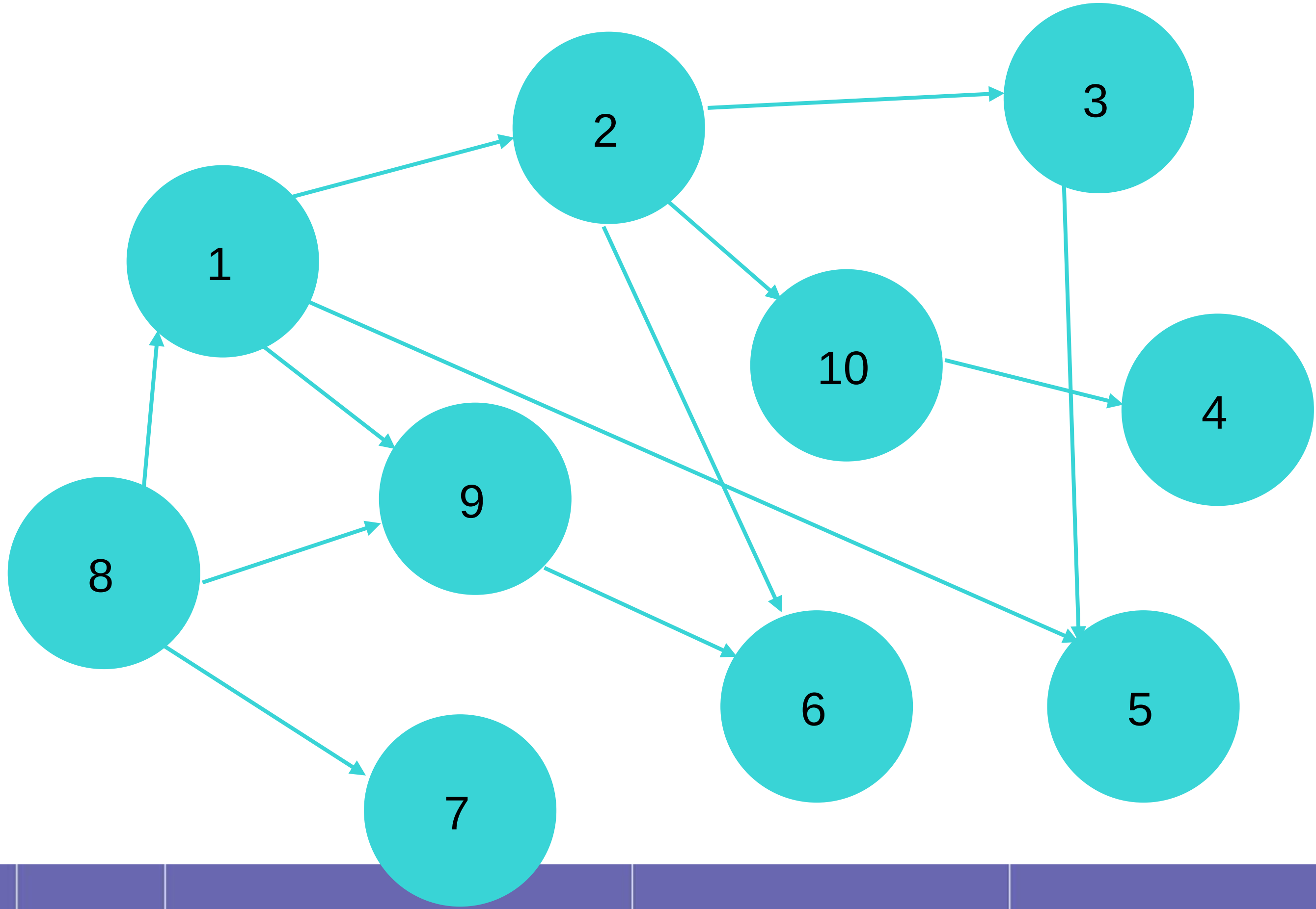
Use a gossip-based approach to estimate to which partition a node belongs

- Scalable
- Robust
- Based on local knowledge
- Fast convergence

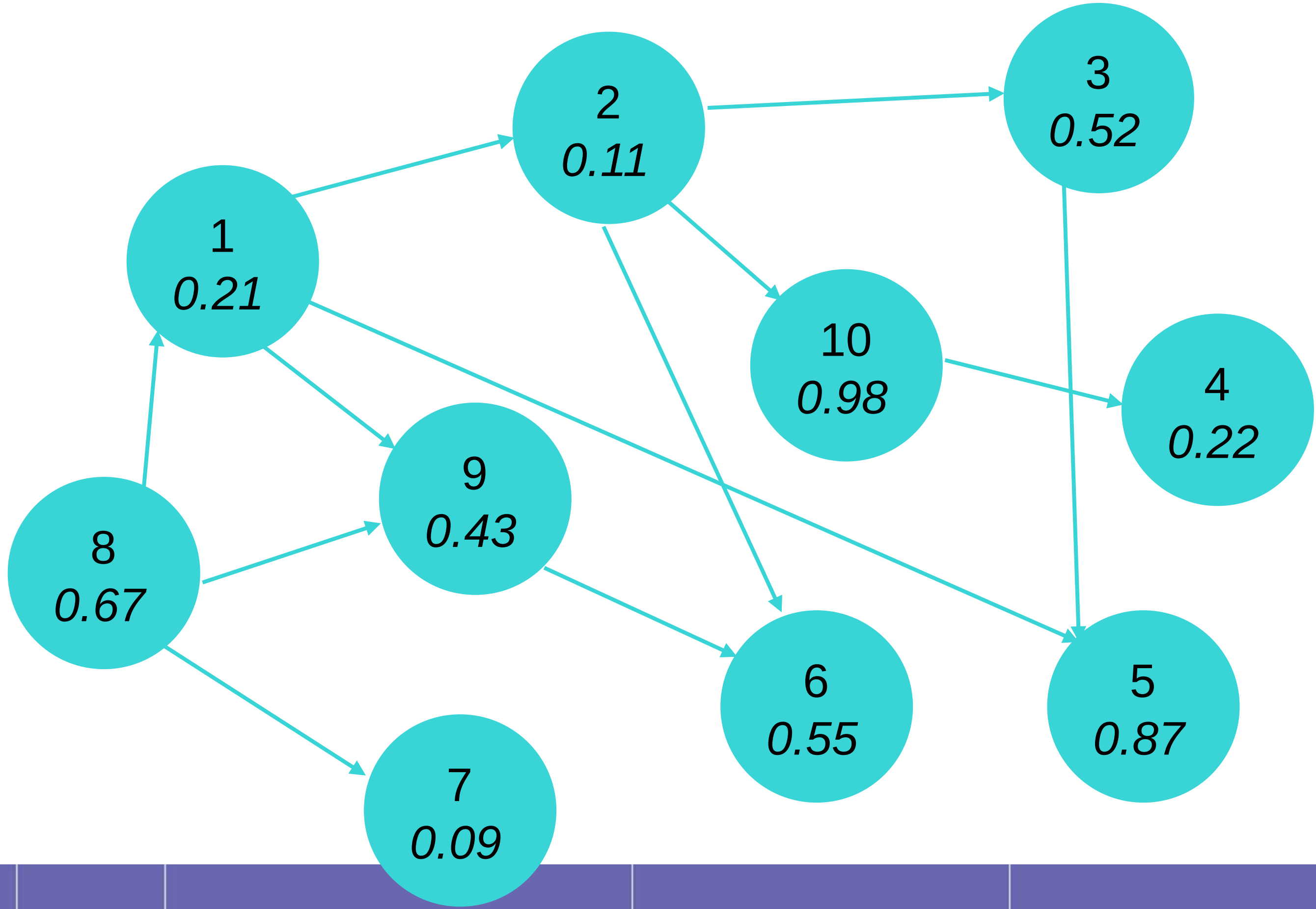
System model

- Dynamic system of peers uniquely identified
- Each node belongs to one slice and has
  - an attribute: capacity in the metric of interest
  - a random number
  - a view of c entries (peer sampling)
  - a time stamp

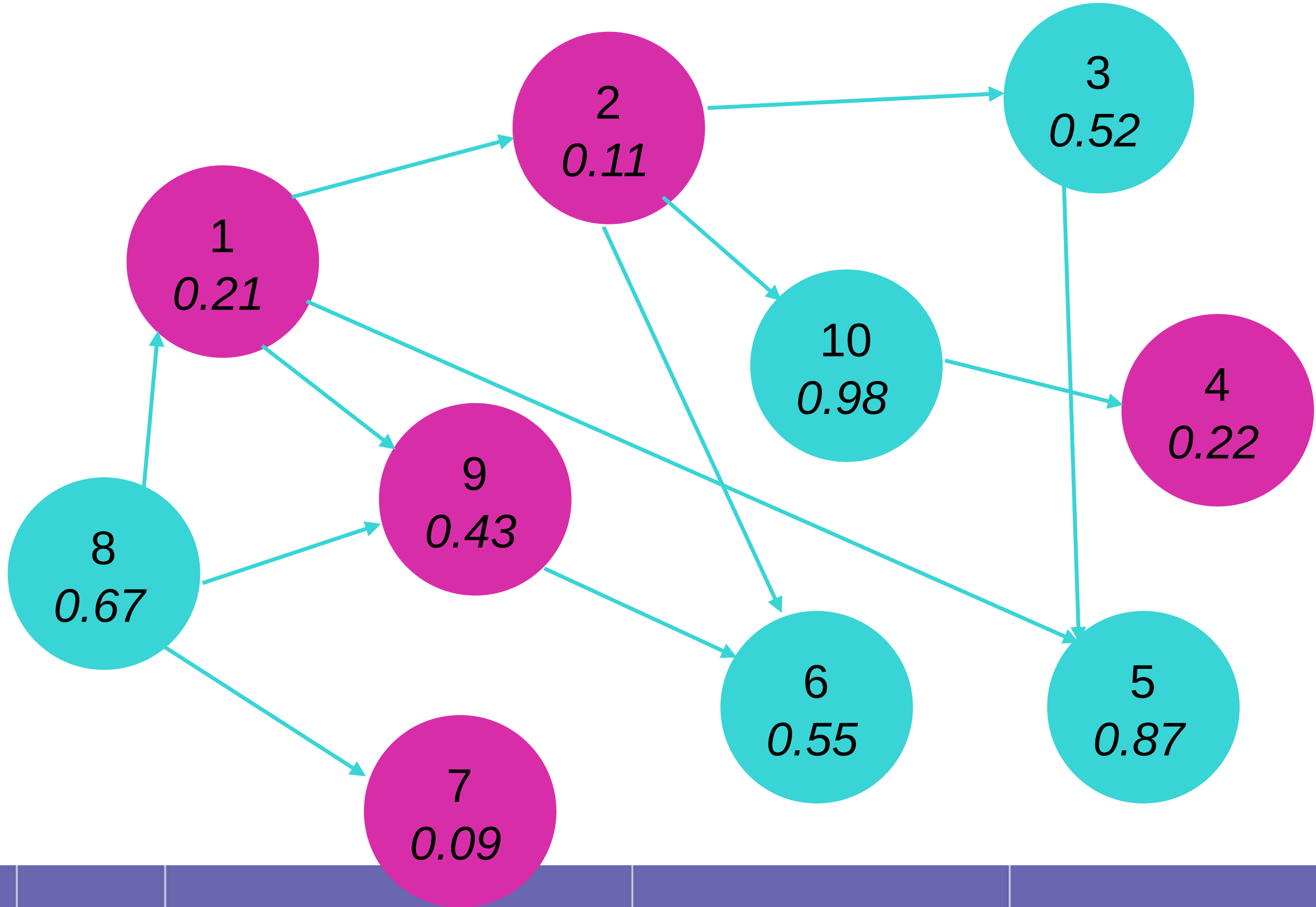
Random slices



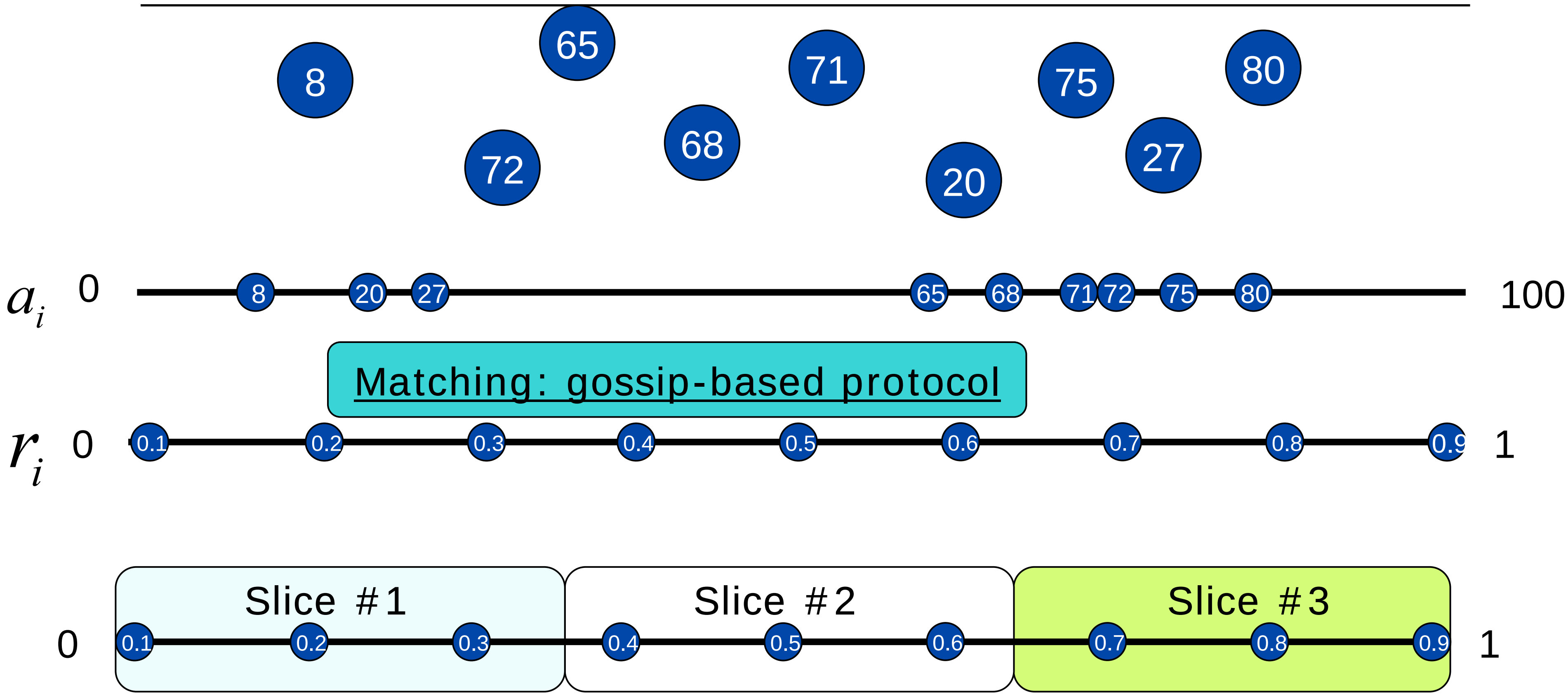
Random slices



Random slices



Gossip-based approach



Ordered slicing algorithm: basic operation

			Node a					Node b	
a_i	72	71	20	65	68	75	27	8	80
r_i	0.6	0.5	0.3	0.7	0.1	0.4	0.2	0.9	0.8

a_i	72	71	20	65	68	75	27	8	80
r_i	0.6	0.5	0.9	0.7	0.1	0.4	0.2	0.3	0.8

a_i	8	20	27	65	68	71	72	75	80
r_i	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9

The ordered slicing algorithm

On each node *q*

- Pick a node *p* at random in its view

- Initiate a gossip with p
  - Send its own a_q, r_q
  - Receive the *freshest c* entries from *p*
- Select *i* such that $(a_i - a_q)(r_i - r_q) < 0$
  - Swap random values

Ordered slicing algorithm: maintenance

- New nodes discovery: peer sampling (age-biased)

- Random values: uniform spread

- Once the order stabilizes: each node knows which slice it belongs to

A peer with a number <0.5 knows in the first 50% of the nodes according to the metric

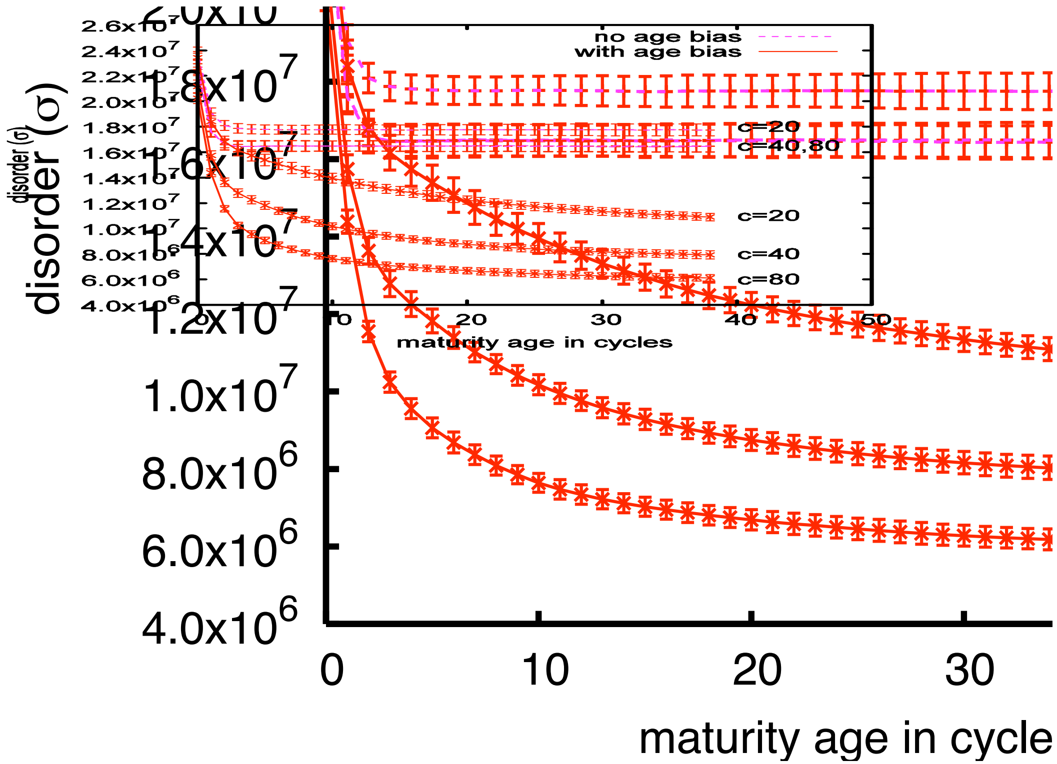
Analogy with average

- Weight conserving property

$$1/N\sum_{j=1}^{N} j-i_j(t)=1/N\sum_{j=1}^{N} j-1/N\sum_{j=1}^{N} i_j(t)=0$$

- The swapping does not influence this value (=0) but always reduces the disorder value

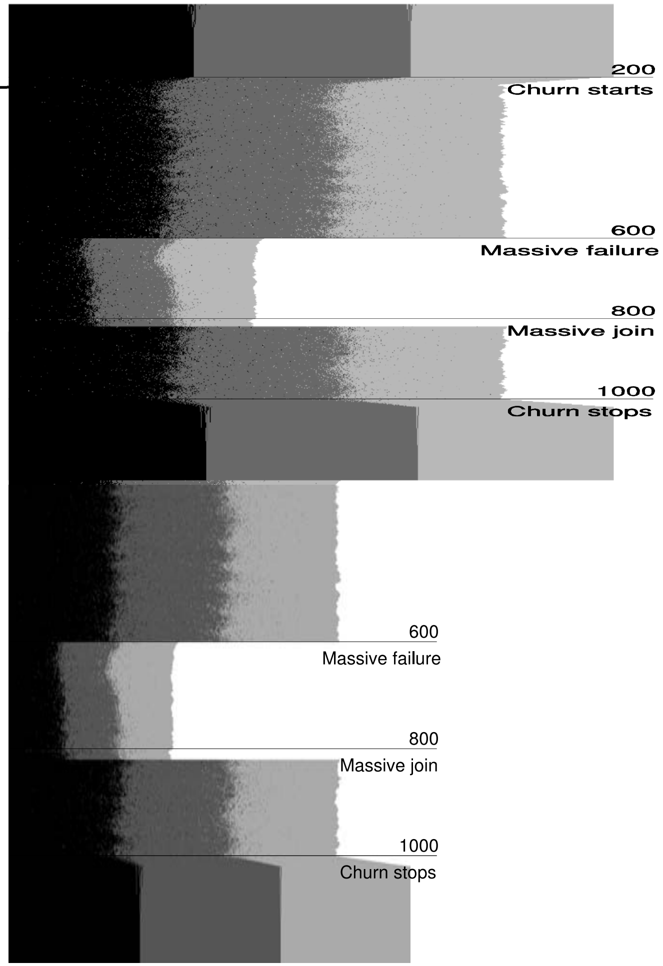
Age-based technique



Young nodes disordered
Old nodes protected

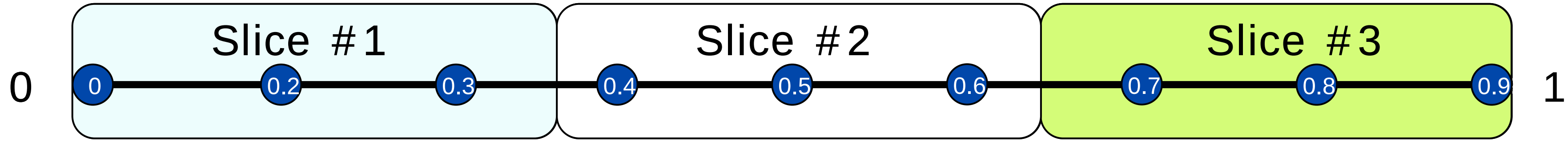
Main results

- Exponential decrease of the disorder
- Quick stabilization
- Relatively well-defined slices
- Stabilizes as soon as churn stops



Ordered slicing: optimizations & issues

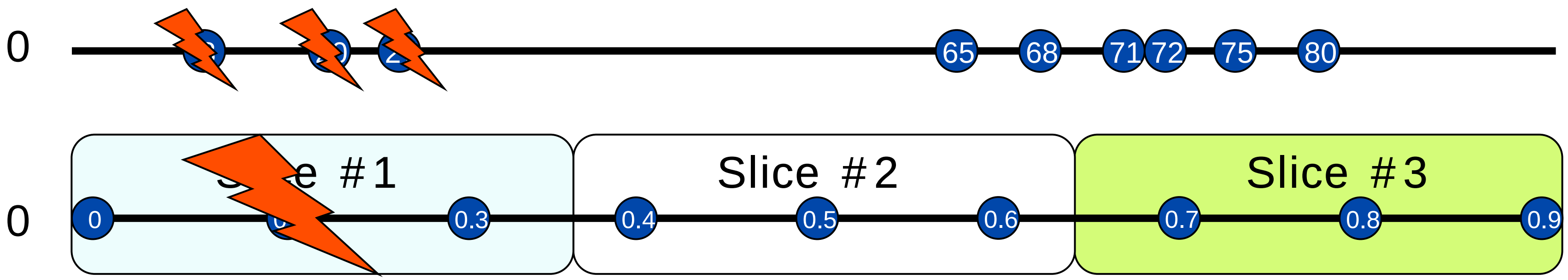
- Further optimization: Local measure of the disorder [Fernandez & al, ICDCS 2007]
- <u>Issues</u>
  - Uniformity requirement



Ordered slicing: issues

- Uniformity requirement
- Failures are correlated to the attribute values

Provides an ordering not an accurate ranking



Ordered slicing

- Issue when failures are correlated to the attribute values
- Fix the uniformity requirement
- [Fernandez & al, ICDCS 2007]
  - Infer slice from a sample of attributes
  - Gossip-based propagation

Why would we want to route efficiently in something else than a DHT? Why is gossip relevant here?

- Range queries in a P2P overlay: VoroNet-RayNet [Beaumont & al, IPDPS 2006, OPODIS 2007]
- Content-based publish-subscribe systems: Sub-2-Sub [Voulgaris&al, IPTPS 2006]

VoroNet
A scalable object network
based on Voronoï tessellations

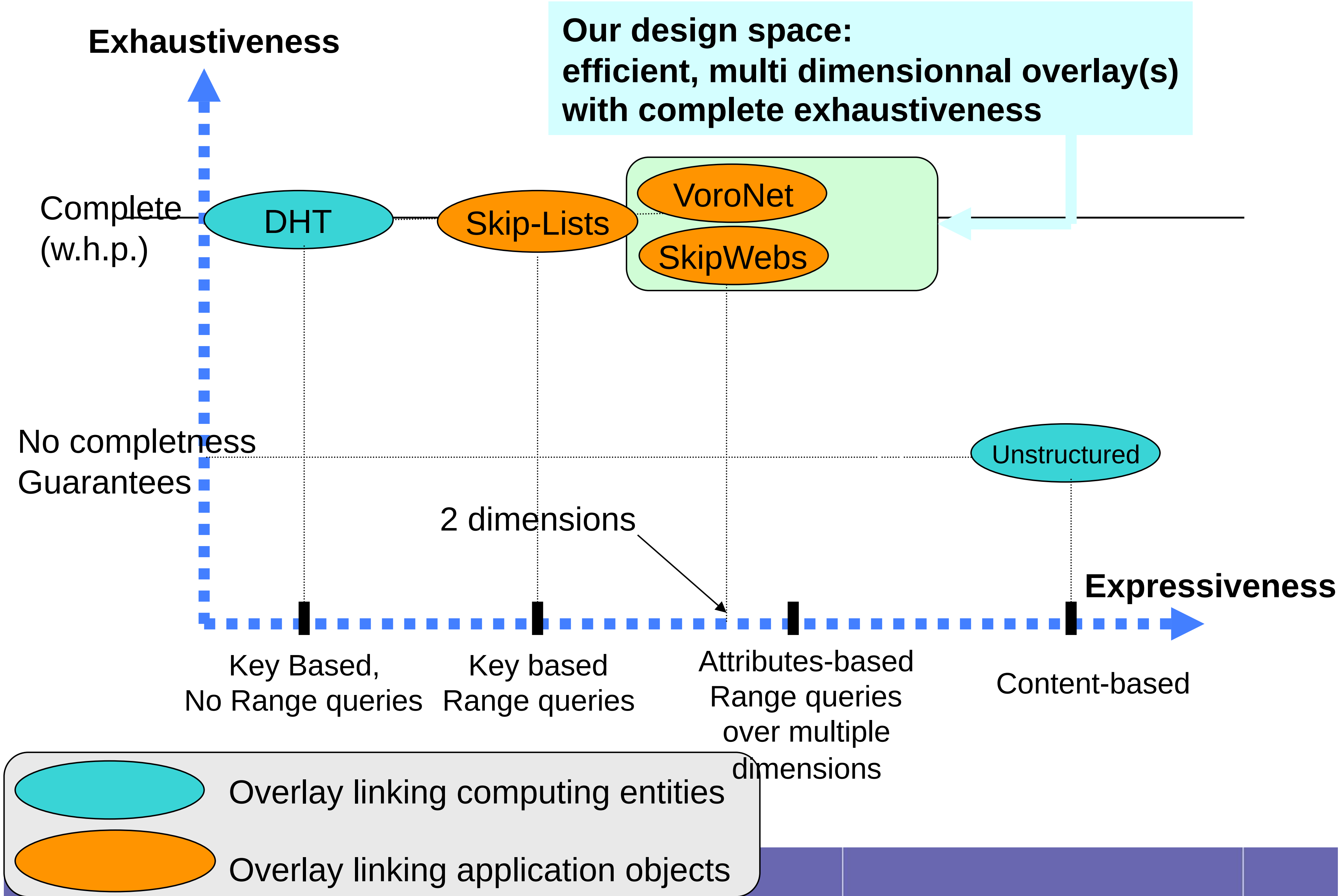
Design rationale

- Efficient data location service

  Efficiency = expressiveness + completeness

- Expressiveness versus completeness
  - Unstructured overlay/Structured overlays (DHT)

- Overlay structure should reflect the application one
  - Linking objects in an efficient routing overlay
  - Use of Voronoï tesselation of the object space
  - Efficient routing: Kleinberg small world model

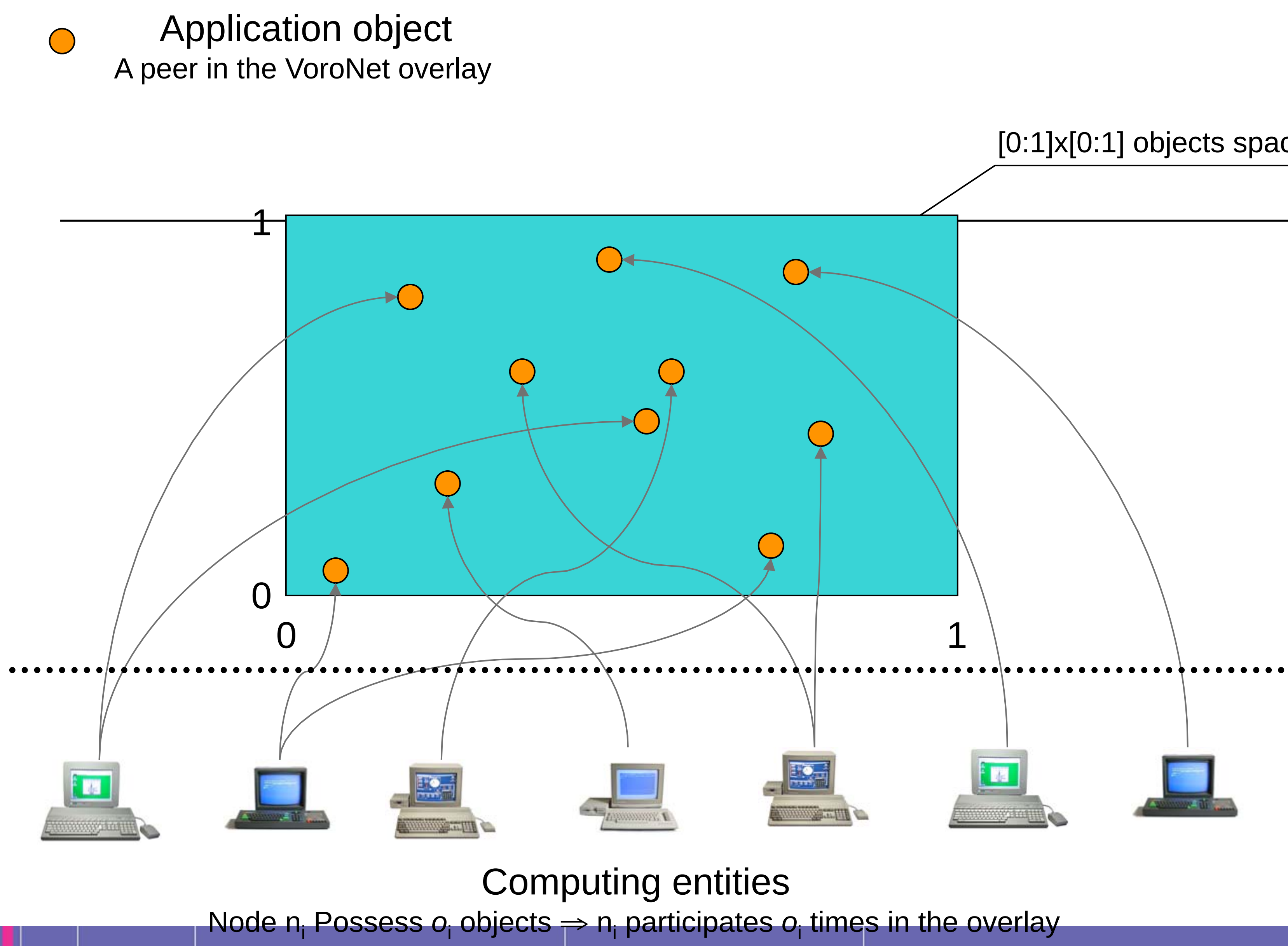

Our design space:
efficient, multi dimensionnal overlay(s)
with complete exhaustiveness
Exhaustiveness
Complete
(w.h.p.)
DHT
Skip-Lists
VoroNet
SkipWebs
No completness
Guarantees
Unstructured
2 dimensions
Expressiveness
Key Based,
No Range queries
Key based
Range queries
Attributes-based
Range queries
over multiple
dimensions
Content-based
Overlay linking computing entities
Overlay linking application objects

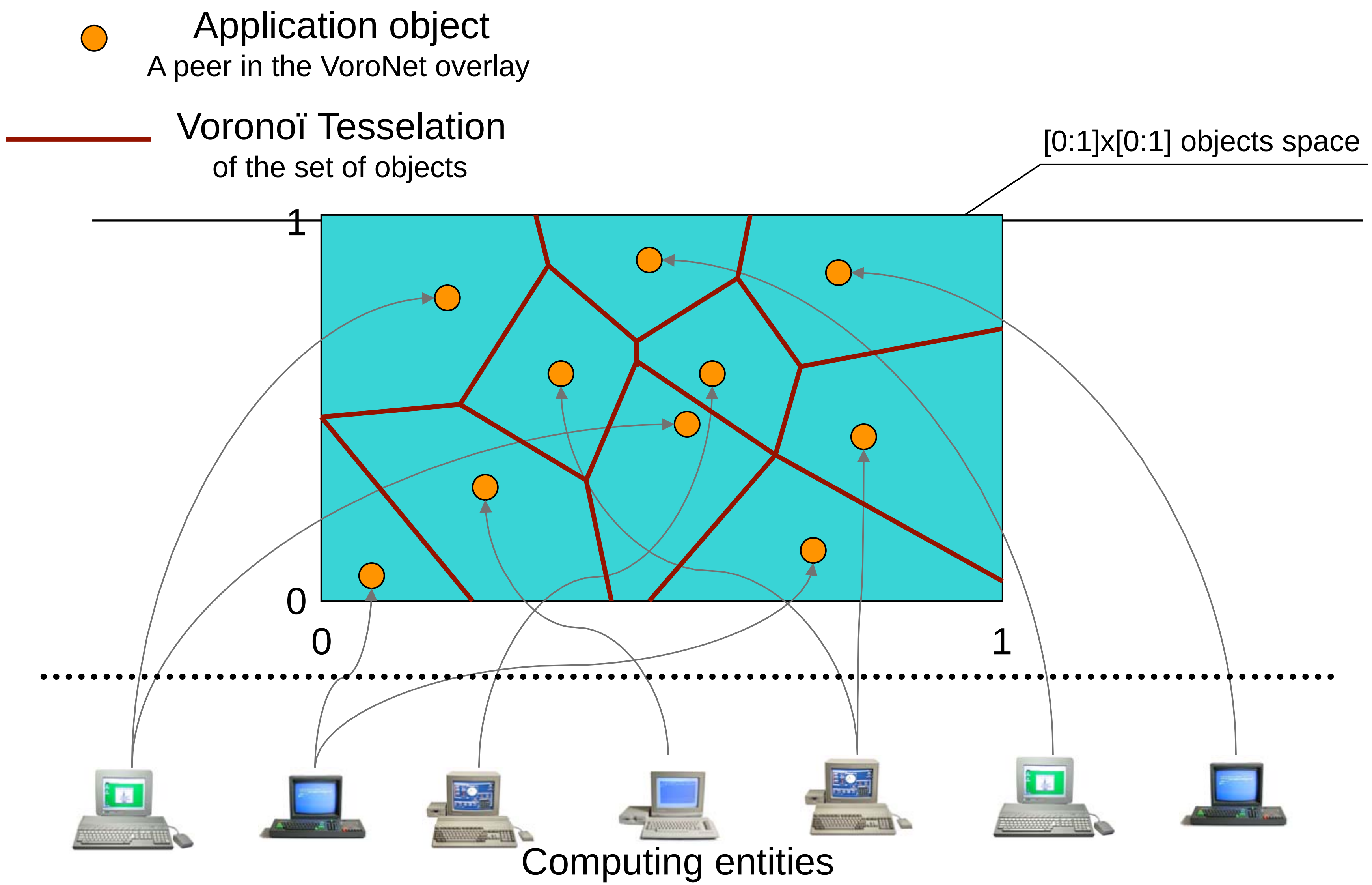
Model

- An object is described by a set of attributes
  - Objects with “near” attributes are neighbours in the overlay
  - Multidimensional naming space
  - ***For ease of explanation***
    - we limit to the case where dimension is 2

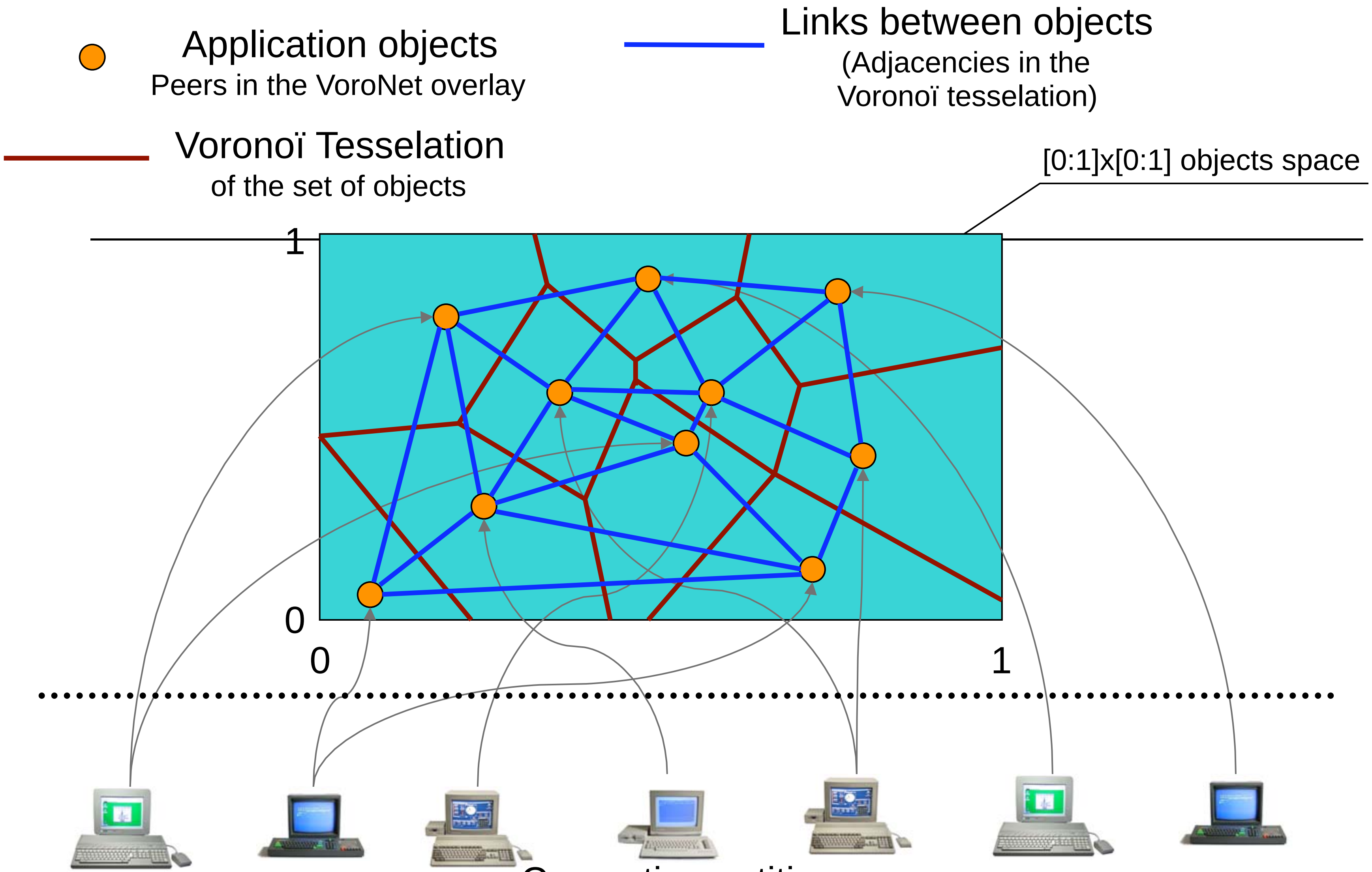
- Native and efficient support for efficient query mechanisms
  - Scalable, polylogarithmic routing
  - No hash mechanisms ⇒ Ordering preserved
  - Generalizes Kleinberg Small-World model

- State per object is O(1)
  - Independently of the object set size and distribution
  - The basic overlay is based on the Voronoï tessellation of the objects set in the Euclidean naming space


Application object
A peer in the VoroNet overlay
[0:1]x[0:1] objects space
1
0
0
1
Computing entities
Node n_i Possess o_i objects ⇒ n_i participates o_i times in the overlay



Node n_i Possess *o* objects $\Rightarrow$ n_i participates *o* times in the overlay

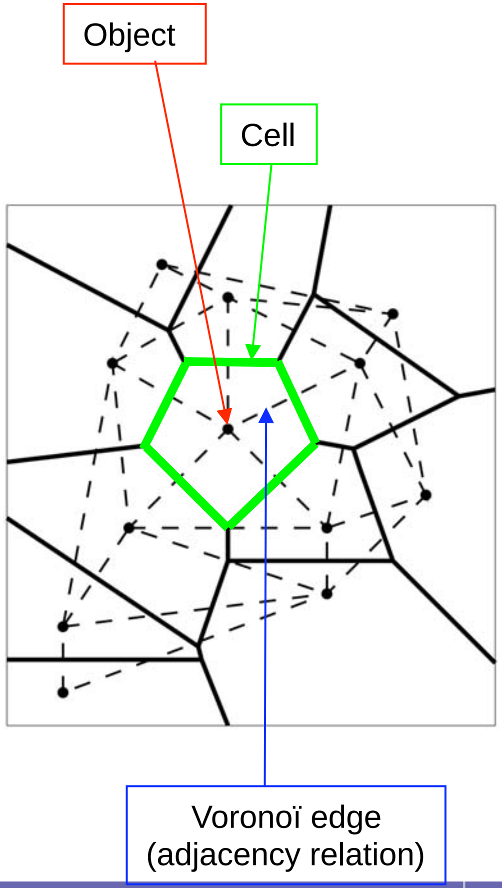


Computing entities

Node n_i Possess *o* objects ⇒ n_i participates *o* times in the overlay

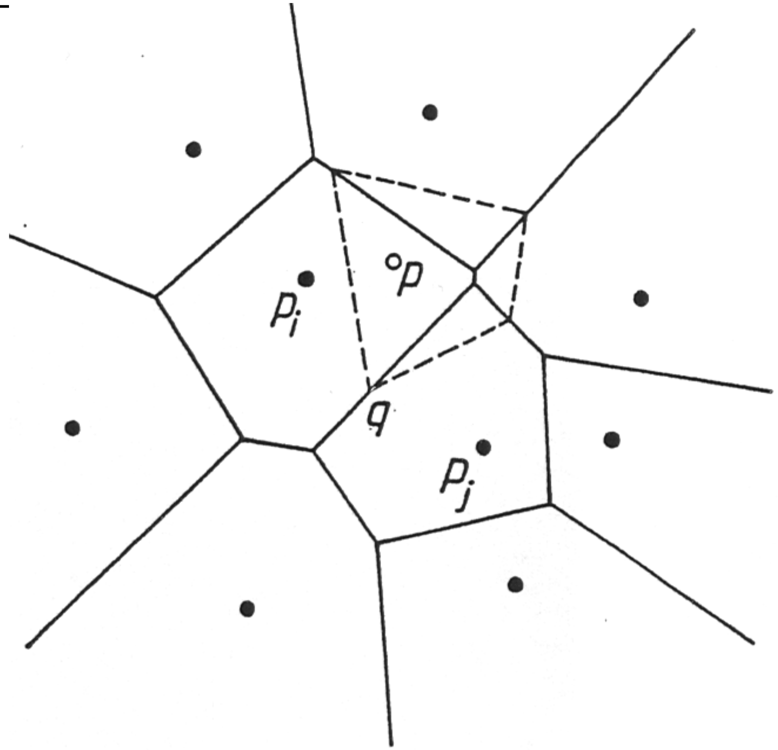
Voronoï tesselation

- Definition
  - For each point p among a set
  - p's cell contains all points nearest to p than to any other point
- The dual of the Voronoï diagram is the Delaunay triangulation
  - Mean(#neighbors) ≤ 6
  - Navigability: greedy Euclidean routing always succeeds (in linear number of steps)
- **Overlay primary links** between objects are adjacency links of objects (virtual) cells



Object insertion

- Each object knows
  - Neighbors coordinates and zones
- A joining peer p routes a message to its coordinate
  - Peer p_i is responsible for p insertion
  - p_i computes p's new zone and modifications to its neighbors's (e.g. p_j) zones
  - p_i disseminates changes to its neighbors and notify p of its new neighborhood



Efficient routing

- Greedy routing
  - Each routing step gets closer to the destination A
  - Delaunay triangulation properties ensure that this succeeds deterministically
  - But…..may be O(N) steps
- Small world routing
  - Additional shortcuts
  - Extension of the Kleinberg's model
  - Polylogarithmic routing in N : $O(\log^x(N))$

Extending the Kleinberg model

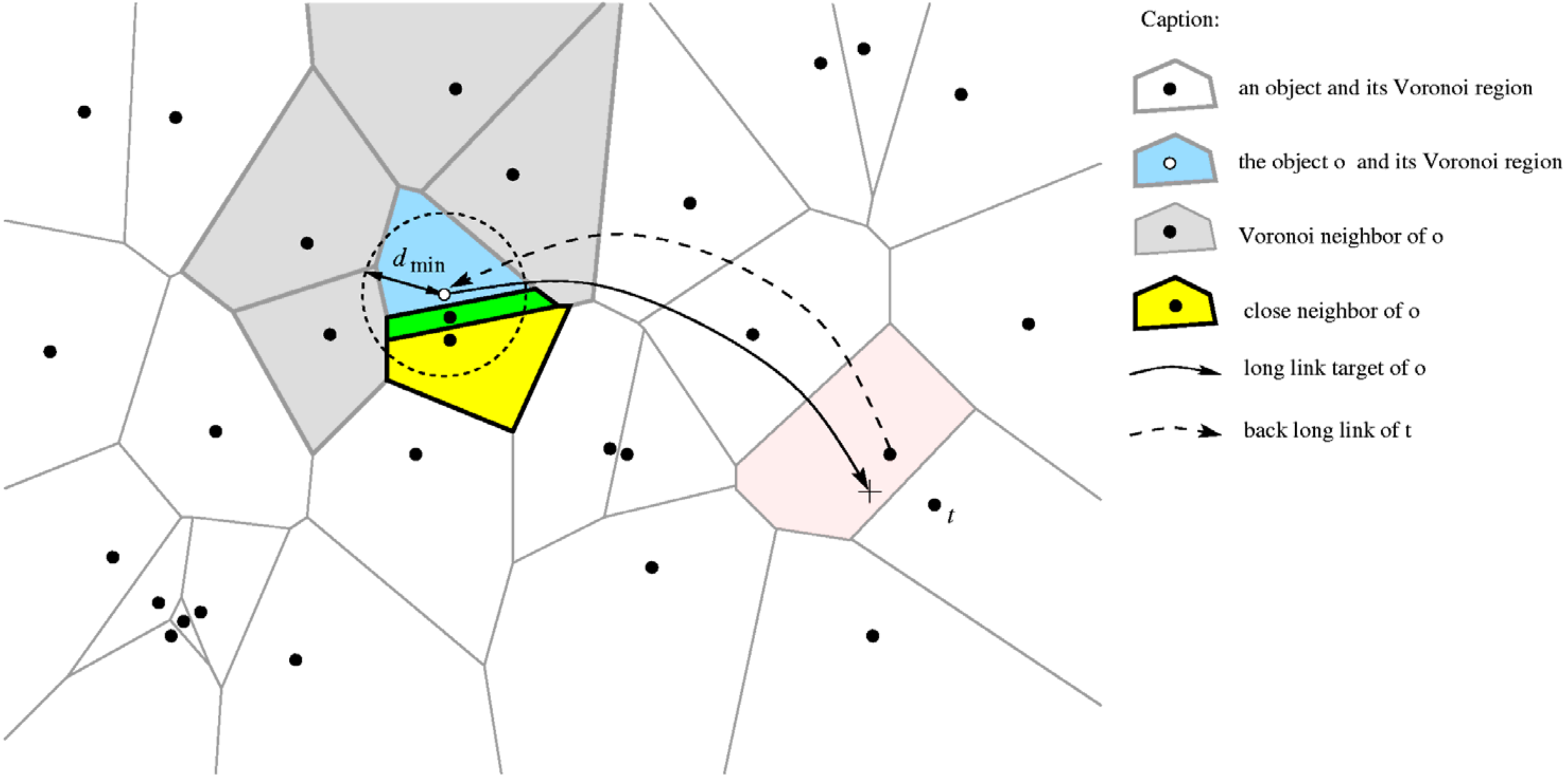
- Each object chooses a shortcut destination point according to a harmonic distribution, and uniform direction

- The topology is not a grid !
  - The destination point is not necessarily an object ...
  - But the destination stands in an object cell

- The object chosen as shortcut neighbour is always the object which has the shortcut destination in its zone

- Greedy routing ensures paths of polylogarithmic size

Management of long links



How many neighbours?

- Close neighbors: Voronoï neighbours (Mean ≤ 6)

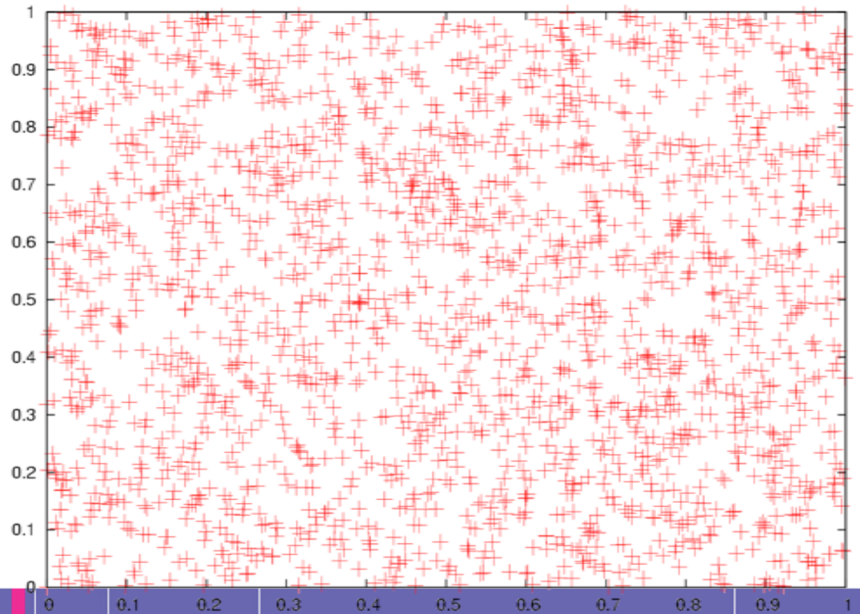
- Shortcuts
  - Simulations have shown that around 6 shortcuts is a good tradeoff between maintenance cost and performance

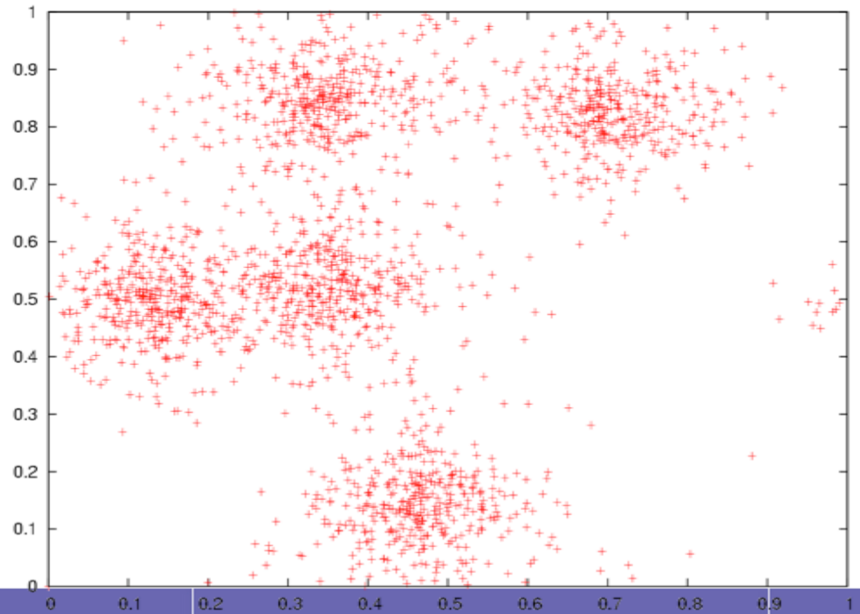
- Back long link neighbours
  - Dependent on the distribution of objects,
  - Balanced even with sparse distributions due to long link properties (random versus harmonic)

- Overall neighbour set size is O(1)
  - Independent of the number of objects
  - Independent of objects distribution

Experimental settings

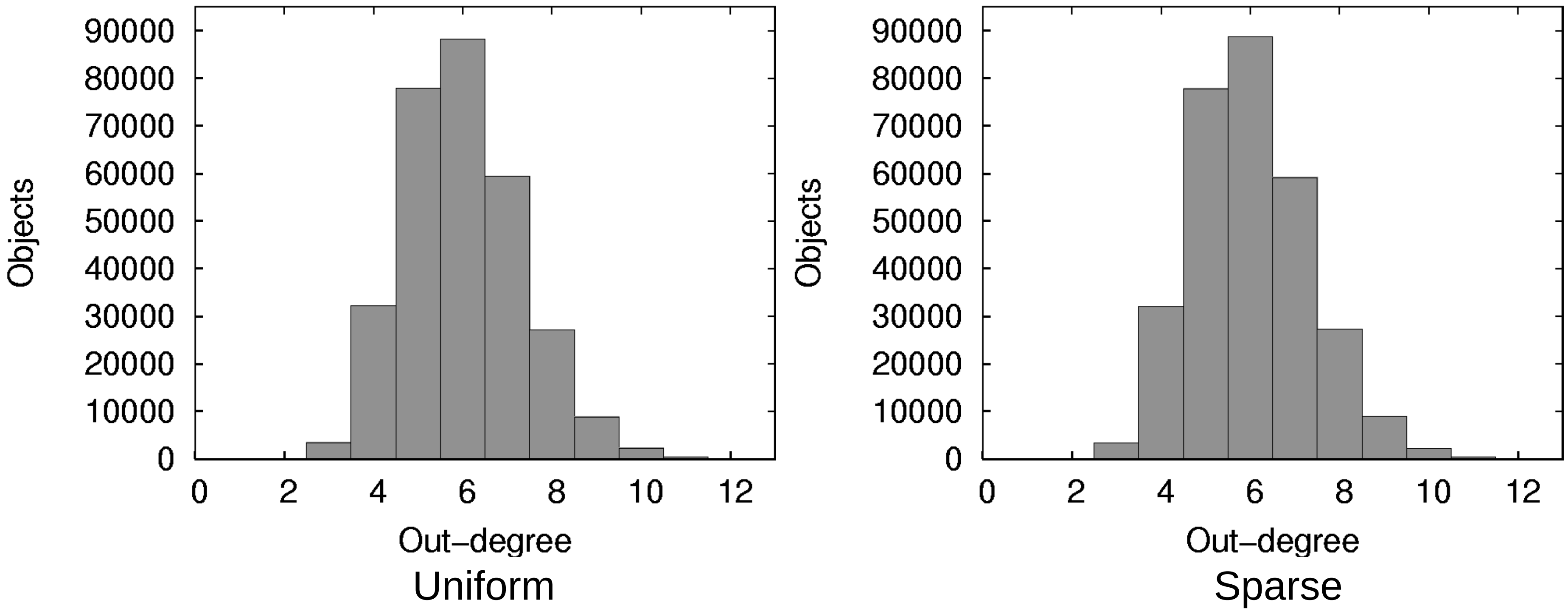
- 300.000 objects (no object leaving)
- 2 object distributions in [0:1]x[0:1]
  - Uniform
  - Sparse: 5 equally popular regions. Popularity of objects around a region follows a power law with $\alpha = 5$





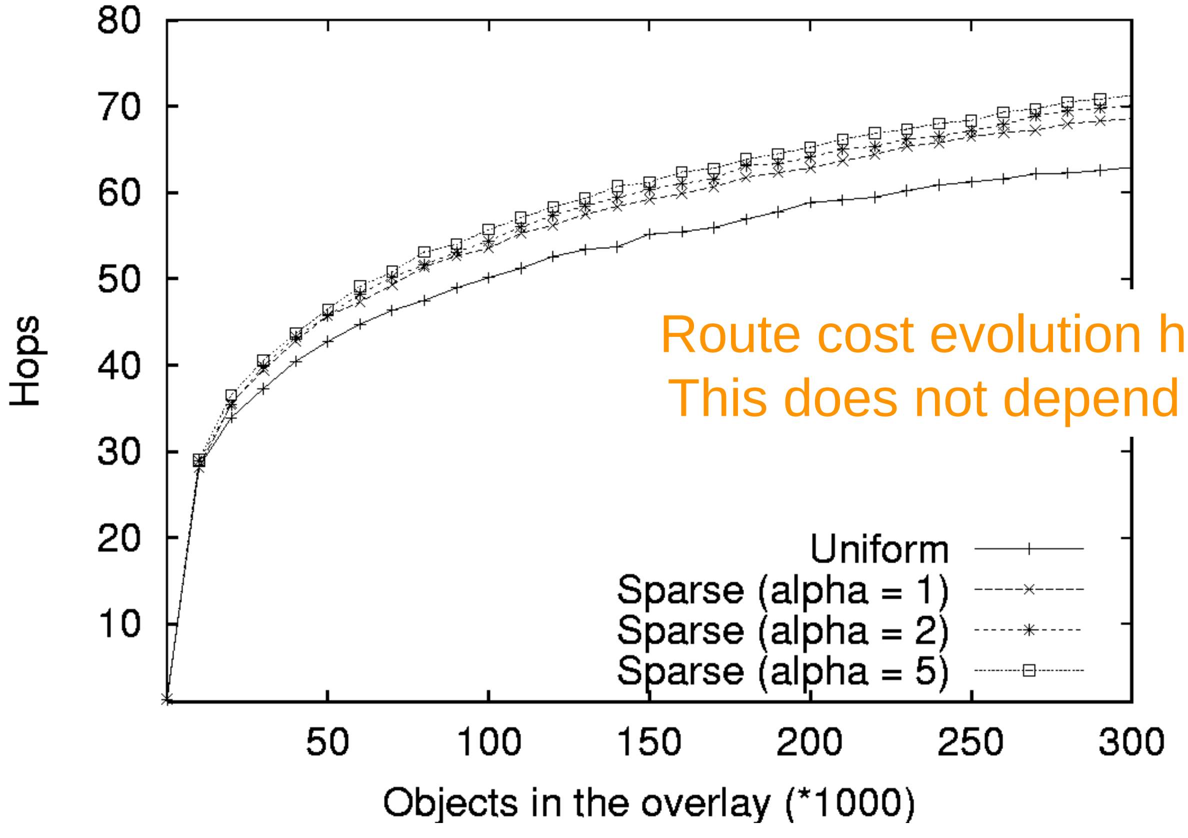
Simulation: object degree

number of Voronoï neighbours



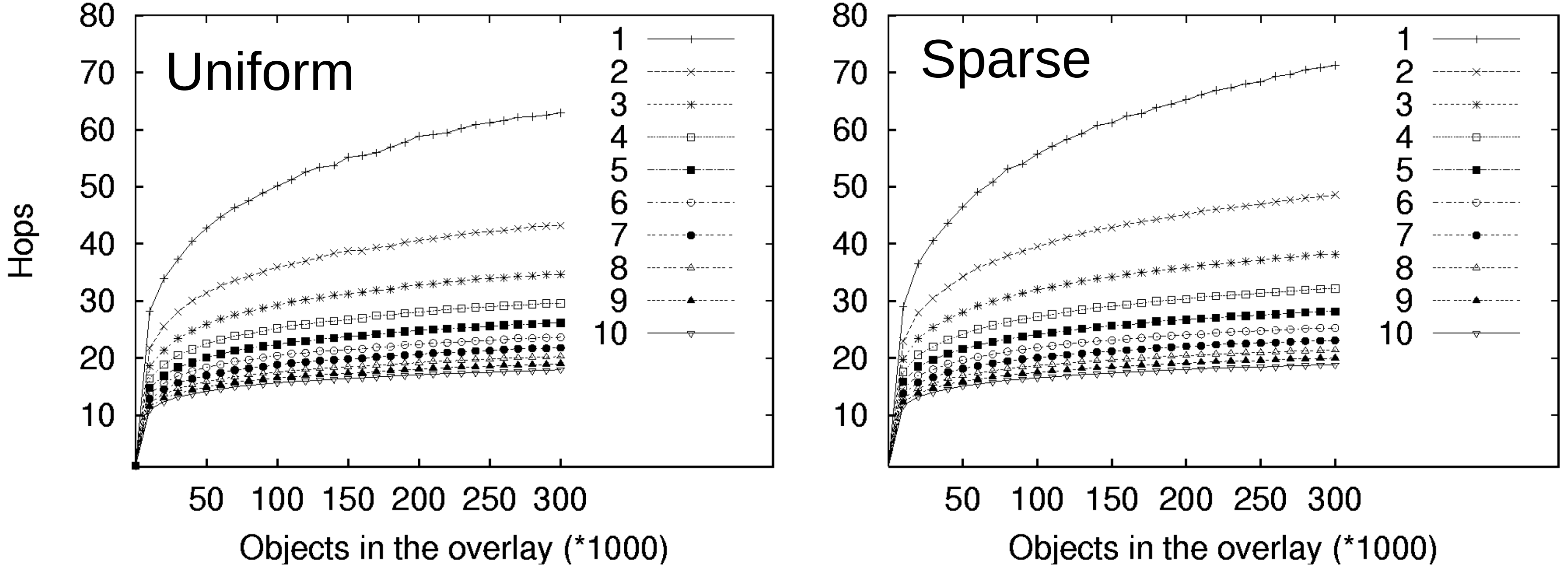
Object out-degree does not depend of the objects distribution in space

Polylogarithmic routes (1)



Route cost evolution has a logarithmic shape.
This does not depend on objects distribution.

Using several long links improves routing performance



- Linear improvement: Using *k* shortcuts provides a routing that is almost *k* times more efficient
  - At each step, the probability of using a long link that divide the path by log(N) is k/log(N)
  - A reasonable amount of long links is ~6 for a 300.000 objects overlay

Voronoï cell computations are an overkill

RayNet: gossip-based approximation of complex structures

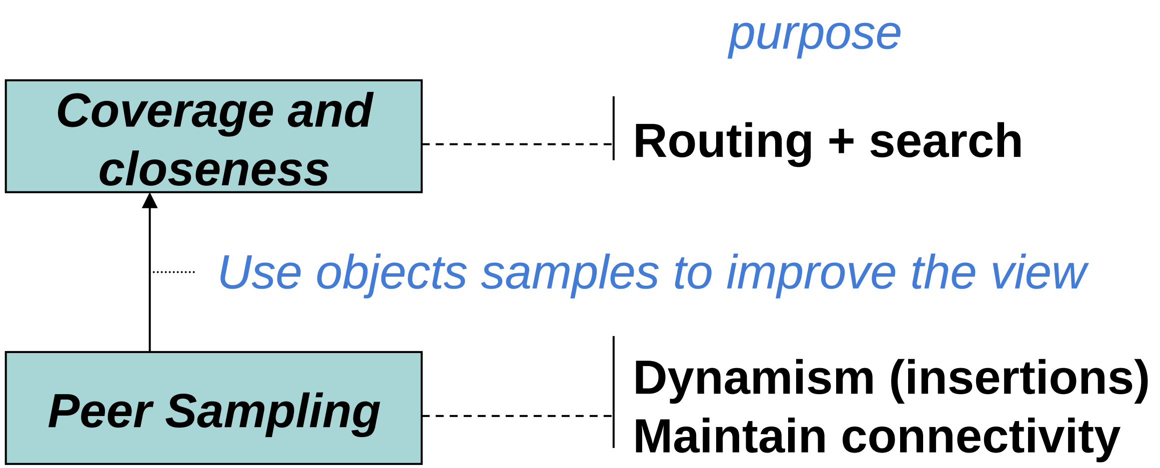
INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Voronoï diagrams, RayNet rationale

- VoroNet
  - Complex structure to compute, to maintain in face of churn, potential unlimited number of neighbours
- **What really matters?**
  - Neighbours: to ensure correct routing
- **Using an approximation of the structure is enough** to compute such neighborhoods
- Gossip-based protocols for Voronoï neighbours and shortcuts

Gossip-based construction of RayNet

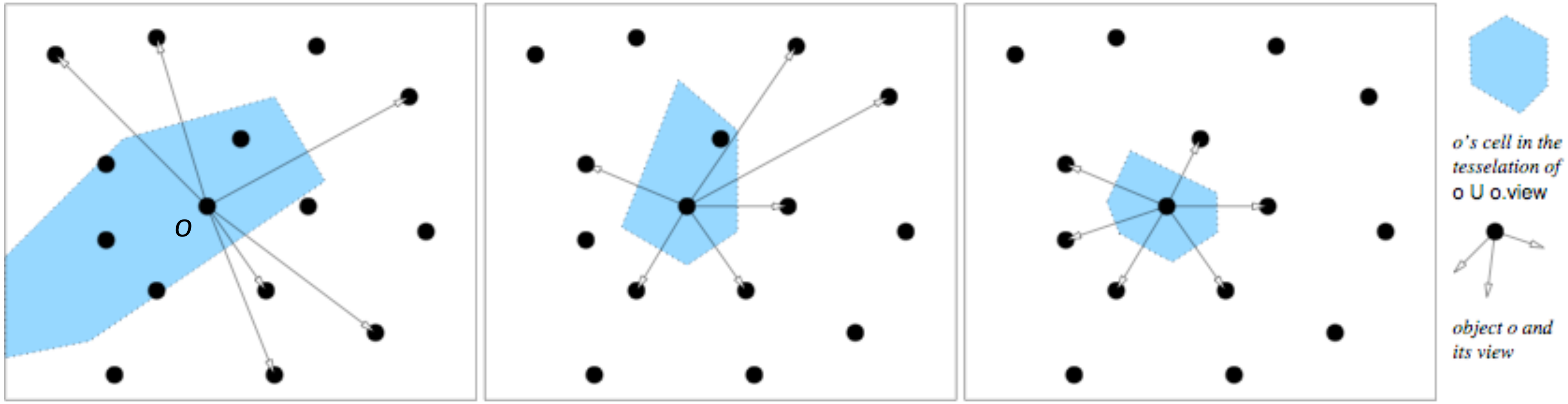
- Local links: Coverage and closeness
  - Gossip-based construction of approximate Voronoï links
  - Close objects (in the semantic space) in all directions
- Shortcuts: Kleinberg peer sampling



Challenge:

Evolution of local views towards a global routing structure

Coverage and closeness



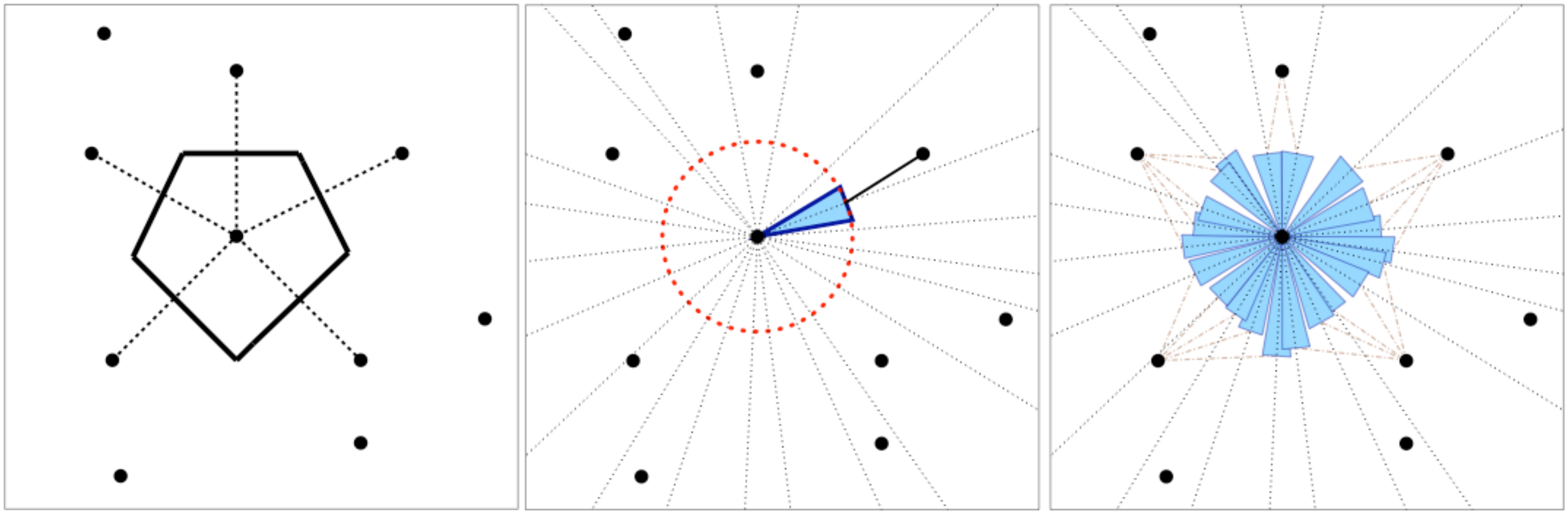
An object o's view == Voronoï neighbours

Idea:

- Exchange views & converge towards an approximation of Voronoï neighbours
- No need to compute the Voronoï cells: use the **volume** as an indication of convergence (the smaller, the better)

Monte Carlo cell size estimation

- **Idea**: sample the boundaries of the zone using "rays"
- Gossip-based protocol: evaluate the view as a whole (configuration)



View update operation: naïve approach

- View size is c=3d+1 peers
- Exchange entire views : o.view + $o_{partner}$.view

- For each set S of objects of size c, in o.view + $o_{partner}$.view
  - Estimate the volume of o's cell in the diagram of S
  - Keep the set with minimal volume as the new view

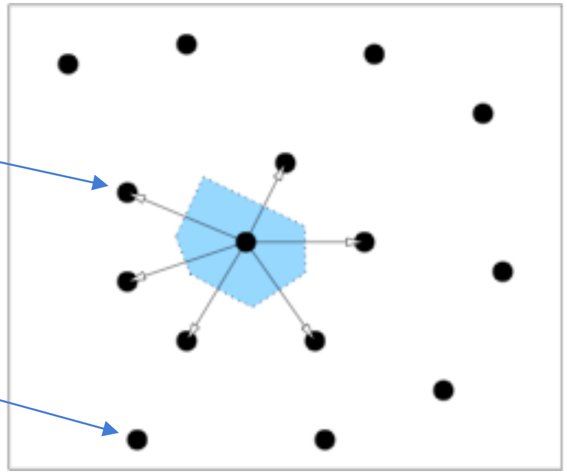
- Effective, but there are O(c!) configurations to examine…

View update operation: efficient approach

- Determine the potential contribution of each object to the coverage and closeness (ie to the volume of o's cell)
- For each object o' in o.view + $o_{partner}$.view
  - Compute the volume of o's cell in o.view + $o_{partner}$.view <u>without</u> o'

Ignoring this object results in a bigger zone: High contribution

Ignoring this object does not impact the size of the zone: No contribution

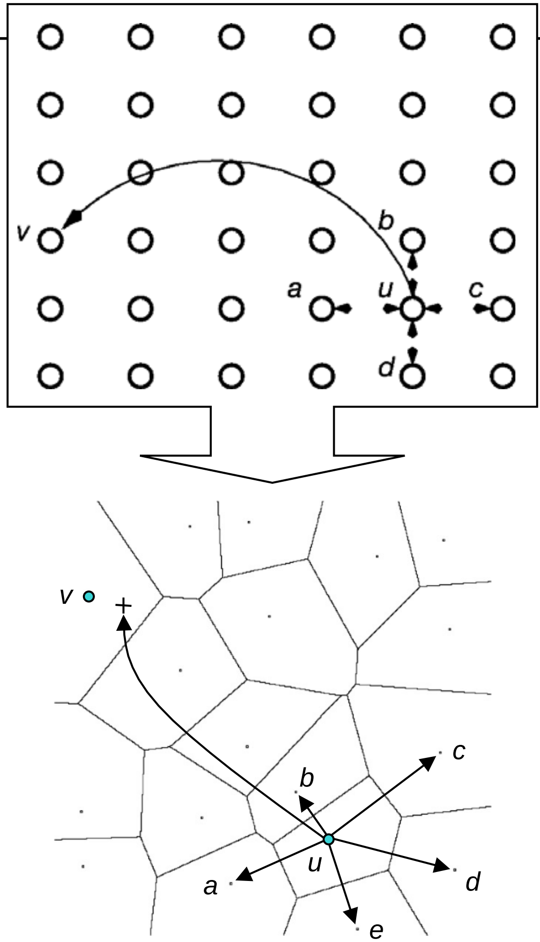


- Keep the c objects with the greatest contribution

Efficient routing

- Routing in the approximate Voronoï diagram requires O(N) hops
- Small-Worlds models:
  - Small paths + navigability
- Using biased peer sampling

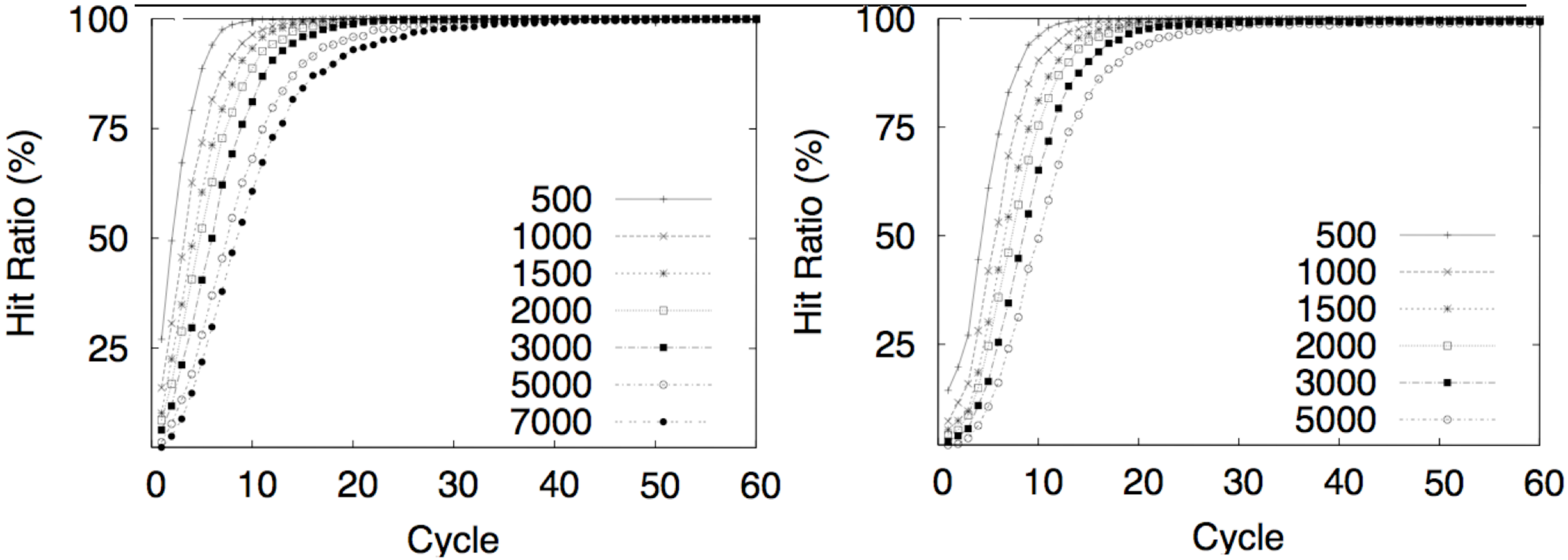
- $O(\log^d N)$ routing with 1 shortcut



Simulations

- Settings
  - 1.000 to 7.000 objects
  - Emergence from a chaotic state
    - No RayNet links
    - Random graph for the Kleinberg-biased peer sampling service
- Metrics
  - Self-organization speed
    - Cycles needed before full routing success
  - Routing efficiency
    - Mean hops

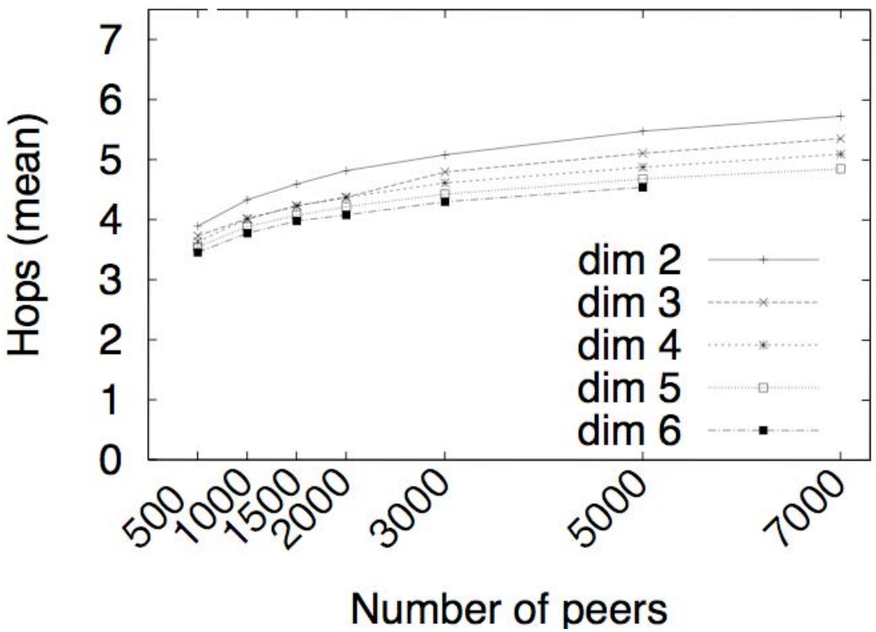
Self-organization speed



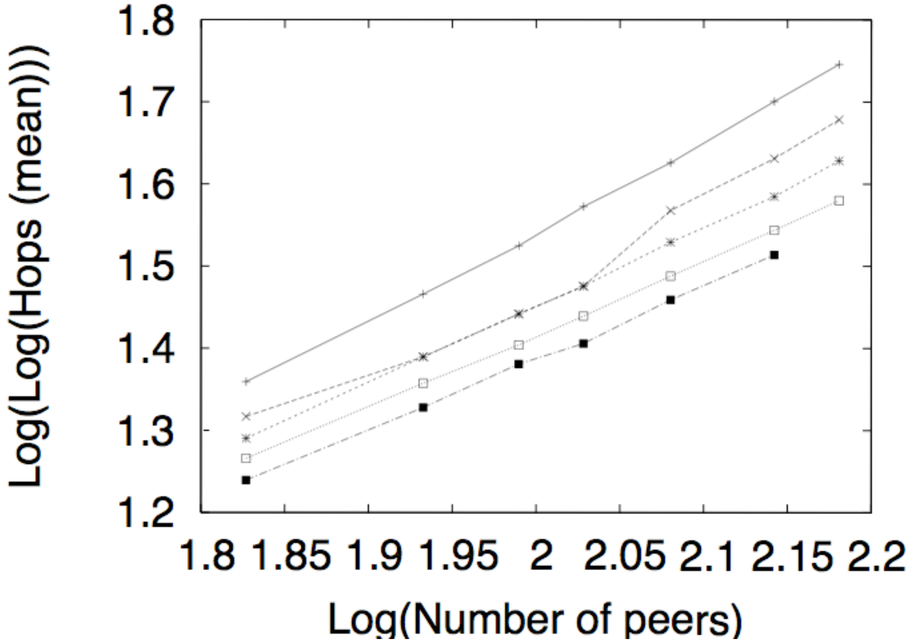
dimension = 2 *dimension = 6*

Less than 35 cycles of exchanges are needed
for reaching a structure where **all routes succeed** onto the correct object

Routing efficiency



a. *Routing hops*



b. Highlights O(log^x N) routing

Routing efficiency is achieved by the biased peer sampling layer

RayNet wrap-up

- RayNet, overlay for exhaustive and expressive queries
  - Self-organizing
  - Routing efficiency

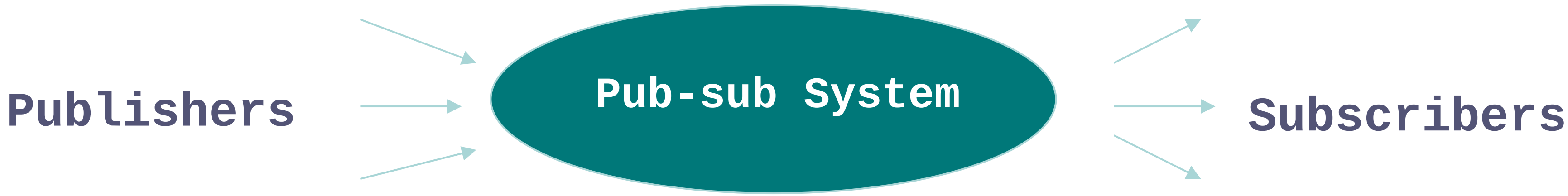
- Approximation of a complex & ‘ideal’ structure while still benefiting from its capacities
  - Expressiveness of the query model preserved
  - Efficient up to 10 dimensions

Gossiping for content-based publish-subscribe systems

Sub-2-sub [VRKvS, IPTPS 2006]

Pub-sub systems

- Asynchronous event notification system
- A set of Subscribers register their interest (subscriptions)
- A set of Publishers issue some events (events)



- Publish-subscribe system
  - Mapping between events and matching subscriptions
  - An event is delivered to all interested subscribers, and no others
  - Loosely coupled events sources and targets
- Flexible and seamless messaging substrate for applications

Pub-sub systems: expressiveness

- Differences in subscription expressiveness
- System classification
  - Topic-based

    = *(peer to peer)* Group Multicast

    Scribe (Pastry), CAN-Multicast, Bayeux (Tapestry), …

    ```
    topic=h
    ```

  - Attribute-based

    ```
    s1=(city=Rennes) (capacity=2_Bedrooms)
    ```

  - Content-based

    ```
    s1=(city=Rennes || Saint Malo) &&
    (capacity=3_Bedrooms || price < 300,000 EUR)
    ```

Content-based: *(semi-)*centralized solutions

- Current systems: One or more centralized servers (brokers)
  - e.g., Tibco (*Web Services Eventing*)

- Servers become a bottleneck/single point of failure
  - Reverse Path Forwarding
    - Notifications follow reverse paths of subscriptions
  - Brokers deliver events to interested subscribers
    - Brokers end up maintaining the whole set of subscriptions

  - network size increases
  - node churn increases
  - more events are published

- Triggered interest in decentralized P2P solutions

Sub-2-Sub

[Voulgaris & al, IPTPS 2006]

- Peers are subscriptions
  - Rather than physical nodes
  - Each peer manages its own subscription(s)
  - The more it subscribes, the more it contributes

- Self-organizing overlay
  - Eventually cluster similar subscriptions
  - Efficient event dissemination structure
  - Adapted to dynamism

Gossip-based algorithm to cluster peers according to their interests

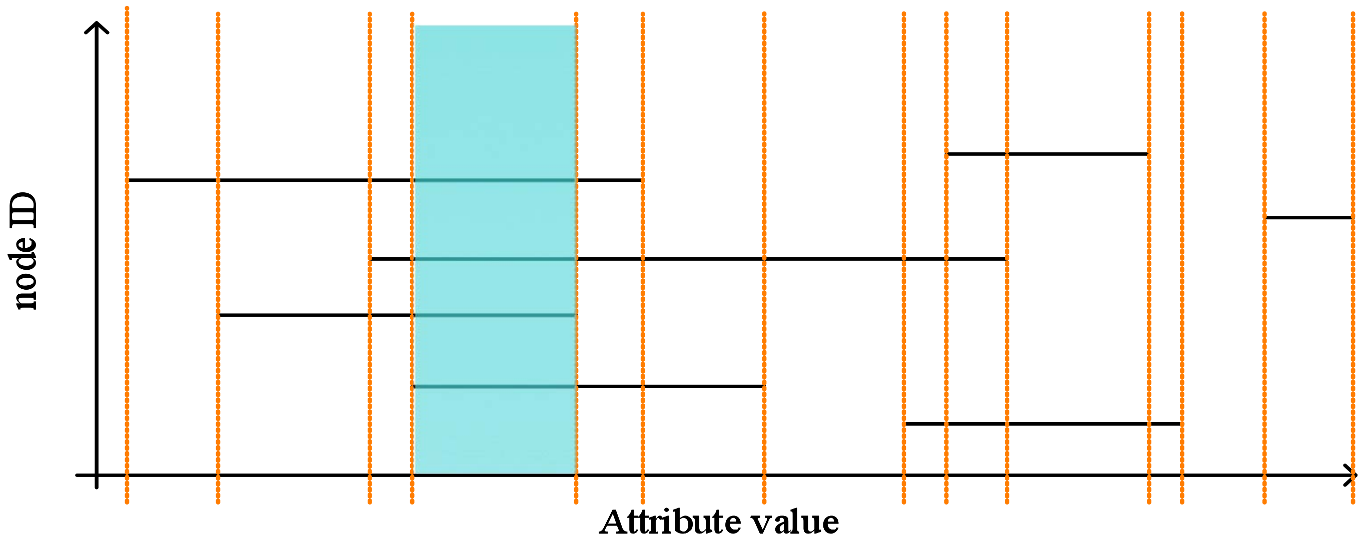
Sub-2-Sub: definitions

- Assume N attributes (real numbers)
  - A_1, A_2, ..., A_N
  - The N-hyperspace
- Subscriptions are range (trivially exact) predicates on one or more attributes
  - E.g. $A_2 == 3.07$ && $(2.5 < A_4 < 4.7)$
  - A N-hypercube
- Events define exact values for all attributes
  - E.g. $\{A_1, A_2, A_3, A_4\} = \{3, 0, 7, 10.5\}$
  - A point
- The set of all possible events define the event space, It's an continuous space of dimension N

Sub-2-Sub: Key Concept

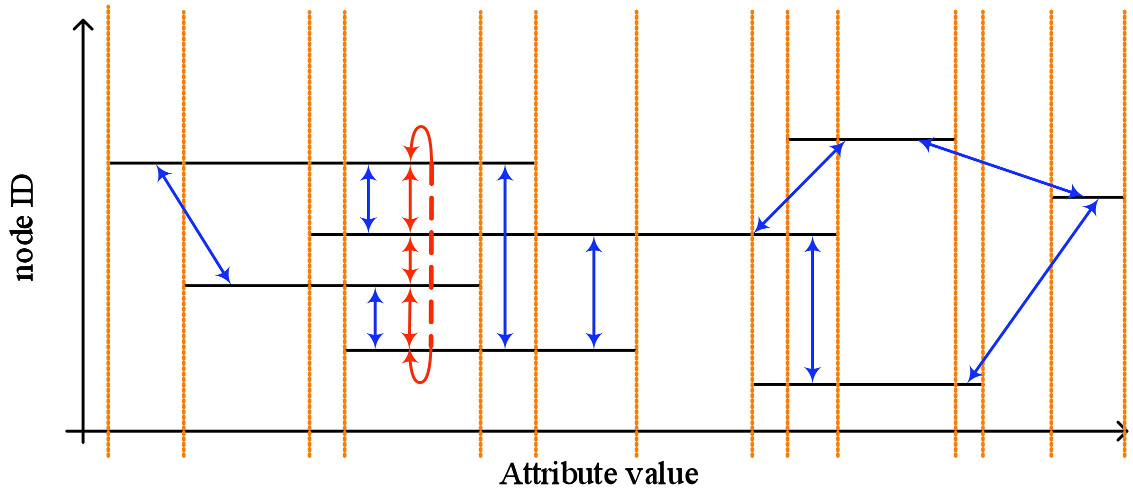
“Partition event space in homogeneous subspaces”

(homogeneous subspace: all its events have the same subscribers)



Sub-2-Sub: Operation

1. Let subscribers of “near” subspaces discover each other
2. Organize subscribers of each subspace in a ring
3. To publish an event, navigate to the right subspace, and hand the event to any one subscriber
   - Event reaches all and only interested subscribers, autonomously!



Sub-2-Sub overlay creation

• Maintain connectivity	• Peer sampling service
• Create clusters of “related” subcribers	• Clustering service
• Organize subscribers within a subspace in a ring ?	• Ranking service

Gossip around

Maintaining connectivity

- Connectivity = no overlay-network partition

- Peer sampling service: Cyclon

[S. Voulgaris, D. Gavidia, M. van Steen. *Journal of Network and Systems Management*, Vol. 13, No. 2, June 2005]

Forming clusters: gossip-based clustering

- Keep a small fixed-sized set of neighbors with similar interests

- Similarity is based on a notion of distance
  - the minimum Euclidean distance between two subscriptions

  (Note: Distance 0 means some overlapping interests)

- peerSelect()
  - Choose a neighbor in the random view provided by peer sampling
- select()
  - Keep neighbors of smallest distance

Organizing clusters in rings: gossip-based structuring

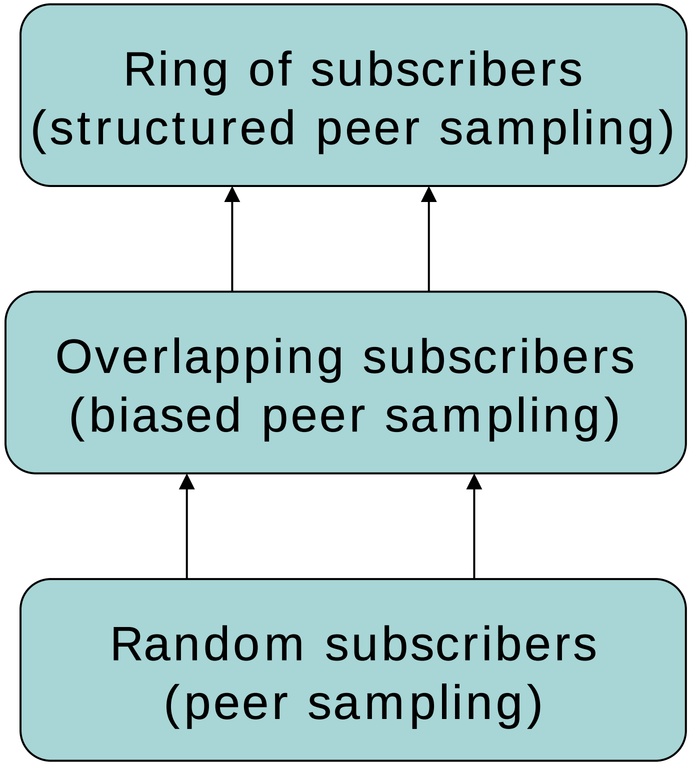
- Each subscription is given a fully random ID upon creation
  - Total order on IDs
  - Defined only to permit ring creation

- peerSelect()
  - Choose a neighbor in the similar interest view
- update()
  - Keep neighbors of smallest distance

- Distance definition :
  - 0 (ZERO), if overlapping and ID is the nearest for part of the supscriptions overlap
  - INFINITE, otherwise

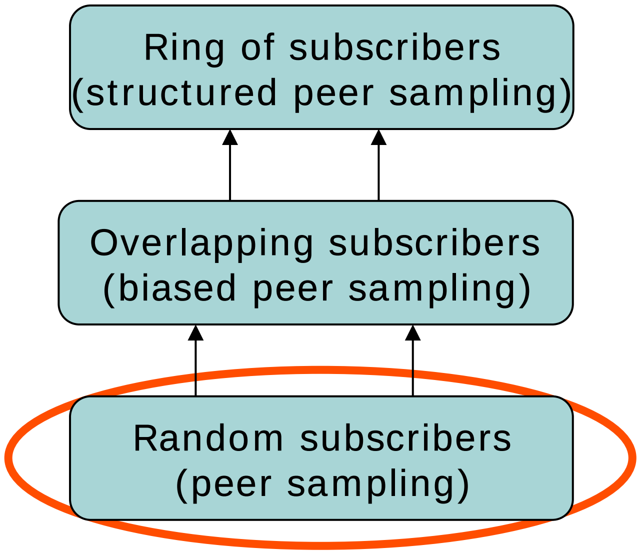
- update() keeps neighbors whose ID is nearer from the subscription ID for any portion of subscription hypercube
  - Size of the neighbor set depends on subscription width

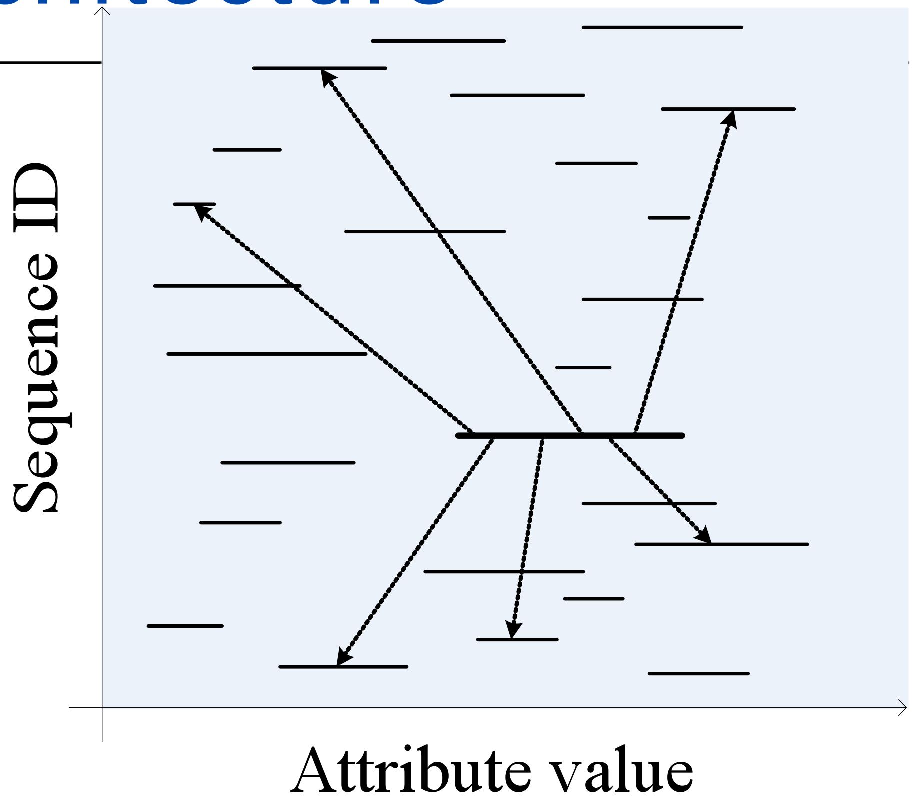
Sub-2-Sub Architecture



- Three-layer architecture
- Each layer gossips to a neighbor's respective layer

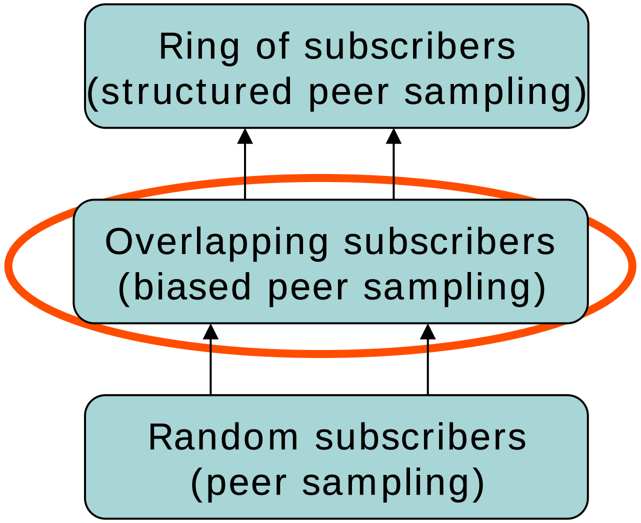
Sub-2-Sub Architecture

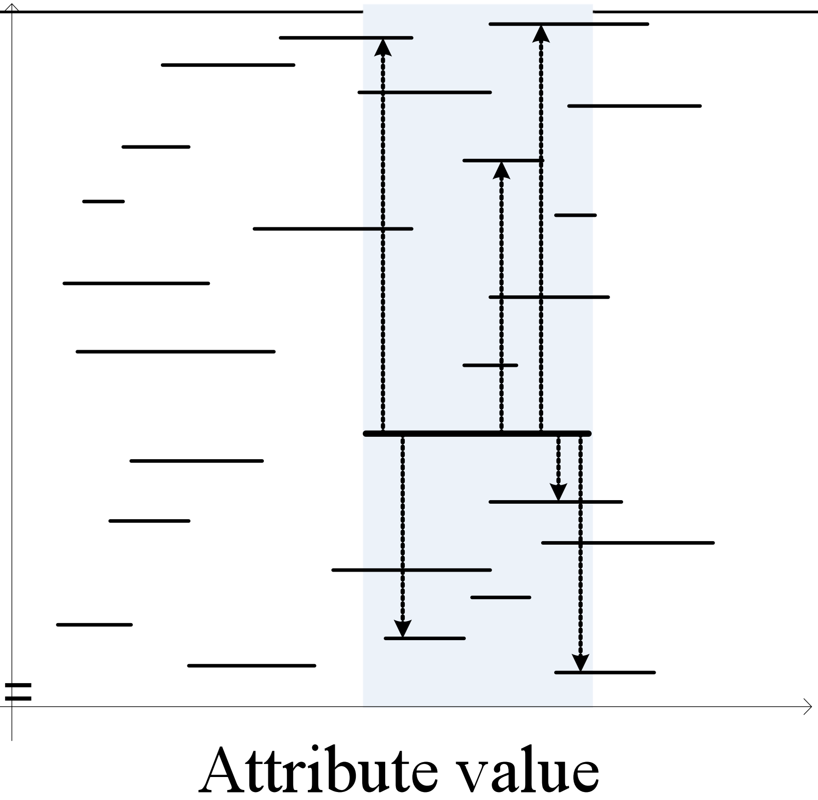




- RPS finds random links (needed for BPS)
- and keeps the overlay connected

Sub-2-Sub Architecture

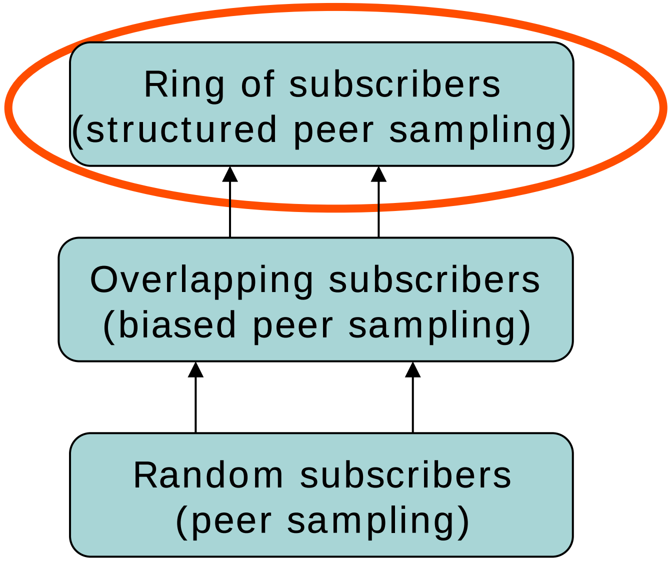


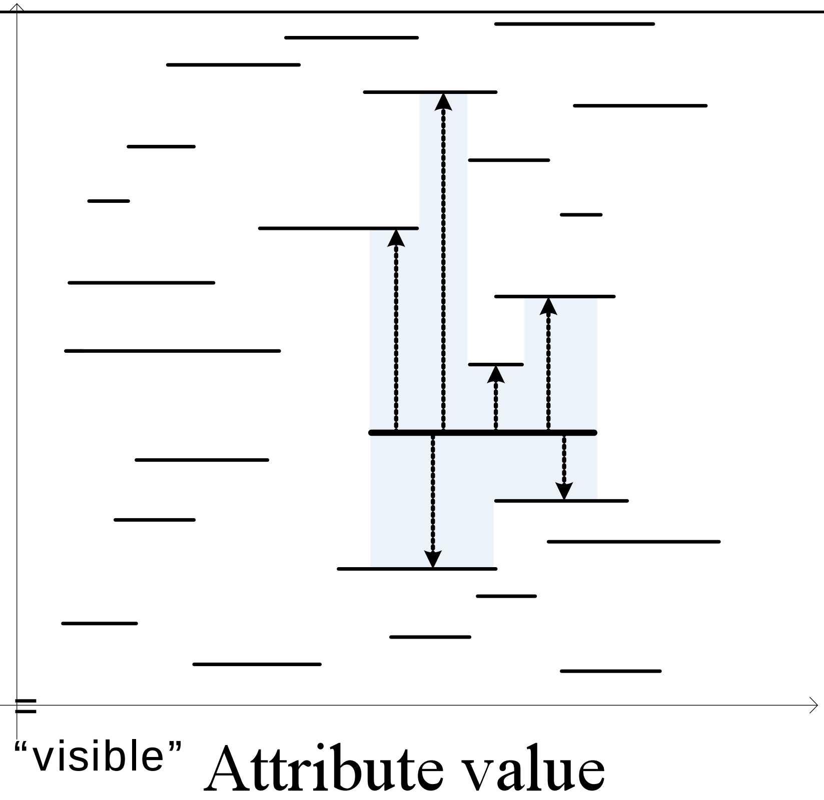


Proximity between 2 subscribers =

- 0 (ZERO), if overlapping
- the Euclidean distance between the 2 hypercubes, otherwise

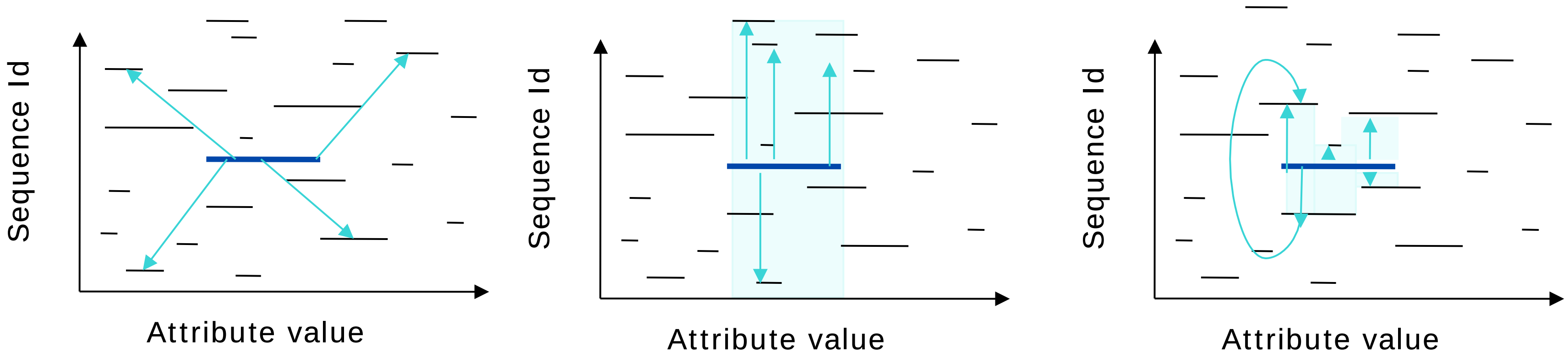
Sub-2-Sub Architecture





- Proximity between 2 subscribers =
  - 0 (ZERO), if overlapping <u>and</u> "visible"
  - INFINITE, otherwise
- Variable length view

Sub-2-Sub in a nutshell



Random links

-Traditional random-based epidemic algorithms

Interest links

- Peer selection and links kept based on proximity in the attribute space

d(i,j) = 0 if no overlap

$$S_i = [l_1^i, r_1^i] \times [l_n^i, r_n^i]$$

$$d(i,j) = \sqrt{\sum_{k=1}^{n} (\min(r_k^i, r_k^j) - \max(l_k^i, l_k^j))^2}$$

Structured links

- Peer selection: in the ring
- Links kept, sorted according to growing id
- Subscription cover

Dissemination of events

- The event is sent to any of the subscription peer
  - Greedy routing using Euclidean distance along random neighbors and interest proximity links
  - It eventually reach one of the interested subscriber ⇒ dissemination begins

- A node receiving an event for the first time, forwards it:
  - Along its two ring links
  - To one random subscriber interested in the event (if exists)

- Load balancing
  - Subscribers forward each event up to 3 times.

Sub-2-Sub summary

- Showed that a dedicated P2P present soundness for complex applications such as content/based
- Sub-2-Sub
  - Accurate → All and only interested nodes receive event
  - Autonomous → No need for extra device
  - Collaborative
  - Self-organized
  - Very scalable (nodes and attributes)
    - Experiments for 10 attributes present the same results
- Current work
  - Limiting the number if neighbours by articially manipulating the size of subscriptions

The take-away slide

Gossip-based protocols are a powerful tool in large-scale distributed computing

- Overlay maintenance
- Dissemination
- Search
- Distributed computations

You've seen a small subset only☺

An exciting research agenda

1. Coping with selfish behaviors
2. Coping with malicious nodes
3. Adapting protocols to node capacities
4. Leveraging multiple overlays
5. Increasing the target applications

References

[Birman & al, 1999] **Bimodal multicast**. K. P. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu and Y. Minsky", ACM Transactions on Computer System, 17(2) 1999.

[Bala & al, 2007] **Build One, Get One Free: Leveraging the Coexistence of Multiple P2P Overlay Networks.** Balasubramaneyam Maniymaran,Marin Bertier, Anne-Marie Kermarrec.**.** In *Proceedings of ICDCS 2007*, Toronto, Canada, June 2007.

[Beaumoont & al, 2007 b] Olivier Beaumont, Anne-Marie Kermarrec, Loris Marchal, Etienne Rivière. **VoroNet: a scalable object network based on Voronoi Tessellations**. In *Proceedings of 21st IEEE International Parallel & Distributed Processing Symposium (IPDPS).* Long Beach, CA, USA, March 2007.

[Beaumont & al,2007] Olivier Beaumont, Anne-Marie Kermarrec, Etienne Rivière*.* **Peer to peer multidimensional overlays: Approximating complex structures.** *In OPODIS, 11th International conference on principles of distributed systems, Guadeloupe, France, December 2007.*

[Bonnet&al, 2007] Francois Bonnet, Anne-Marie Kermarrec, Michel Raynal*.* **Small world networks: From theoretical bounds to pratcical systems***.In OPODIS, 11th International conference on principles of distributed systems, Guadeloupe, France, December 2007.*

[Demers & al, 1988] **Epidemic algorithms for replicated database maintenance.** A. Demers, D. Greene, C. Houser, W. Irish, J. Larson, S. Shenker, H Sturgis, D, Swinehart and D. Terry ACM SIGOPS Operating Systems Review . 22 (1). 1988.

[Eugster & al, 2003] **Lightweight Probabilistic Broadcast**. P. Eugster, S. Handurukande, R. Guerraoui, A.-M. Kermarrec, and P. Kouznetsov. *ACM Transaction on Computer Systems*, 21(4), November 2003.

[Eugster & al, 2004] **From Epidemics to Distributed Computing.** P. Eugster, R. Guerraoui, A.-M. Kermarrec, and L. Massoulié. *IEEE Computer*, 37(5):60-67, May 2004

References

[Fernandez & al] **Distributed Slicing in Dynamic Systems,** Antonio Fernández, Vincent Gramoli, Ernesto Jiménez, Anne-Marie Kermarrec, Michel Raynal, Proceedings of the 27th International Conference on Distributed Computing Systems (ICDCS'07) jun 2007

[Jelasity & al, 2003] **Newscast Computing** M. Jelasity, W. Kowalczyk, M. van Steen. Internal report IR-CS-006, Vrije Universiteit, Department of Computer Science, November 2003. *Submitted for publication.*

[Jelasity & al, 2004] **The Peer Sampling Service: Experimental Evaluation of Unstructured Gossip-Based Implementations.** M. Jelasity, R. Guerraoui, A.-M. Kermarrec, M. van Steen. *Proc. 5th ACM/IFIP/USENIX International Middleware Conference,* Toronto, Canada, Oct. 2004

[Jelasity & al, 2005] **Gossip-based aggregation in large dynamic networks** M. Jelasity, A. Montresor, and O. Babaoglu. *ACM Transactions on Computer Systems*, 23(3):219–252, August 2005.

[Jelasity & al, 2005] **Gossip-based aggregation in large dynamic networks** M. Jelasity, A. Montresor, and O. Babaoglu. *ACM Transactions on Computer Systems*, 23(3):219–252, August 2005.

[Jelasity & Kermarrec, 2006] **Ordered Slicing of Very Large-Scale Overlay Networks**. M. Jelasity and A.-M. Kermarrec. In *The Sixth IEEE Conference on Peer to Peer Computing (P2P)*, Cambridge, UK, 2006.

[Jelasity & Babaoglu, 2006] **Gossip-based overlay topology management.** M. Jelasity and O. Babaoglu. T-Man: In *Engineering Self-Organising Systems: Third International Workshop (ESOA 2005), Revised Selected Papers*, volume 3910 of *Lecture Notes in Computer Science*, pages 1–15. Springer-Verlag, 2006.

References

[Kermarrec & Steen, 2007] **Gossipong in Distributed Systems**. Anne-Marie Kermarrec & Maarten van Steen, ACM Operating System Review 41(5). October 2007

[Kermarrec & al, 2003] **Probabilistic Reliable Dissemination in Large-Scale Systems**. A.-M. Kermarrec, L. Massoulié, and A. J. Ganesh. *IEEE Transactions on Parallel and Distributed Systems*, 14(3), March 2003.

[Patel&al, 2006] **JetStream: Achieving Predictable Gossip Dissemination by Leveraging Social Network Principles** J. Patel, I. Gupta, N. Contractor Proc. of the Fifth IEEE Intnl. Symp. on Network Computing and Applications (NCA), pp. 32-39, 2006.

[Voulgaris & al, 2005] **CYCLON: Inexpensive Membership Management for Unstructured P2P Overlays.** (preprint). S. Voulgaris, D. Gavidia, M. van Steen. Journal of Network and Systems Management, vol. 13(2):197-217.

[Voulgaris & Steen, 2005] **Epidemic-style Management of Semantic Overlays for Content-Based Searching**. S. Voulgaris, M. van Steen. *Proc. Int'l Conf. on Parallel and Distributed Computing (Euro-Par),* Lisbon, Portugal, August 2005.

[Voulgaris & al, 2006] **SUB-2-SUB: Self-Organizing Content-Based Publish and Subscribe for Dynamic and Large Scale Collaborative Networks**. S. Voulgaris, E. Riviere, A.-M. Kermarrec, M. van Steen. *Proc. 5th Int'l Workshop on Peer-to-Peer Systems (IPTPS),* Santa Barbara, CA, February 2006