MIDLAB Middleware Laboratory

Università di Roma "La Sapienza"
Dipartimento di Informatica e Sistemistica

P2P

Peer-to-Peer Systems

Distributed Systems(AA 10/11)

Adriano Cerocchi
cerocchi@dis.uniroma1.it
http://www.dis.uniroma1.it/~cerocchi

Agenda

- Introduction
- P2P as middleware
- The churn phenomena
- Gnutella: a famous example
- What is an overlay network
- Peer Sampling
- Structured Vs unstructured overlay network:
  - Cyclon: an unstructured P2P system
  - Chord/DHT: a structured overlay network
  - P2P in domotics environment
  - Kelips: an hybrid system
- Search in P2P network

Agenda

- Introduction
- P2P as middleware
- The churn phenomena
- Gnutella: a famous example
- What is an overlay network
- Peer Sampling
- Structured Vs unstructured overlay network:
  - Cyclon: an unstructured P2P system
  - Chord/DHT: a structured overlay network
  - P2P in domotics environment
  - Kelips: an hybrid system
- Search in P2P network

Introduction

- A system is distributed when its functionalities are located in two or more actors
- A system is client/server when different actors implement different functionalities (horizontal partitioning)
- A system is P2P when each actor implements the same functionalities (vertical partitioning)

centralized

distributed
client/server

distributed
P2P

Introduction - P2P System: Definitions

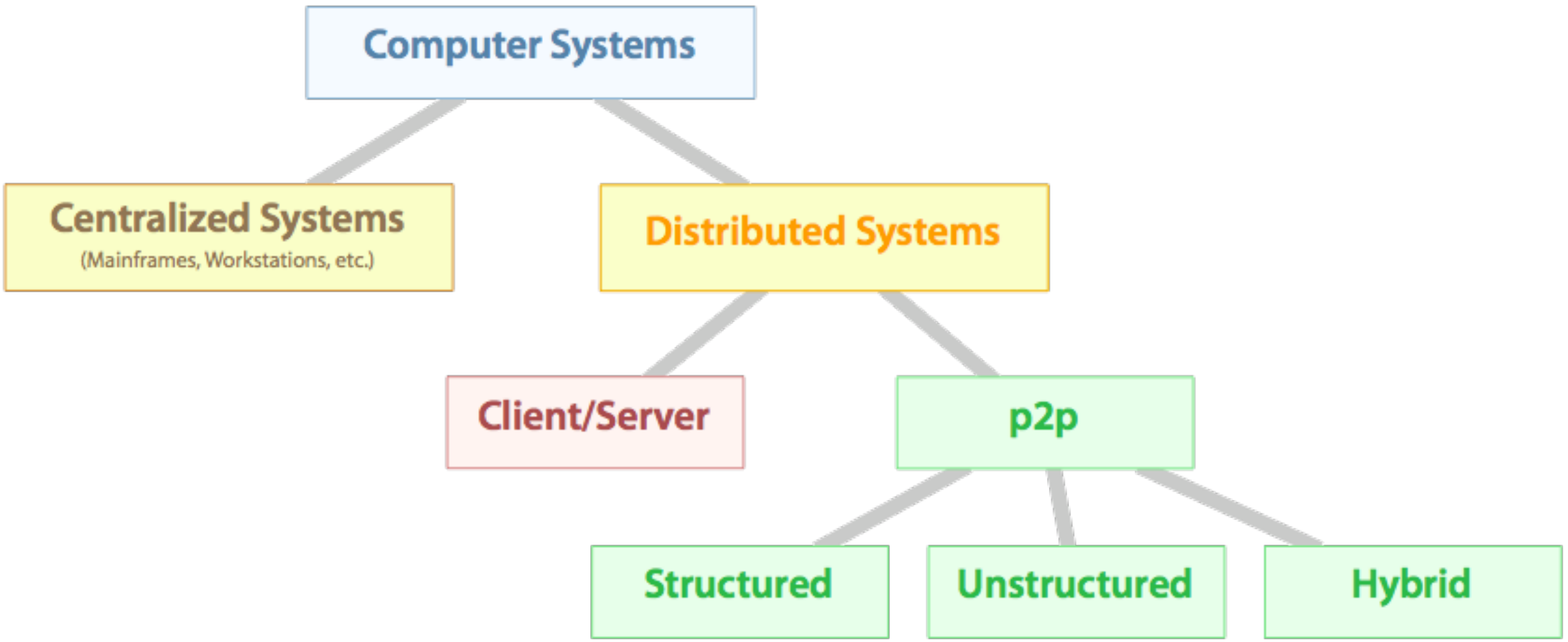
"P2P is a communications model in which each party has the same capabilities and either party can initiate a communication session" - *Whatis.com*

"P2P is a class of applications that takes advantage of resources – storage, cycles, content, human presence – available at the edges of the internet" - *Clay Shirky, O'Reilly*

"A type of network in which each workstation has equivalent capabilities and responsibilities" - *Webopedia.com*

"A P2P computer network refers to any network that does not have fixed clients and servers, but a number of peer nodes that function as both clients and servers to other nodes on the network" - *Wikipedia.org*

Introduction - Computer Systems



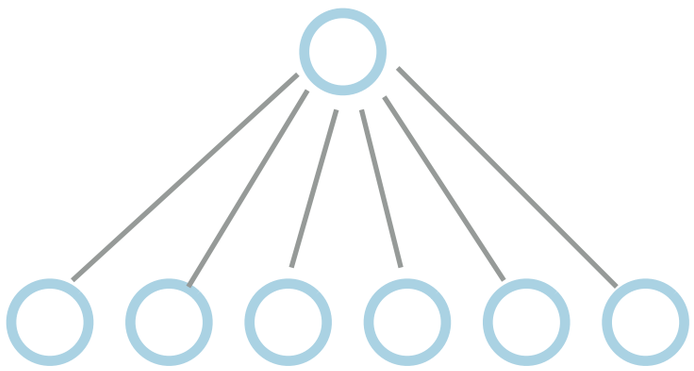
Introduction - Client/Server Vs P2P

Client/Server

Each node:
- knows the server
- does not interact with other non-server nodes

Each node knows who has the information and in any case it can ask to the server

Assumption: server and internet service available

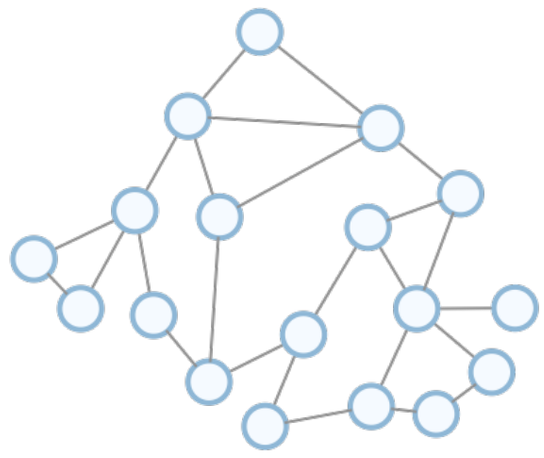


P2P

Each node:
- knows a set of other nodes named "neighbourhood" or "local view"
- interacts only with the neighbour nodes

Each node does not know who has the information, it can just try to ask to its own neighbourhood

Assumption: internet service available



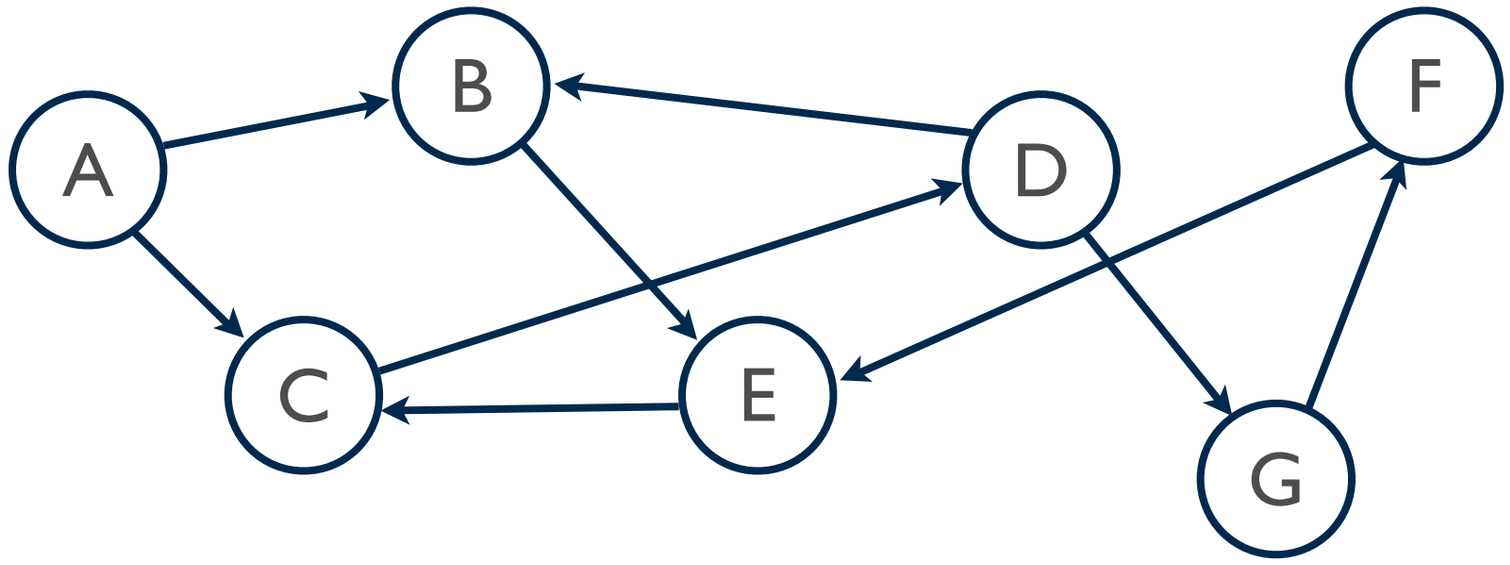
Introduction - P2P basic terms

- Neighbourhood and neighbours: the set of references to other peers owned by each node
- local view: it is a synonymus of neighbourhood
- churn or dynamism: it represent the presence of continuous joins and leaves of the network peer
- peer samplig: a service that should be able to select a uniform random peer in the network
- diameter: the maximum number of hops between two nodes of the network
- connectivity: the capability of the network to ensure that exists at least one path between two nodes

Introduction - about the Connectivity

The nodes and their neighbourhoods define an oriented graph:

A = {B,C}
B = {E}
C = {D}
D = {G,B}
E = {C}
F = {E}
G = {F}



Is the graph connected?

- no, in fact A has not in-coming edges, i.e. no one can reach A

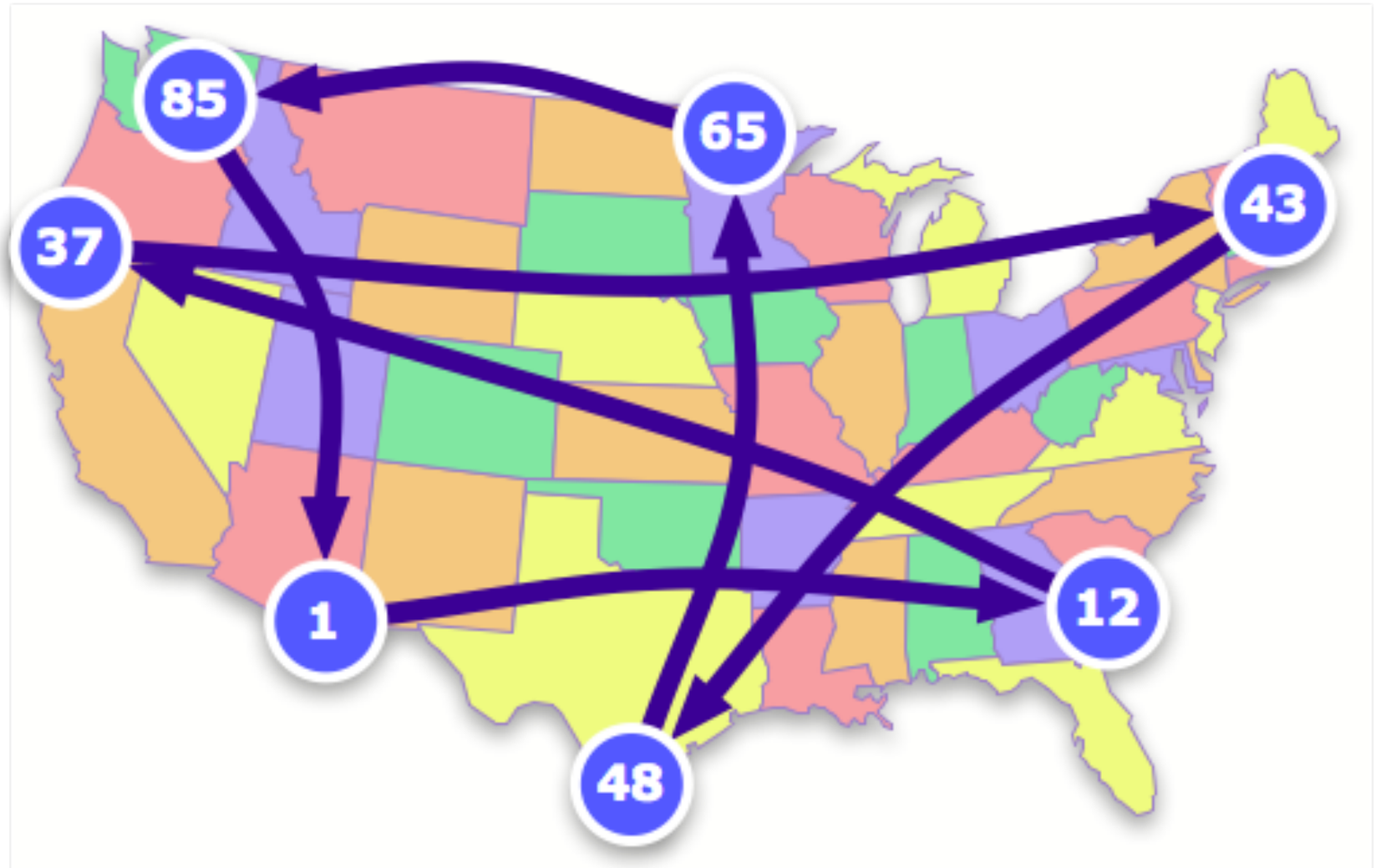
but...

- sometimes/often with "connectivity" is meant the connectivity of the associated undirected graph, then the graph is connected...

Introduction - logical Vs Physical network

The decoupling between "logic network" and "physical network" can introduce very bad performance in practical implementations.

"Neighbour" nodes in the logical network can be very far away in physical distance.



Introduction - Client/Server Architecture

- Client/Server paradigm is the most used communication paradigm in network communications.
- It is very simple and efficient in most of the cases
- Scalability of the client-server architecture it is not impossible but it has an high cost
- Server is the "single point of failure" of the whole system

Introduction - (Desired) Properties of a P2P System

- Scalability
  - A P2P should grow up to million of participating nodes, managing huge loads
- Fault Tolerance
  - A P2P system should work flawlessly also in presence of a large number of faults (crash or Byzantine)
- Dynamism
  - A P2P system should automatically manage the arrival/ departure of nodes

Introduction - Challenges of a P2P system

- Self-Organization
  - There is no central authority
  - Each node is autonomous
- Heterogeneous Nodes
  - Each node has different capabilities (CPU, memory, storage, bandwidth, etc...)
  - Heterogeneity must not influence performance of the whole system
- Churn Rate Resilience
  - High frequency of arrival/departure of nodes
  - No performance loss

Introduction - Time perspective

- P2P is not a new concept:
  - ARPANET
  - USENET
  - …
- The "hype" started with file sharing applications like Napster, Gnutella, **Kazaa** …
- Applications born from the smart intuition of simple developers, without any theoretical foundations.
- These application showed the potentialities of this approach, but where quickly superseded by more complex/efficient solutions.

Introduction - Time perspective

- Today the cloud computing and the renewed interest in cluster computing have strongly "resized" the interest in P2P research
- Google and other huge industries have proved that scalability is not a problem for centralized architectures
- Why should we be still interested in P2P?
  - cost prospective
  - disaster resilience:
    - 11th September 2001 - Twin towers - Internet resists...
    - 11th March 2011 - Japan Hearthquake - Internet resists...
    - 21th April 2011 - Amazon EC2 outage - A wrong procedure make the system unavailable for hours...
    - 29th April 2011 - Aruba UPS blaze - A "small" fault make the system unavailable for about half a day...
    - 10th May 2011 - Microsoft buys Skype (8.5 G$) (see next slide)

Introduction - Time perspective

- Skype:
  - born in 2003
  - 800 millions of users
  - 124 millions of pc connected each month
  - ~15-20 millions of users simultaneously connected (28 millions is the highest record - 17 january 2011)
  - 850 employees
  - International phone call maket share in 2005: 2.9%
  - International phone call maket share in 2010: 13%
  - Skype was developed by three people and zero server...

- Vodafone:
  - 252 milions of user
  - 60.000 employees

Users/Employees ratio
Skype: ~950K
Vodafone: ~4.2K

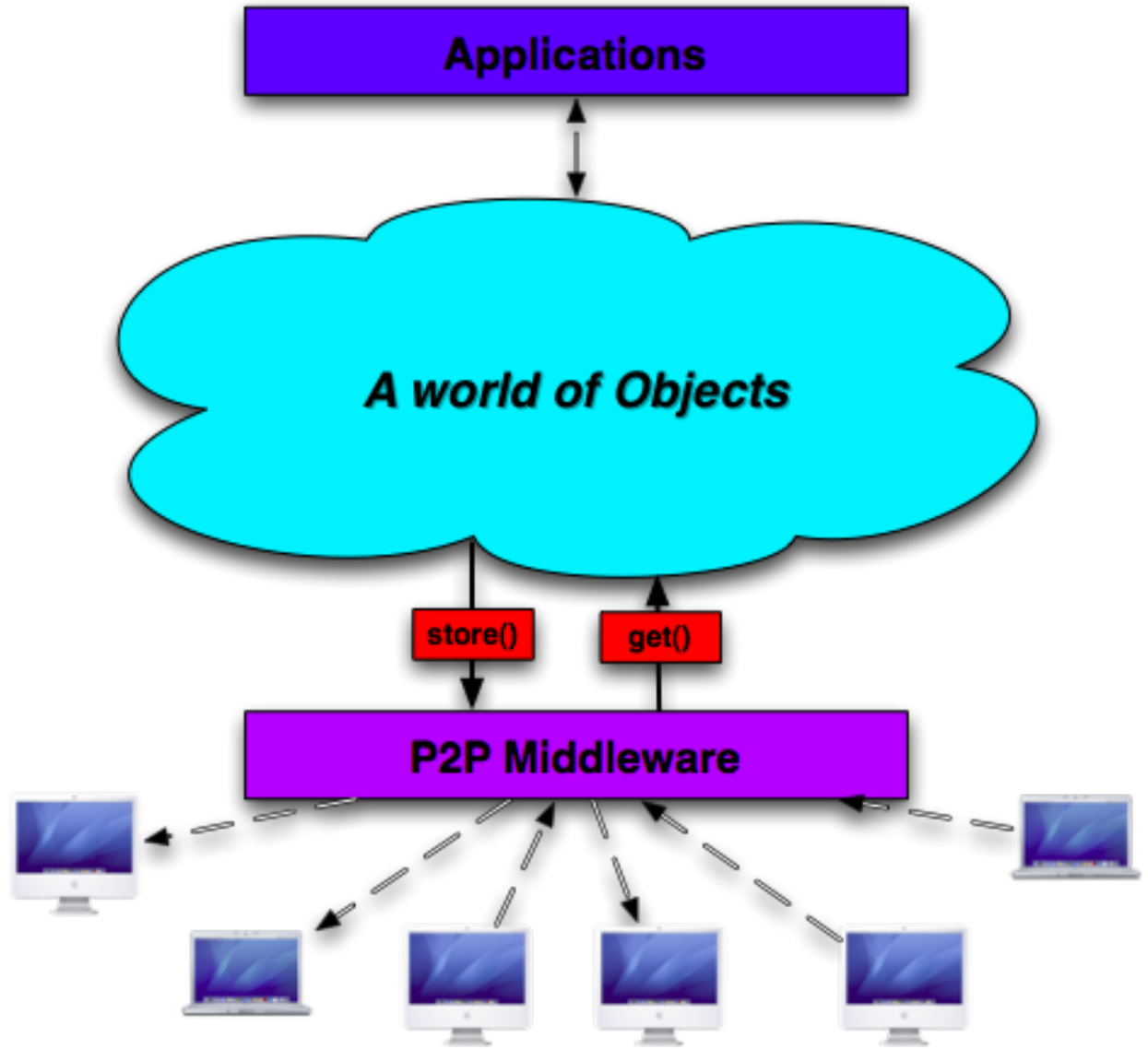
Agenda

- Introduction
- P2P as middleware
- The churn phenomena
- Gnutella: a famous example
- What is an overlay network
- Peer Sampling
- Structured Vs unstructured overlay network:
  - Cyclon: an unstructured P2P system
  - Chord/DHT: a structured overlay network
  - P2P in domotics environment
  - Kelips: an hybrid system
- Search in P2P network

P2P as a Middleware

The Peer-to-Peer model defines a communication paradigm where:

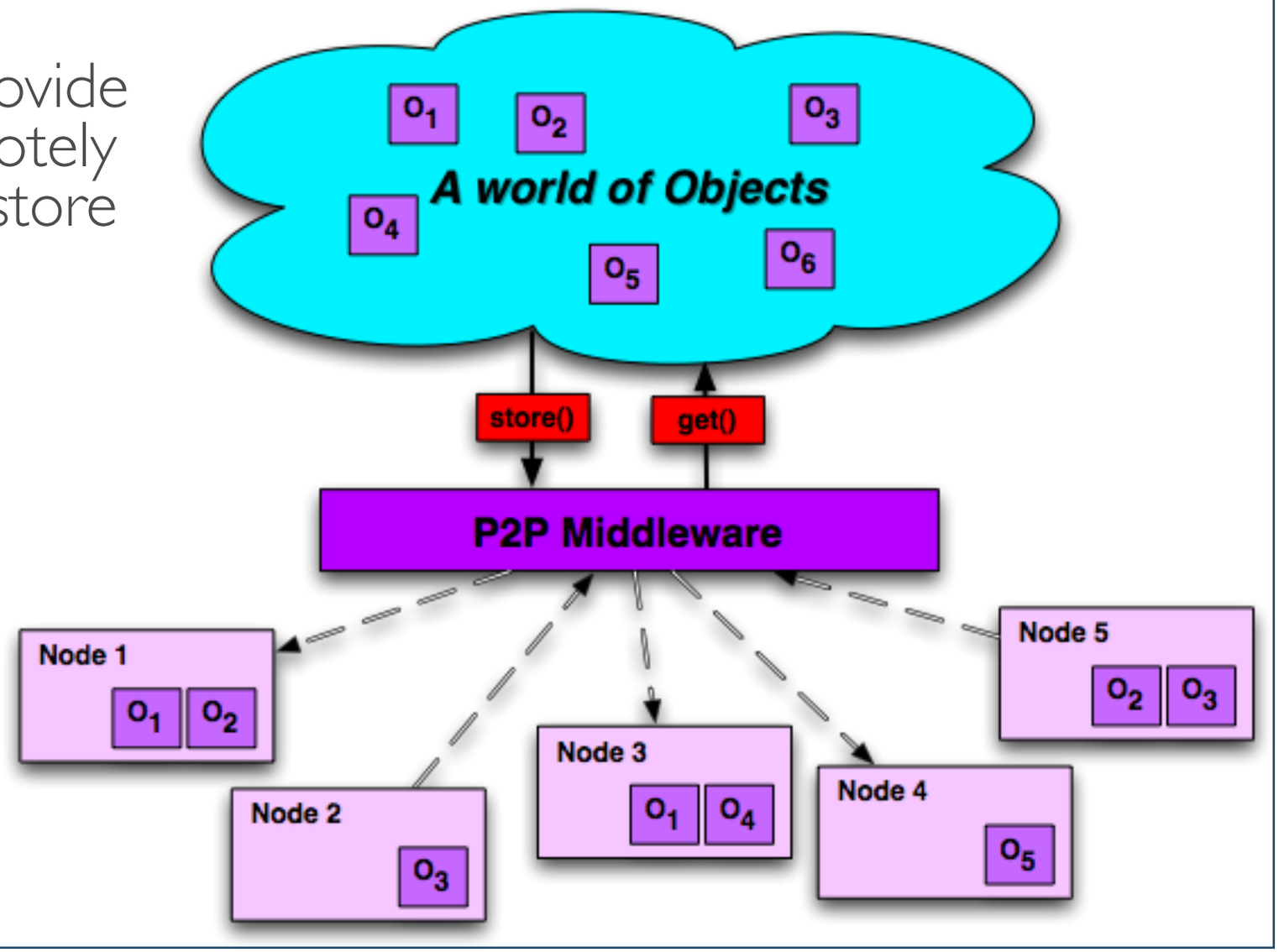
- there is a **decoupling** between **nodes** and **objects** that nodes have to maintain



P2P as a Middleware

The Peer-to-Peer model defines a communication paradigm where:

- there is a **decoupling** between **nodes** and **objects** that nodes have to maintain
  - p2p systems provide primitives to remotely search **get()** and store objects **store()**



P2P as a Middleware

The Peer-to-Peer model defines a communication paradigm where:

- there is a **decoupling** between **nodes** and **objects** that nodes have to maintain:
  - p2p systems provide primitives to remotely search **get()** and store object **store()**
- there is a resource **sharing** among nodes:
  - Resources can be cpu load, memory, bandwidth, ecc....
- there is a direct interaction between any pair of nodes (no intermediate)

Agenda

- Introduction
- P2P as middleware
- The churn phenomena
- Gnutella: a famous example
- What is an overlay network
- Peer Sampling
- Structured Vs unstructured overlay network:
  - Cyclon: an unstructured P2P system
  - Chord/DHT: a structured overlay network
  - P2P in domotics environment
  - Kelips: an hybrid system
- Search in P2P network

The Churn Phenomena

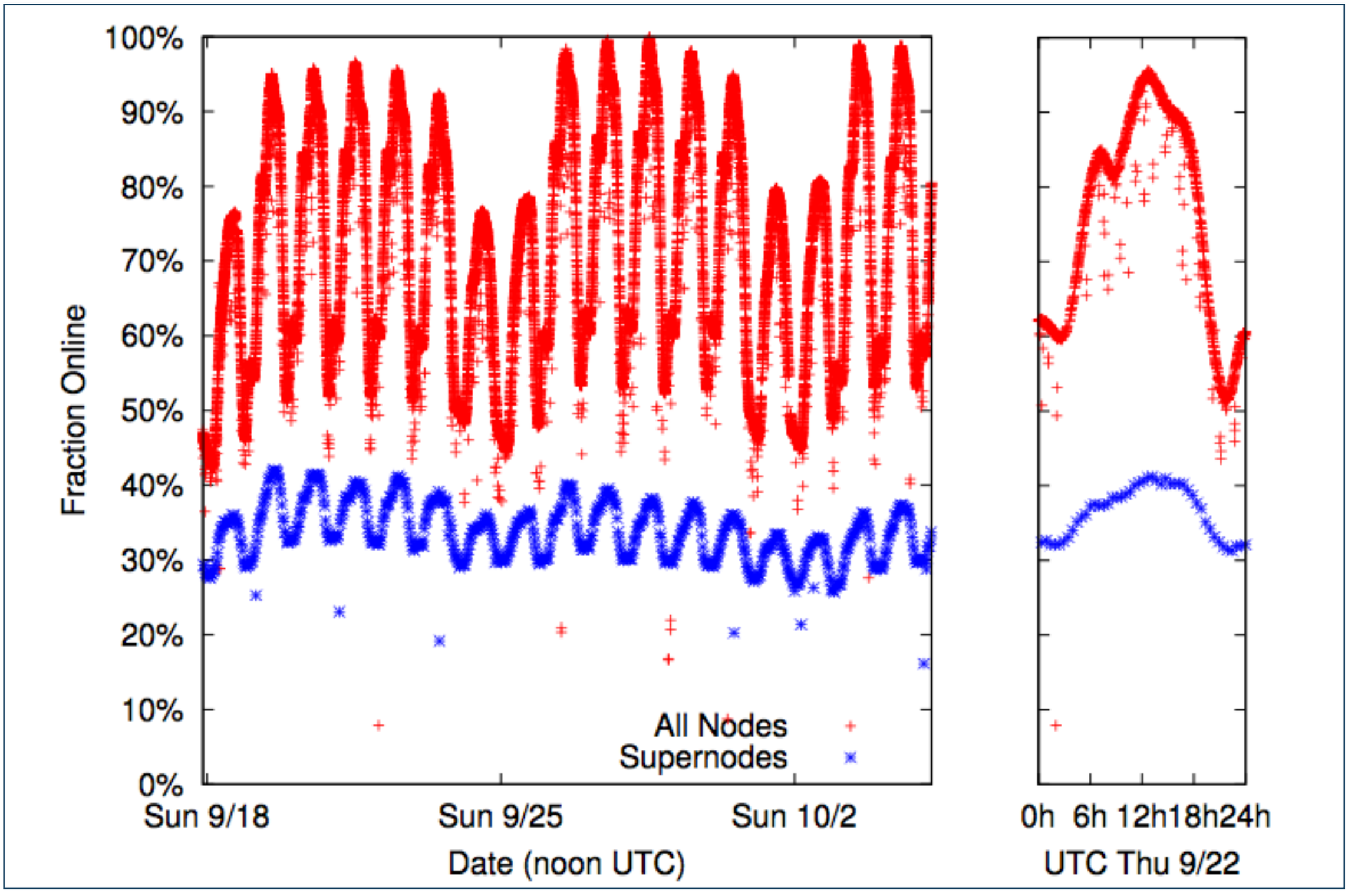
- Static Distributed System
  - There is a fixed number N of nodes in the system
  - Nodes can crash
- Dynamic Distributed System
  - The number of nodes in the system can change continuously
  - Nodes can crash or join/leave the system

The Churn Phenomena - Session Time

Systems Observed	Session Time
Gnutella, Napster	50% $\leq$ 60 min.
Gnutella, Napster	31% $\leq$ 10 min.
FastTrack	50% $\leq$ 1 min.
Overnet	50% $\leq$ 60 min.
Kazaa	50% $\leq$ 2.4 min.

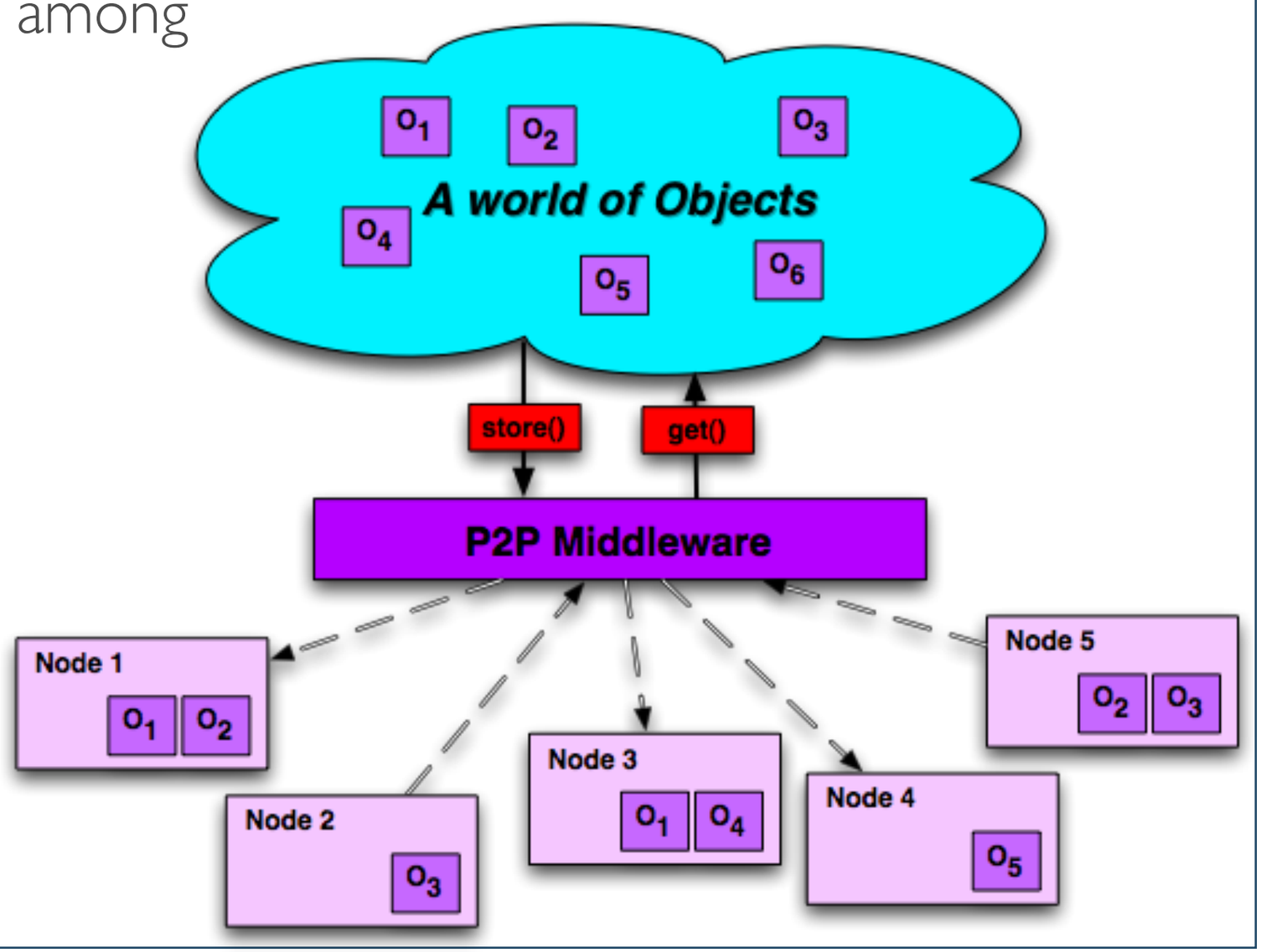
Usually the join/leave rate in file sharing systems is ~0.1% per second

The Churn Phenomena - common behaviour (Skype net. size)



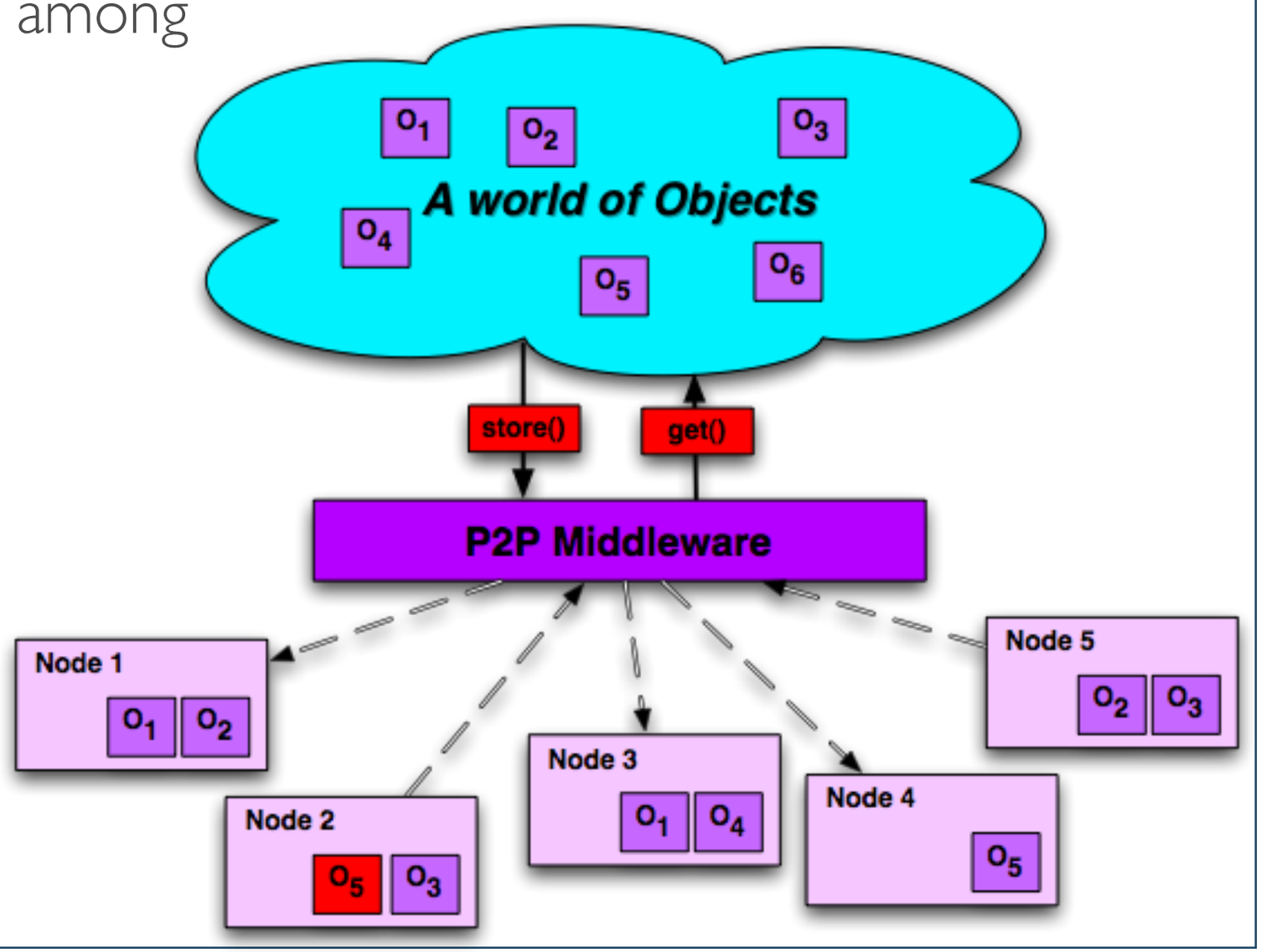
The Churn Phenomena

- P2P system works in presence of nodes that join/leave the system
  - Objects migrate among nodes



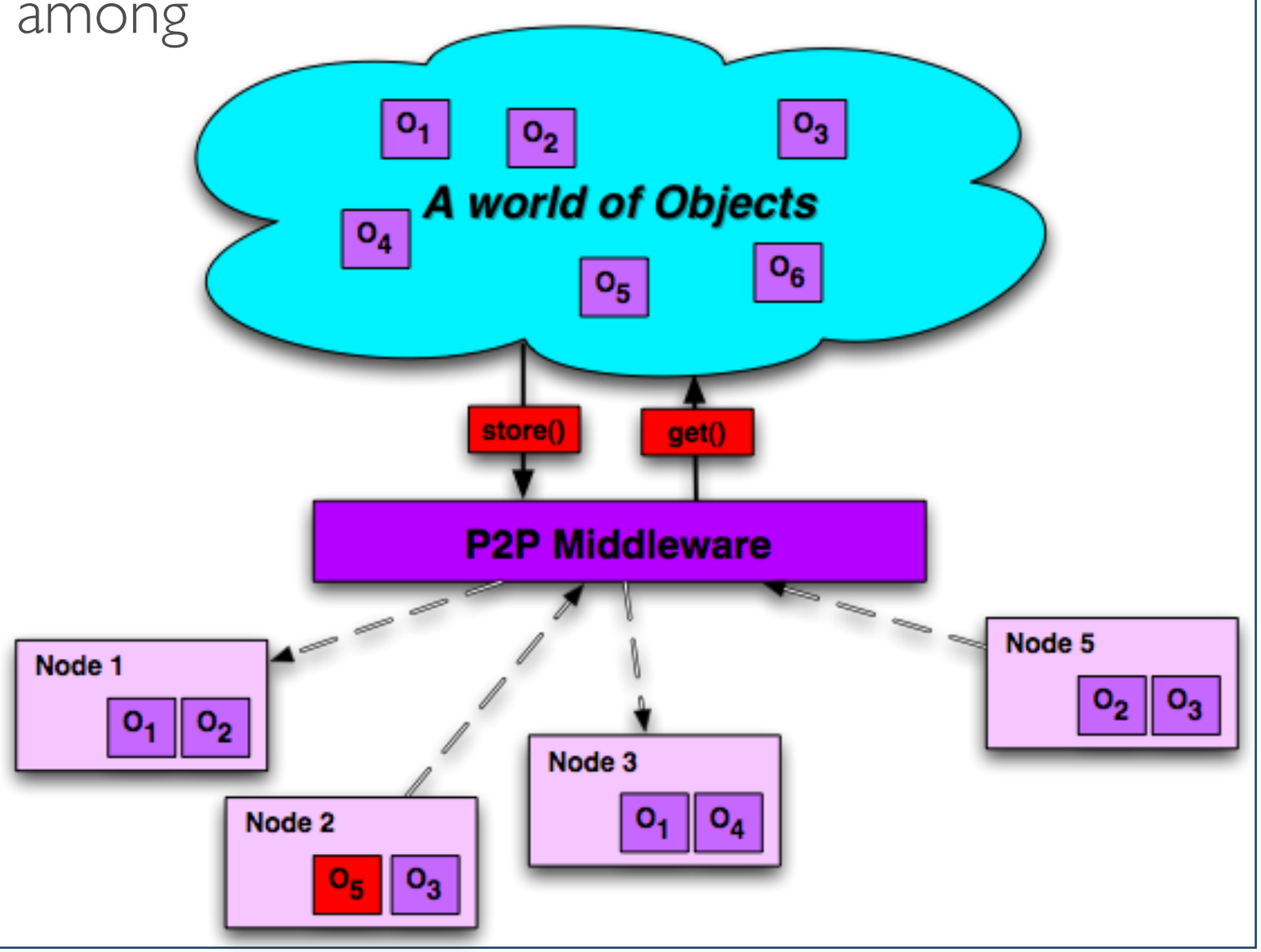
The Churn Phenomena

- P2P system works in presence of nodes that join/leave the system
  - Objects migrate among nodes



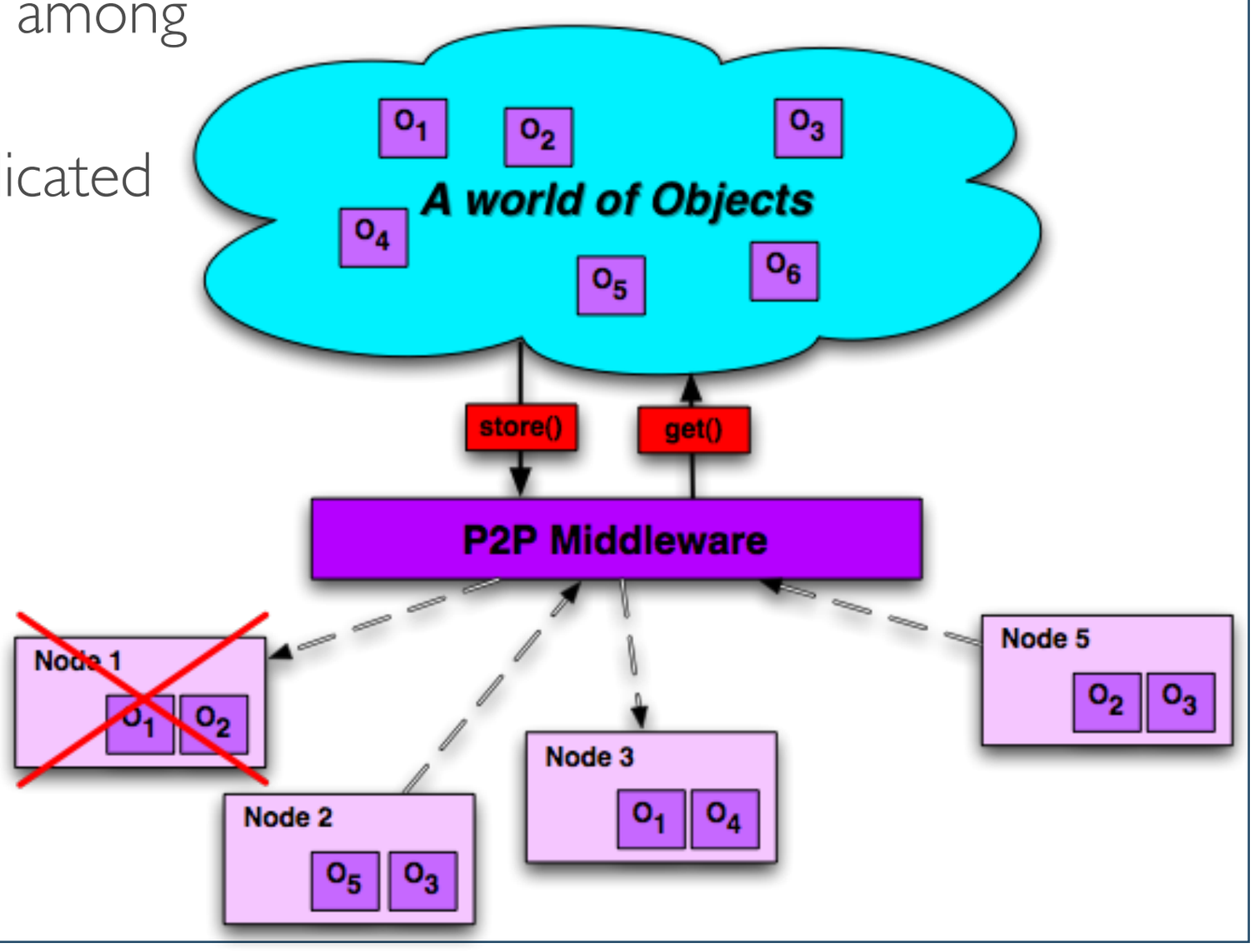
The Churn Phenomena

- P2P system works in presence of nodes that join/leave the system
  - Objects migrate among nodes



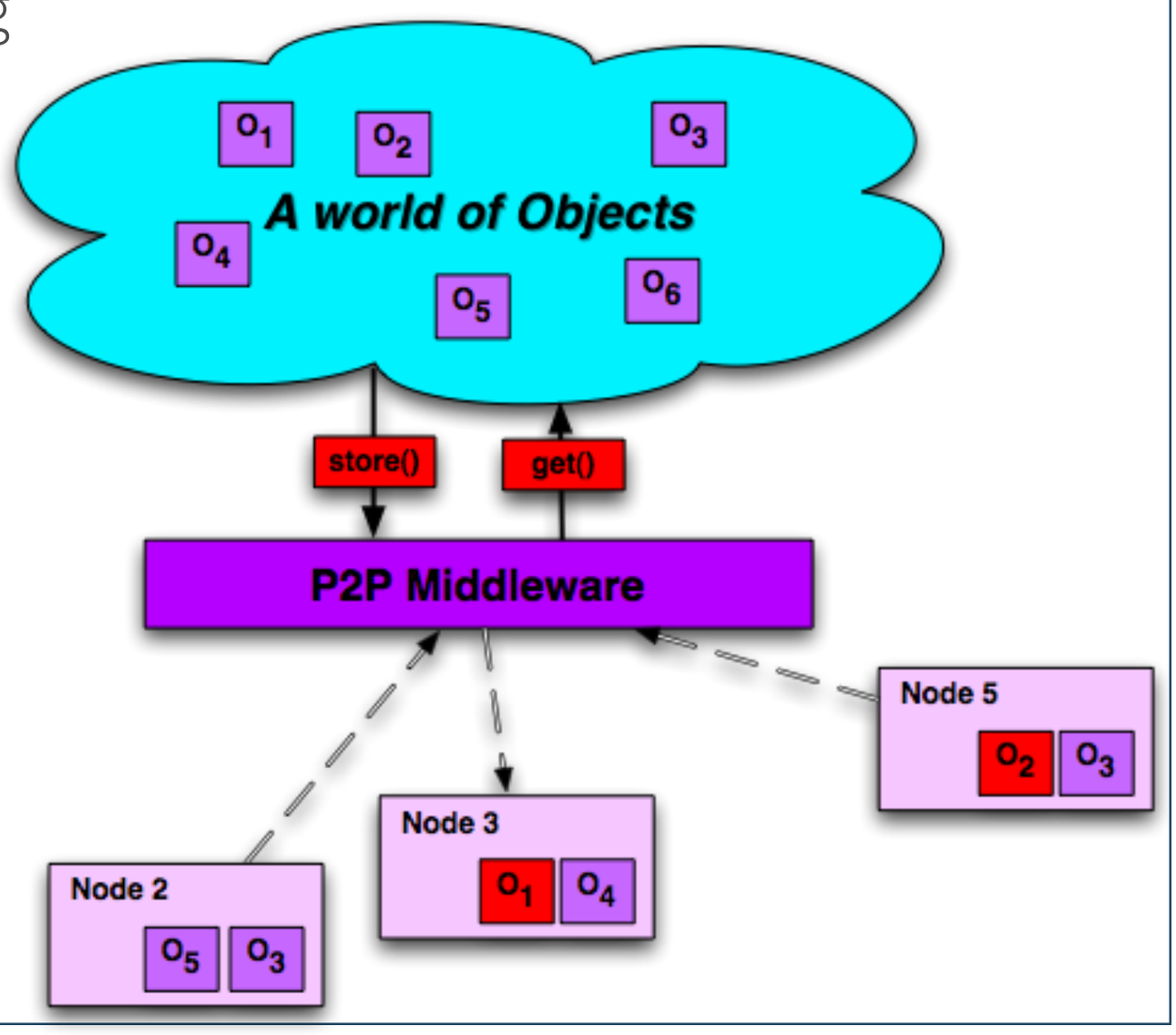
The Churn Phenomena

- P2P system works in presence of nodes that join/leave the system
  - Objects migrate among nodes
  - Objects are replicated among nodes



The Churn Phenomena

- P2P system works in presence of nodes that join/leave the system
  - Objects migrate among nodes
  - Objects are replicated among nodes



Agenda

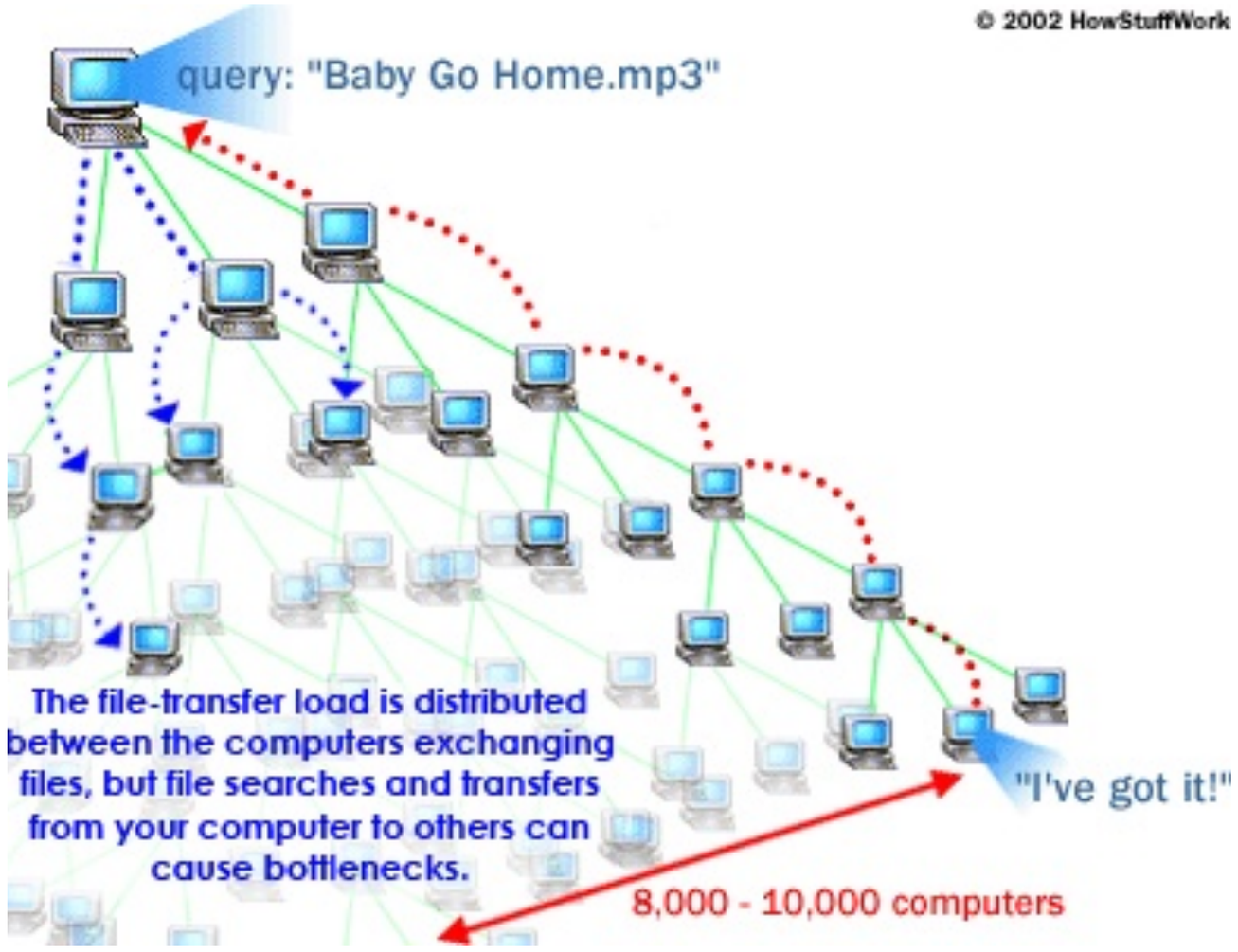
- Introduction
- P2P as middleware
- The churn phenomena
- Gnutella: a famous example
- What is an overlay network
- Peer Sampling
- Structured Vs unstructured overlay network:
  - Cyclon: an unstructured P2P system
  - Chord/DHT: a structured overlay network
  - P2P in domotics environment
  - Kelips: an hybrid system
- Search in P2P network

Gnutella: a famous example

Gnutella allows to share every kind of files.

The search is realized using ***flooding*** in a decentralized manner.

Each user asks objects to its neighbors. They forward the request to their neighbors and so...



After a predefined number of forwards (TTL) the request message is discarded.

Each user having the object replays to the requiring user.

Gnutella: a famous example

The system is completely decentralized

- There is not a single point of failure
- It does not suffer from DoS attacks
- It is not able to always provide correct results.

Searching objects using "flooding" allows to realize a completely decentralized search.

- Distributed but not scalable

Agenda

- Introduction
- P2P as middleware
- The churn phenomena
- Gnutella: a famous example
- What is an overlay network
- Peer Sampling
- Structured Vs unstructured overlay network:
  - Cyclon: an unstructured P2P system
  - Chord/DHT: a structured overlay network
  - P2P in domotics environment
  - Kelips: an hybrid system
- Search in P2P network

What is an overlay network

Even if file sharing applications continue to meet huge success (BitTorrent, emule, etc.) a new generation of P2P system is emerging today:

Overlay networks: distributed protocols that offer primitives implemented through a p2p approach:

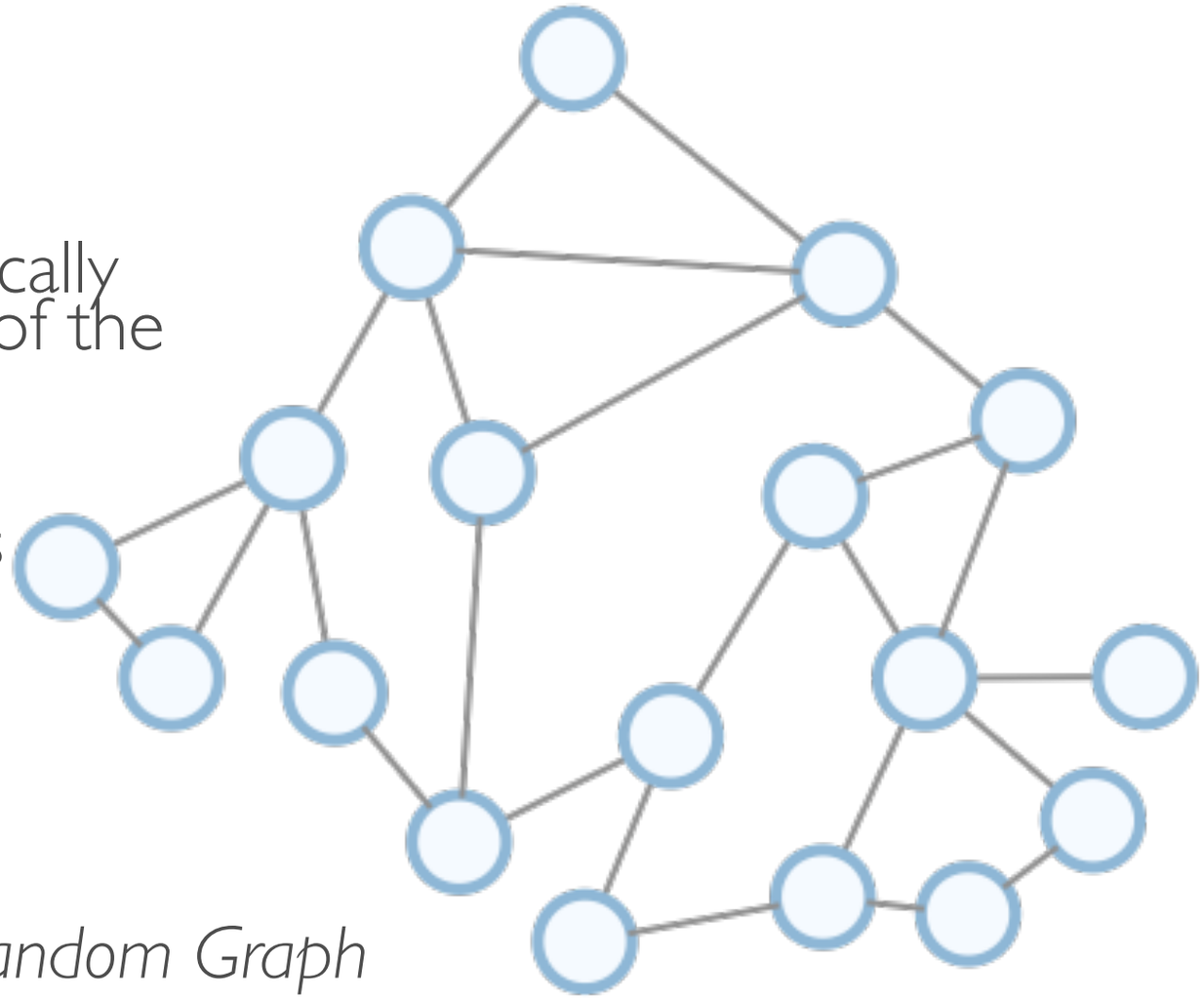
- An overlay network is an application-level network of nodes built on top of an existing network substrate (often the Internet)
- These nodes all play the same role
- They cooperate to offer complex primitives
- These primitives can be leveraged by distributed applications built on top of them

Unstructured Systems

- Each Object is located on the providing node
- The system is organized according to very simple rules
  - Searching an object can be hard because of the lack of "precise" organization
  - Adding/removing nodes is a relatively simple and cheap operation

What is an overlay network - Unstructured Networks

- Unstructured systems usually try to realize random graphs
- Random graphs have two fundamental properties:
  - Logarithmic diameter
  - Strong connectivity
- Performance of get() is drastically dependent from the diameter of the connectivity graph
- Resilience to failures depends from connectivity

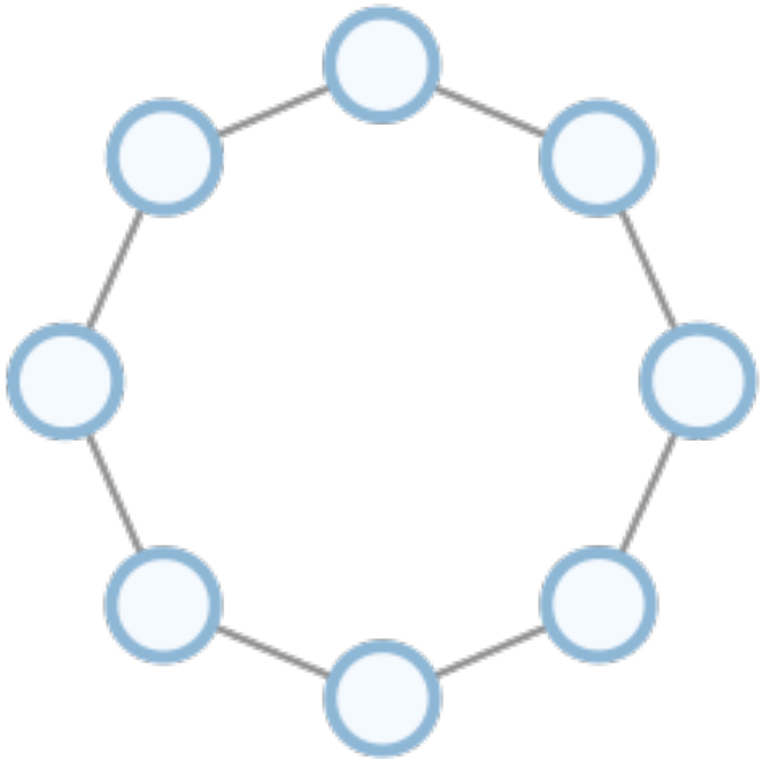


Random Graph

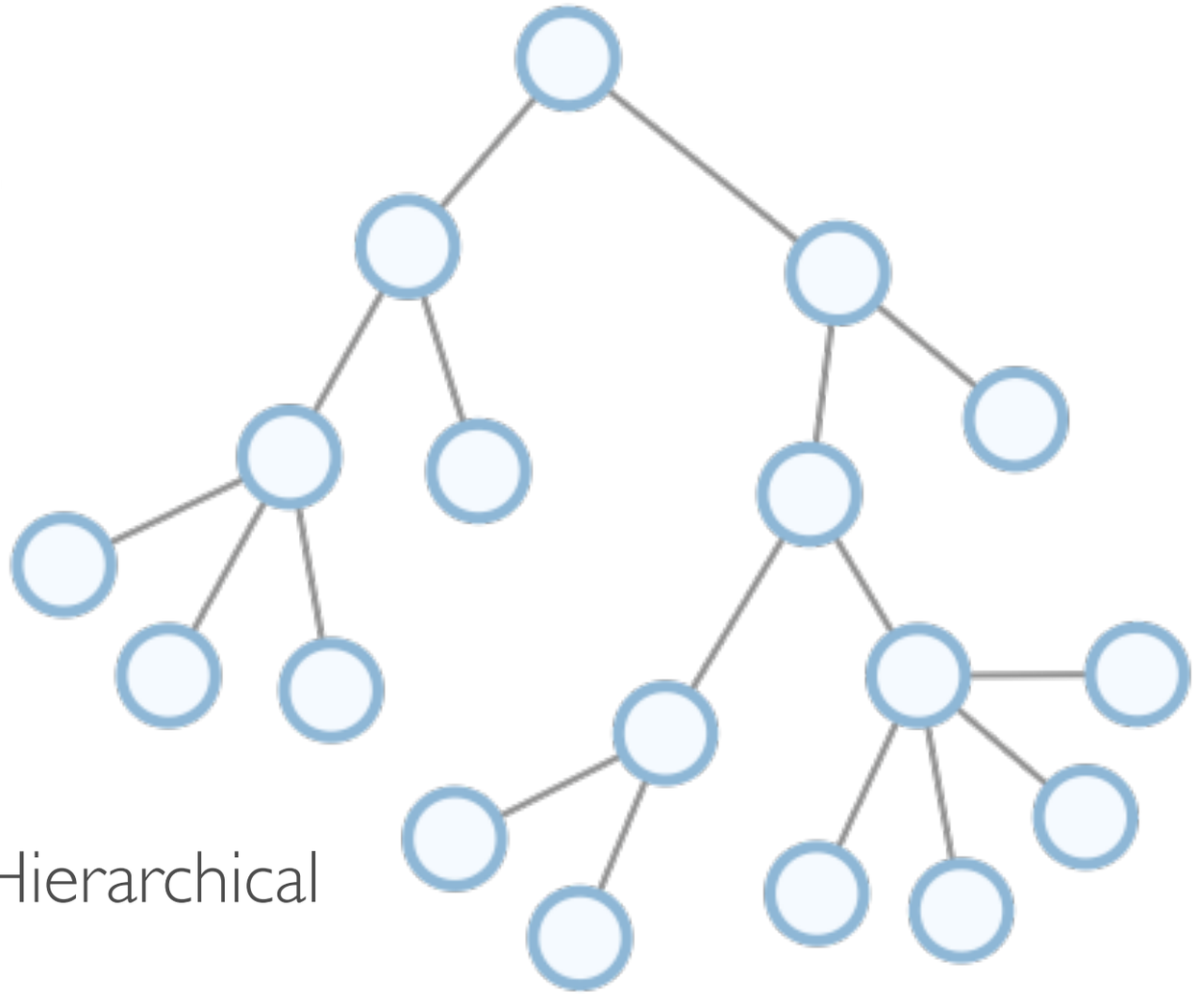
Structured Systems

- The location of an object in the system could be with the help of the structured topology
- example: it is possible to established computing a function f() on an identifier of the object itself.
- The organization of the system follows strict rules.
  - Locating an object is a trivial operation (ex: computing the function f() on the identifier of the object we can directly know where the object is)
  - Adding/removing nodes is usually an expensive operation

What is an overlay network - Structured Networks



Ring



Hierarchical

What is an overlay network

- the structure imposed by the structured topologies have to deal with the churn
- the presence of the churn force to implement some recovery/ managing procedure that often impose replication
- some procedure that run on specific architecture are very difficult to implement in p2p system:
  - example: the aggregate queries require usually a static spanning-tree to be run but how can we implement a tree overlay topology in presence of churn? (see next slides)

Agenda

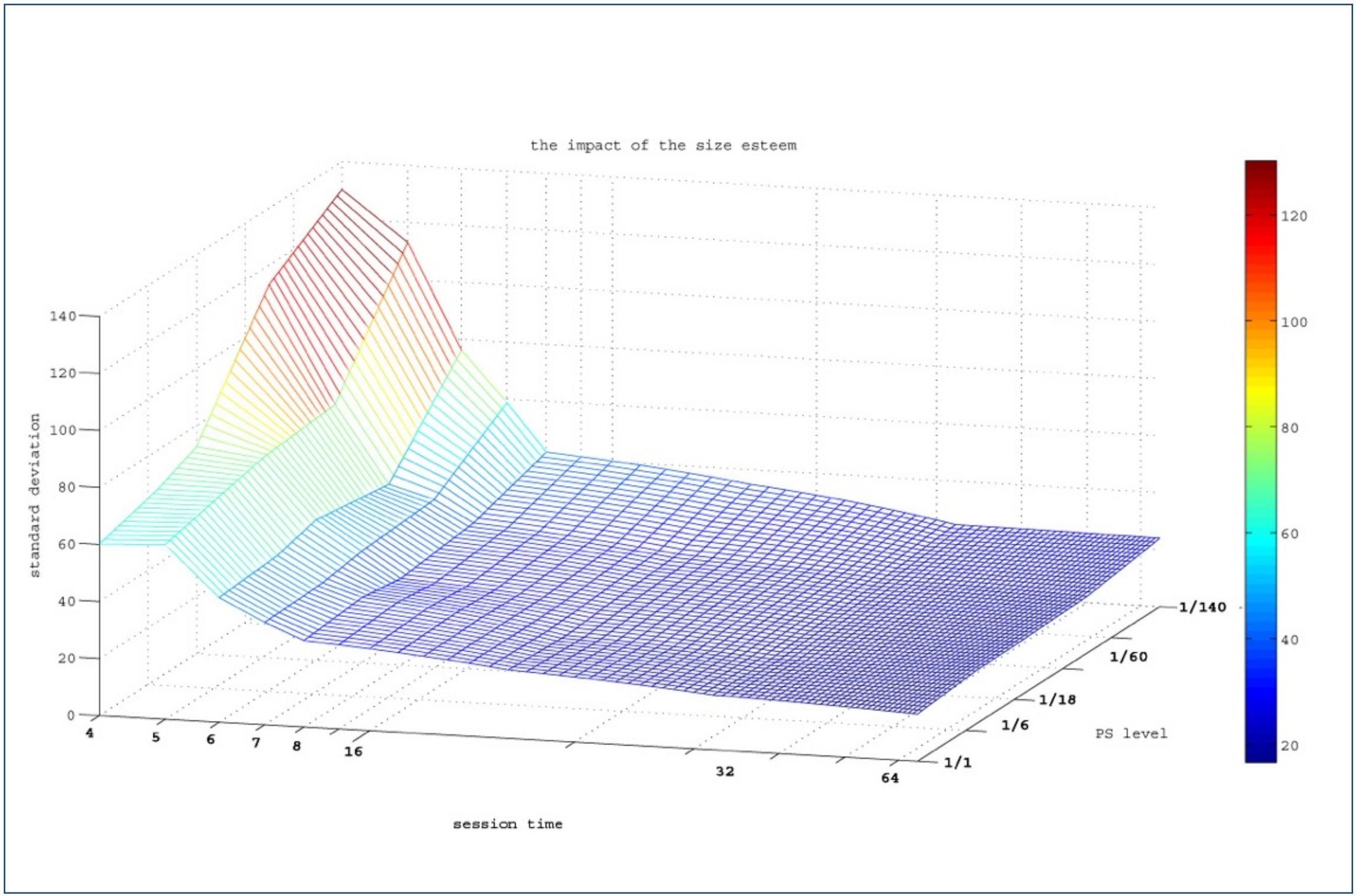
- Introduction
- P2P as middleware
- The churn phenomena
- Gnutella: a famous example
- What is an overlay network
- Peer Sampling
- Structured Vs unstructured overlay network:
  - Cyclon: an unstructured P2P system
  - Chord/DHT: a structured overlay network
  - P2P in domotics environment
  - Kelips: an hybrid system
- Search in P2P network

Peer Sampling

- In P2P systems is commonly implemented the Peer Sampling Service
- Peer Sampling is accessible trough primitive ***getPeer()***
- getPeer() (should) gives back a uniform random peer of the P2P network - it is a research challenge

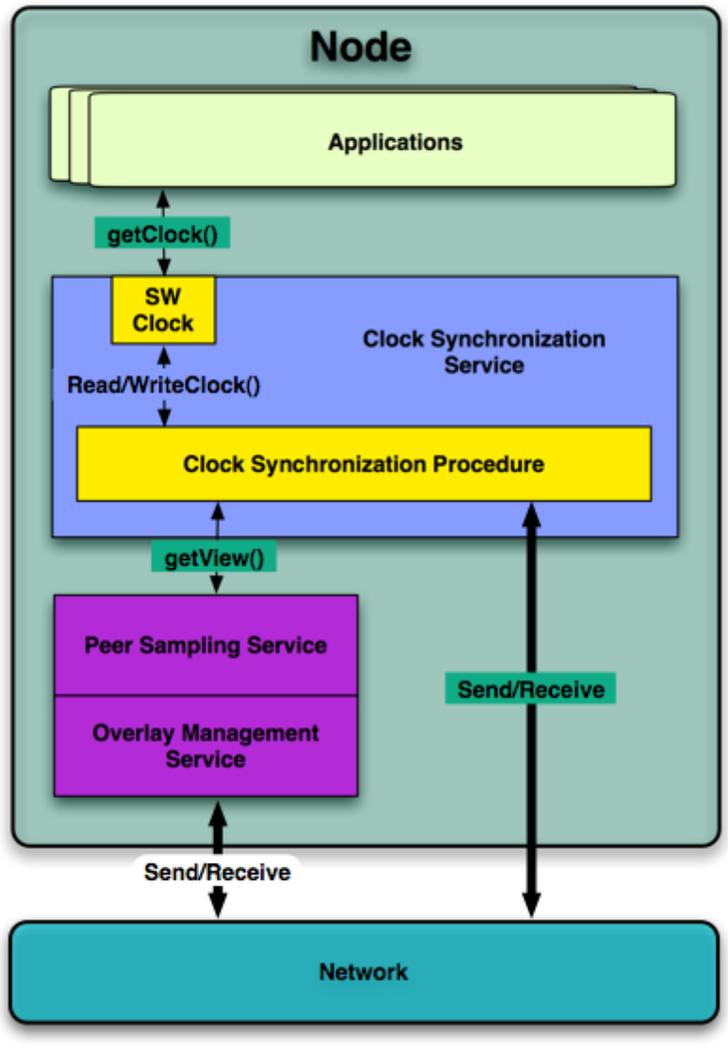
- With the peer sampling it is possible to:
  - build neigbourhood
  - implement **clock synchronization**
  - performe statistics and estimation
  - ...

Peer Sampling - The impact of a bad peer sampling service on Tera protocol



Peer Sampling - Clock Sync

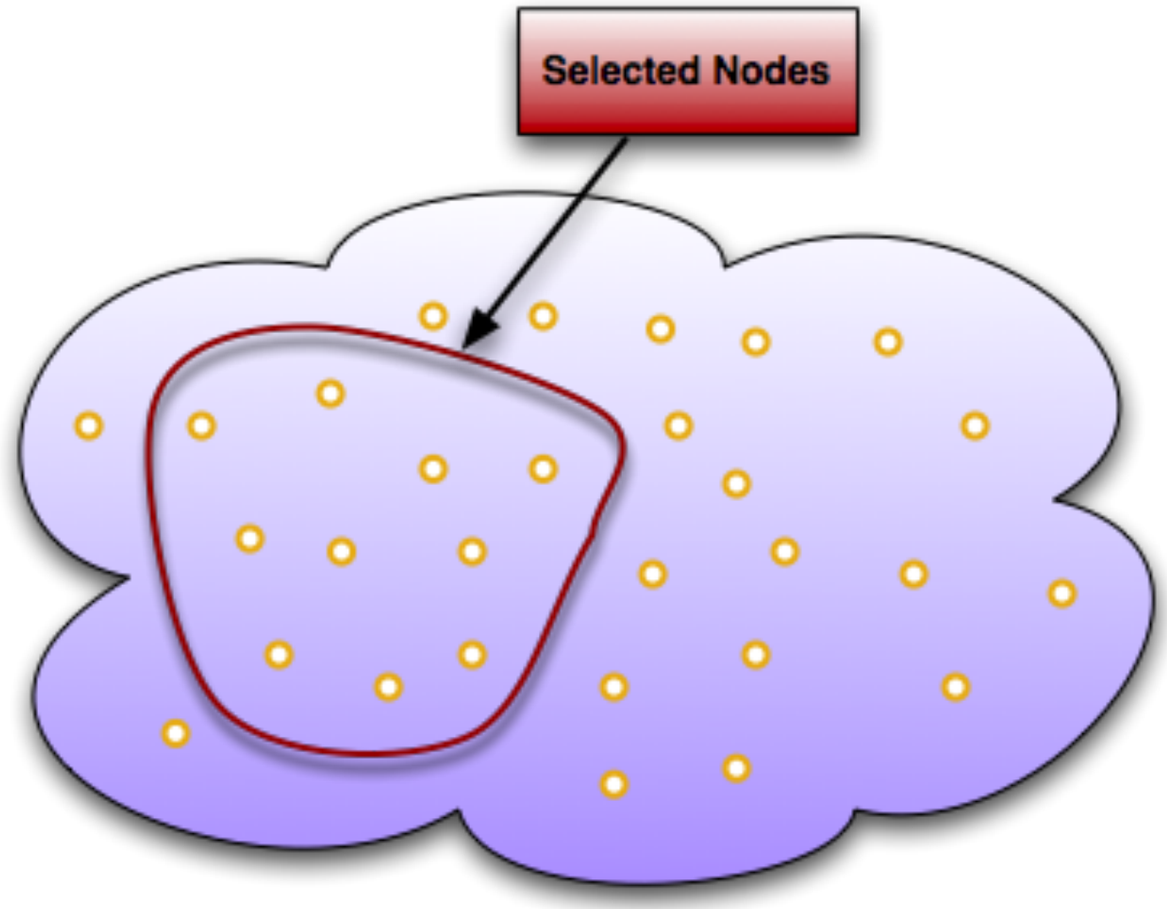
- A **clock synchronization** algorithm can work over a **Peer Sampling Service**
  - A node asks clock values to neighbors provided by the Peer Sampling Service
  - An Uniform Peer Sampling Service is usually a requirement in order to obtain a "good" quality of the clock synchronization



Peer Sampling - Clock Sync

In each algorithmic step a node performs a mean of *n* samples chosen uniformly at random from the entire population.

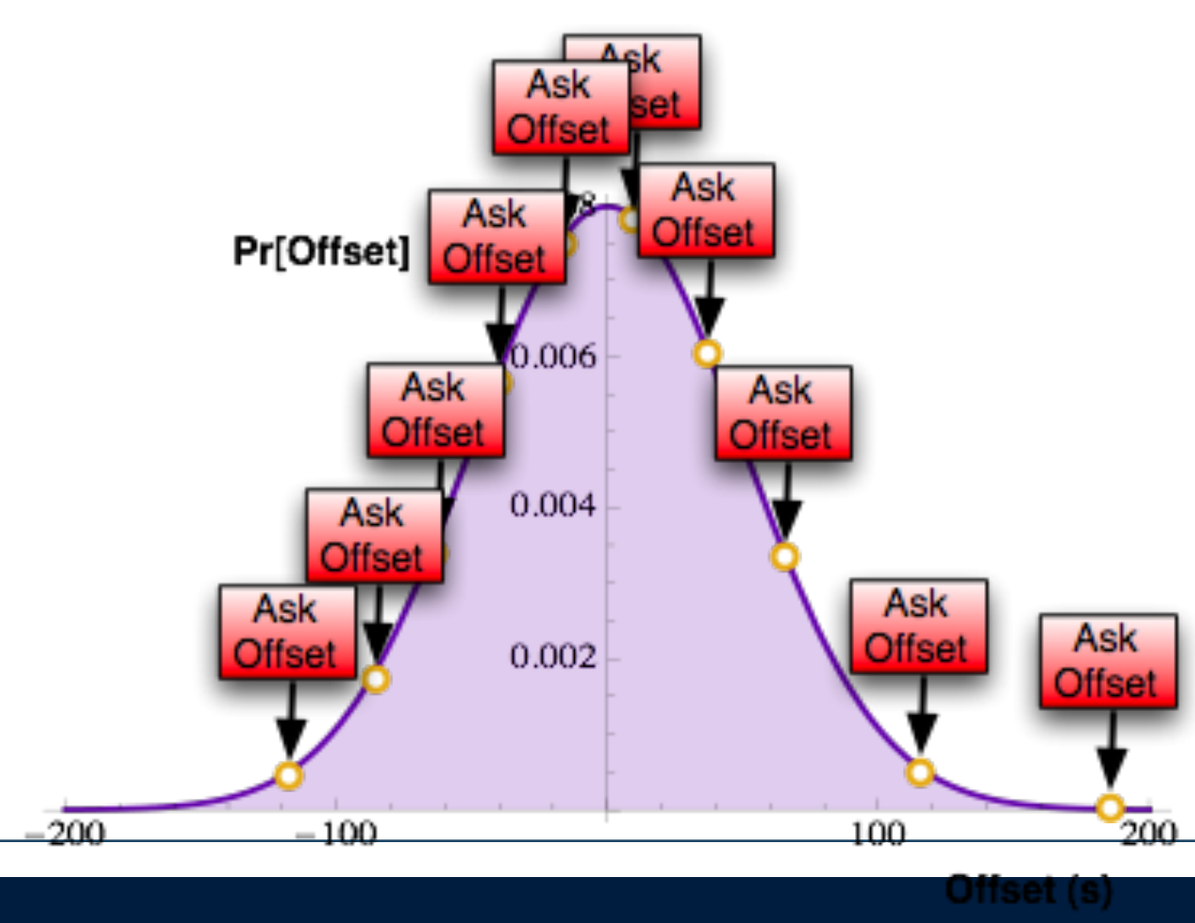
- The Peer Sampling Service selects *n* nodes from the whole system with which a node interacts
- Selected nodes are samples of distribution of clock values



Peer Sampling - Clock Sync

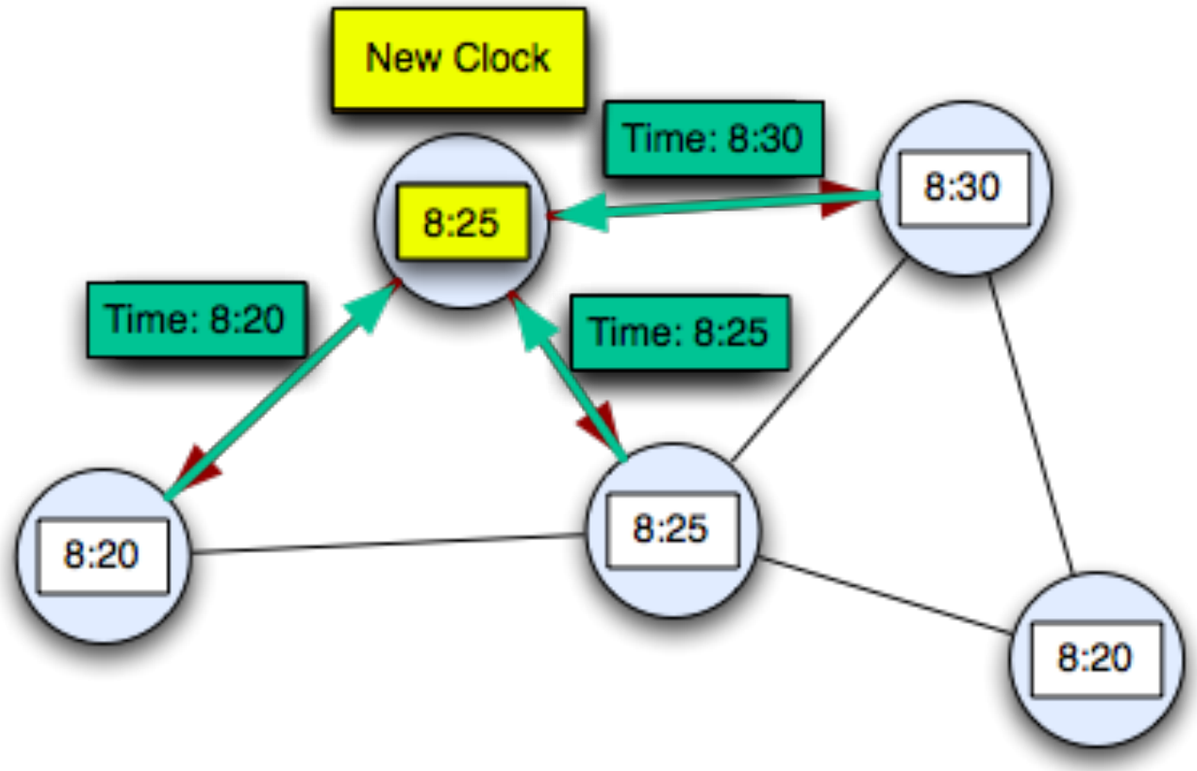
In each algorithm step a node performs a mean of *n* samples chosen uniformly at random from the entire population.

- The Peer Sampling Service selects *n* nodes from the whole system with which a node interacts
- Selected nodes are samples of distribution of clock values



Peer Sampling - Clock Sync

- The clock synchronization protocol is based on
  - **pull**, request-driven mechanism
  - a **gossip-based** approach
  - **random graph topology**



Agenda

- Introduction
- P2P as middleware
- The churn phenomena
- Gnutella: a famous example
- What is an overlay network
- Peer Sampling
- Structured Vs unstructured overlay network:
  - Cyclon: an unstructured P2P system
  - Chord/DHT: a structured overlay network
  - P2P in domotics environment
  - Kelips: an hybrid system
- Search in P2P network

Cyclon

S. Voulgaris, D. Gavidia, and M. van Steen, "CYCLON: Inexpensive Membership Management for Unstructured P2P Overlays", Journal of Network and Systems Management 13 (2005), no. 2

Cyclon is a distributed overlay management protocol:

- Builds and maintains a logical network connecting a set of nodes
- This network resembles a random graph
  - Strong connectivity
  - Small diameter

It is based on the "view exchange" mechanism (**shuffle**):

- Each node n_p maintains a small *view* V_p (set of neighbors) that is a subset of the entire system population.
- Periodically a node exchanges a part of its view with one of its neighbors

At runtime each local view can be considered as a uniform random sample of the entire system population

Cyclon: Shuffling

Shuffling at node n_p

- Select a random subset of neighbors L_p ($L_p \subseteq V_p$) of size $|L_p|$
- Select a random neighbour n_q from L_p
- Replace n_q's address with n_p's address in L_p
- Send L_p to n_q
- Receive L_q from n_q
- Update view to include L_q

At n_q on receipt of L_p

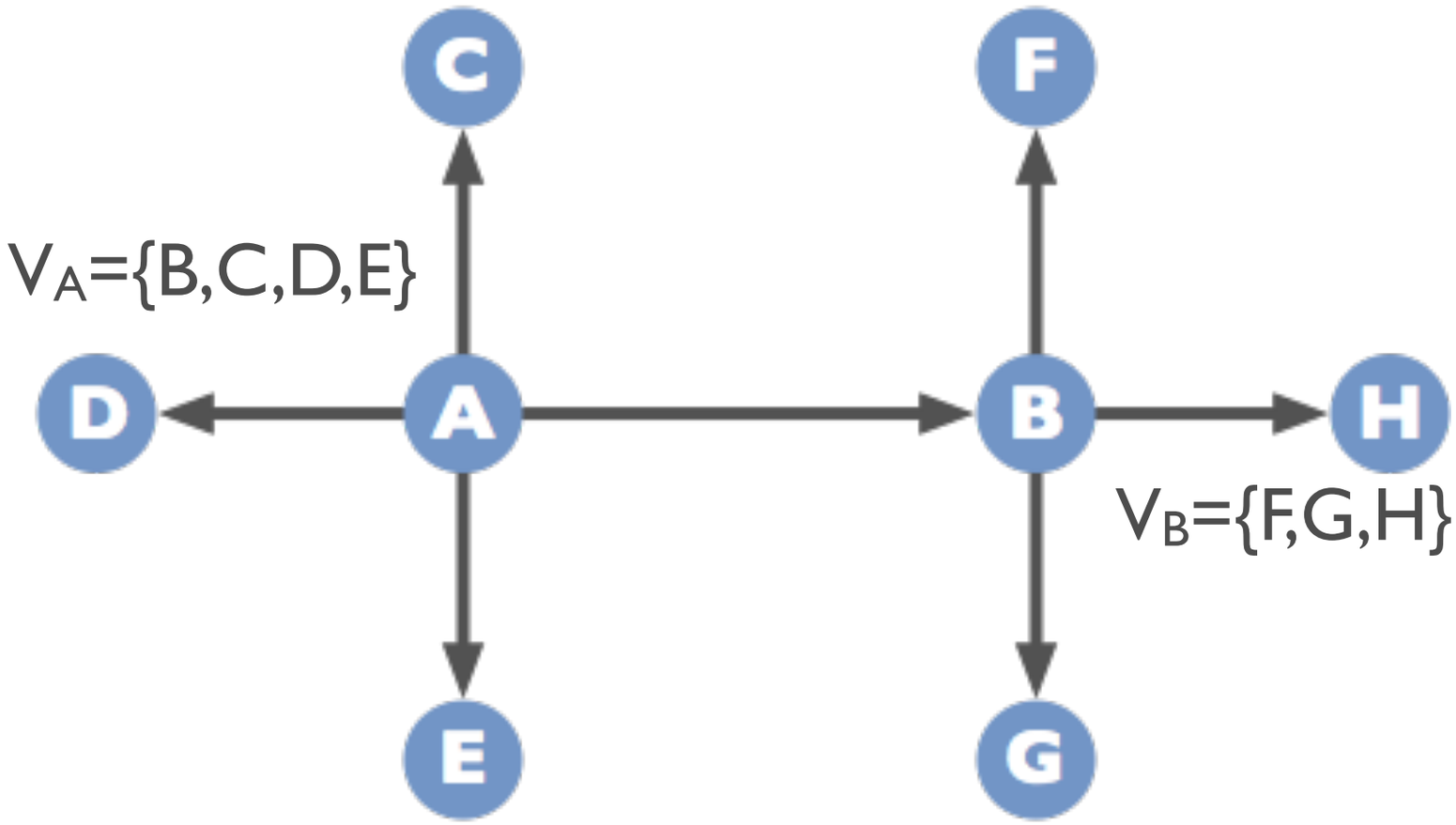
- Select a random subset from view
- Send L_q to n_p
- Update view to include L_p

Cyclon: Shuffling

- A peer is never its own neighbour
- At n_p on receiving L_q
  - Discard entries pointing to n_p in L_q
  - Fill any empty cache slots
  - Replace entries in L_q

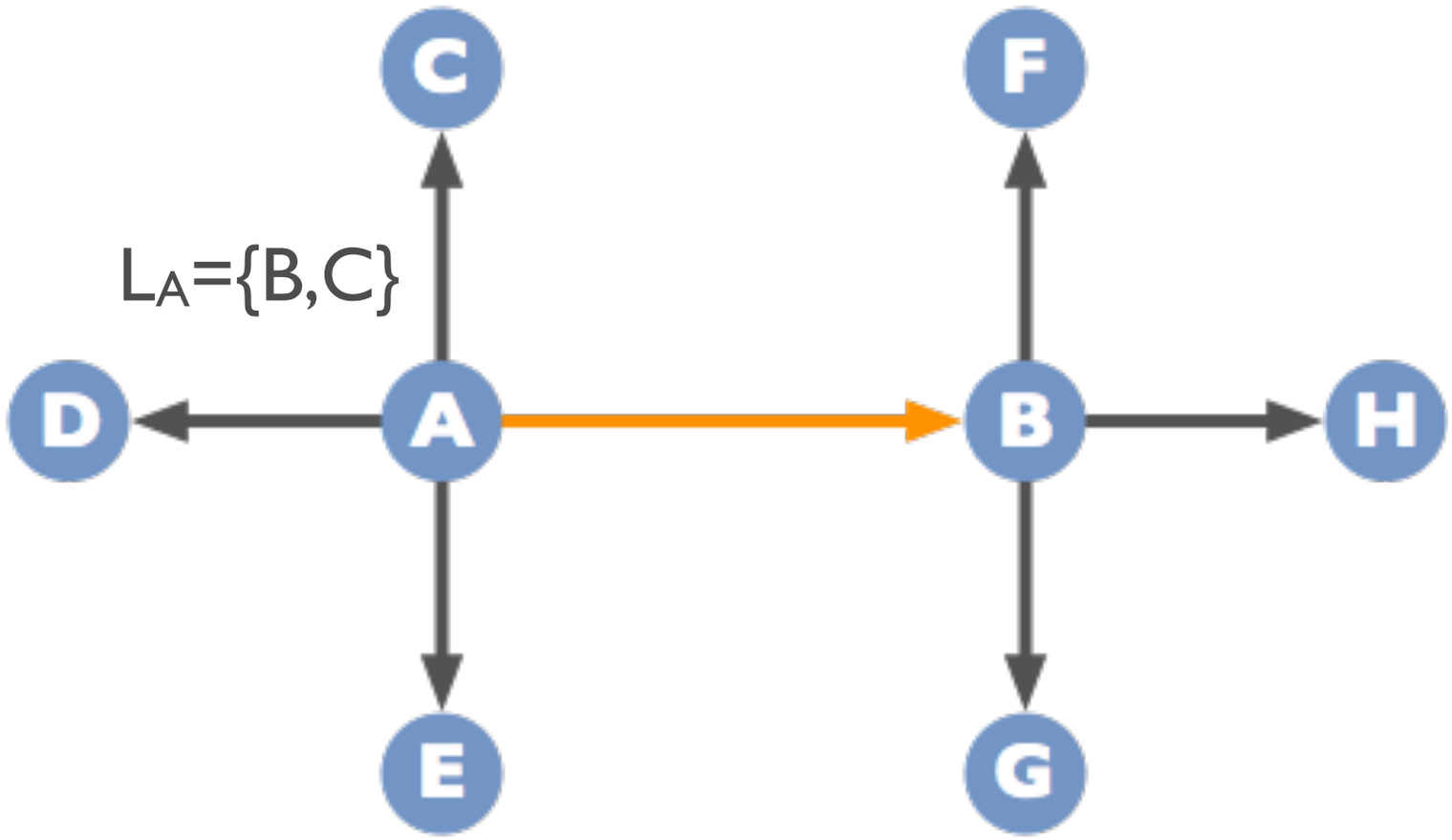
Cyclon

An example:



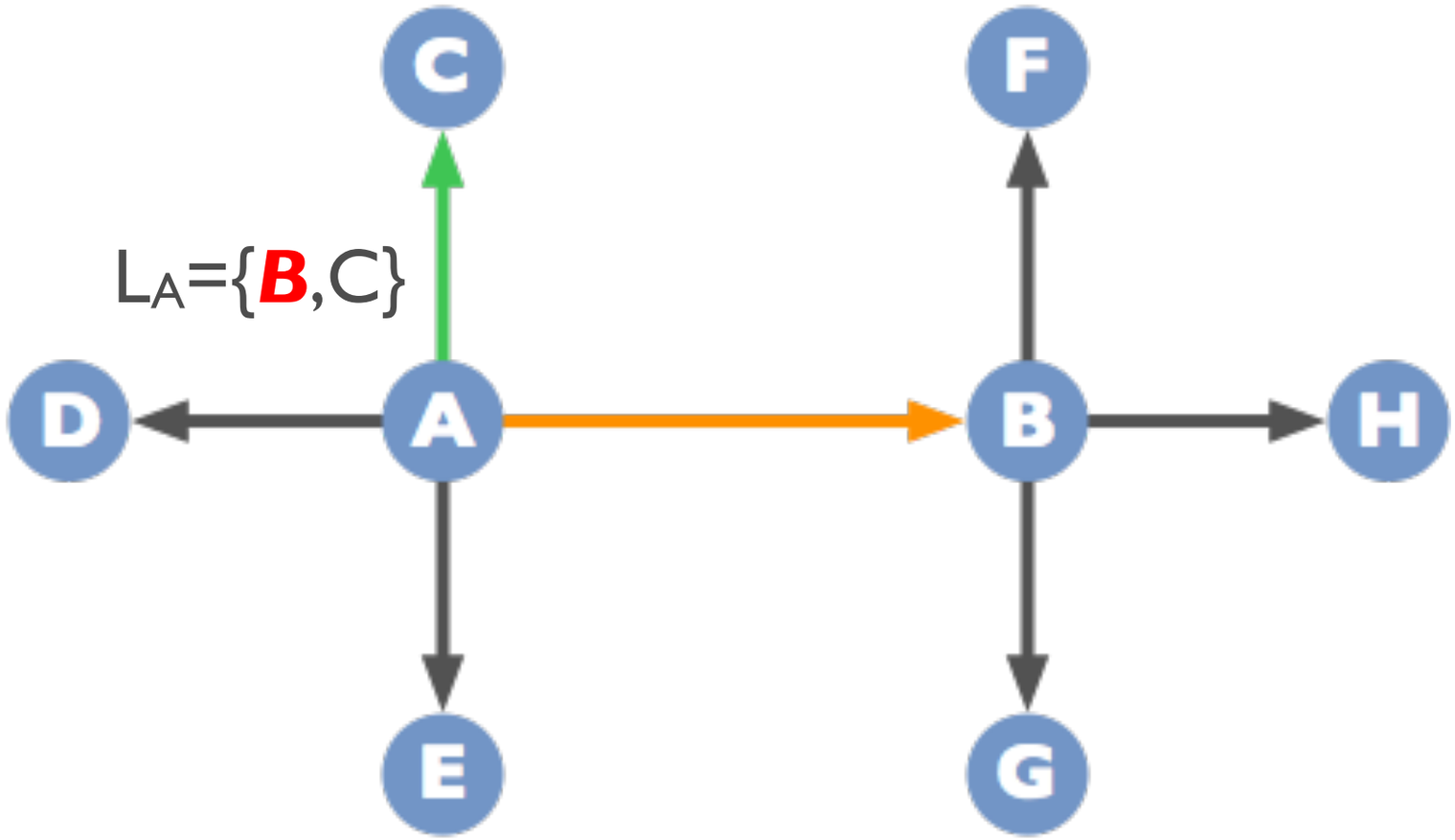
Cyclon

An example:



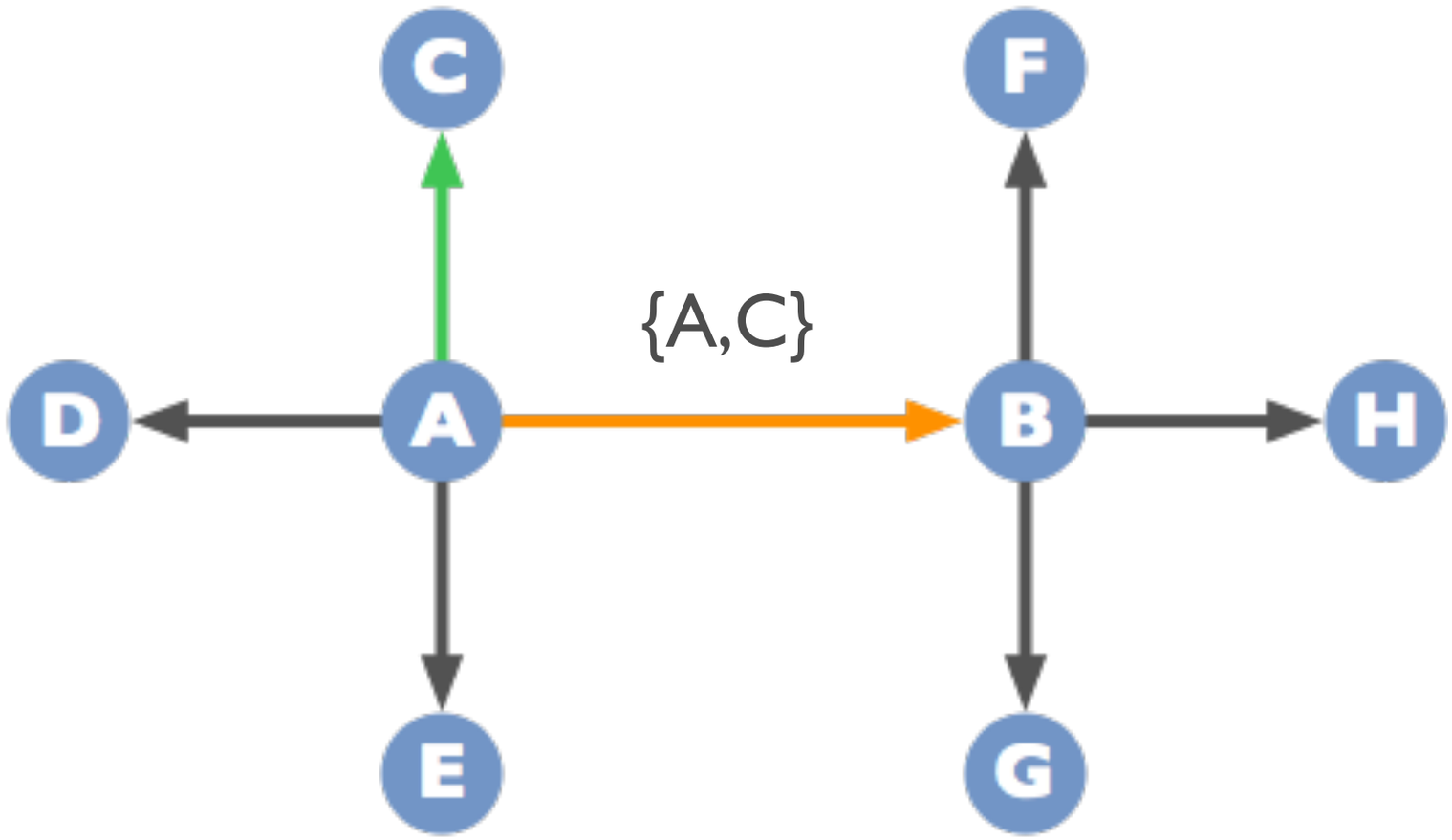
Cyclon

An example:



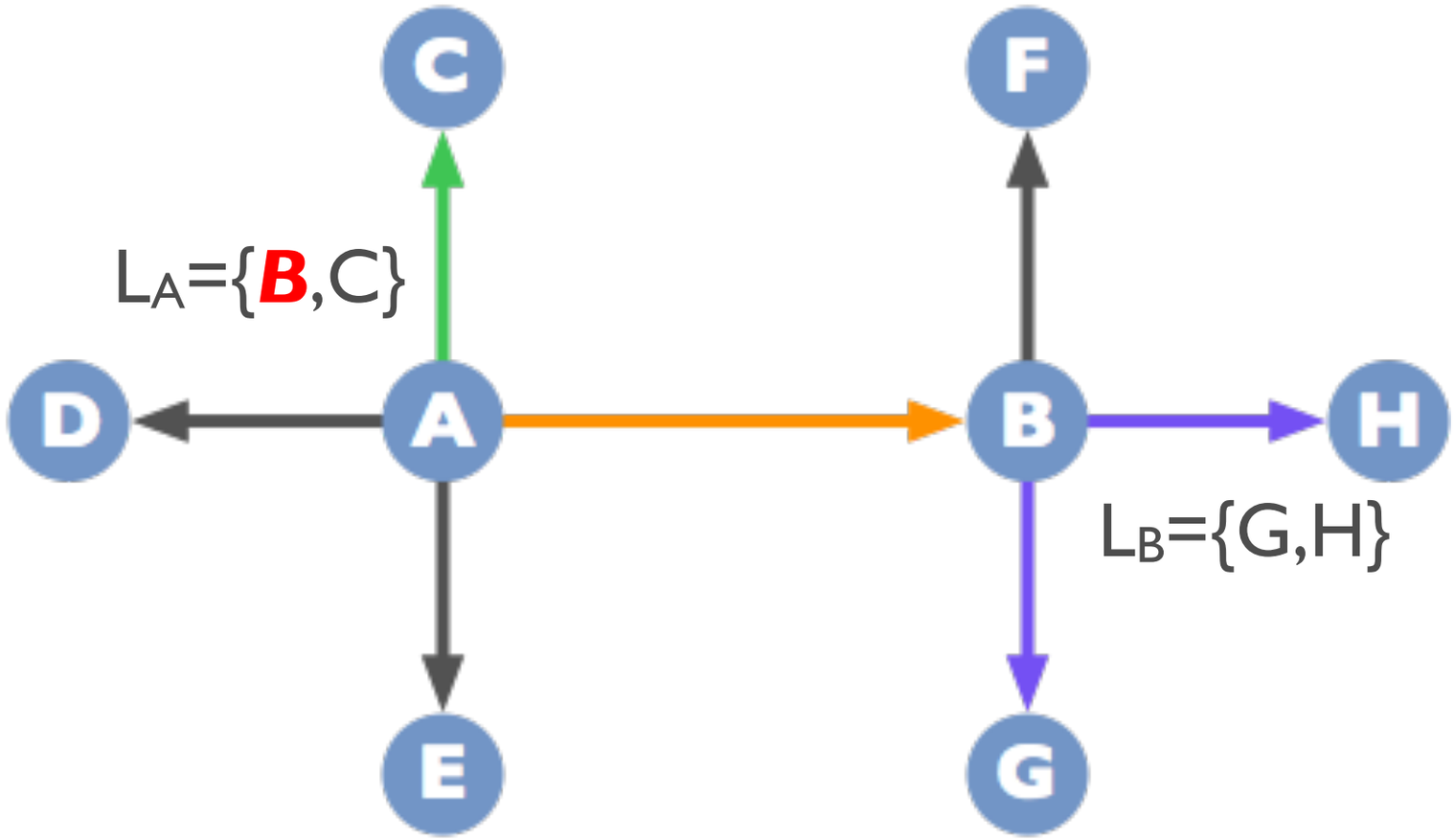
Cyclon

An example:



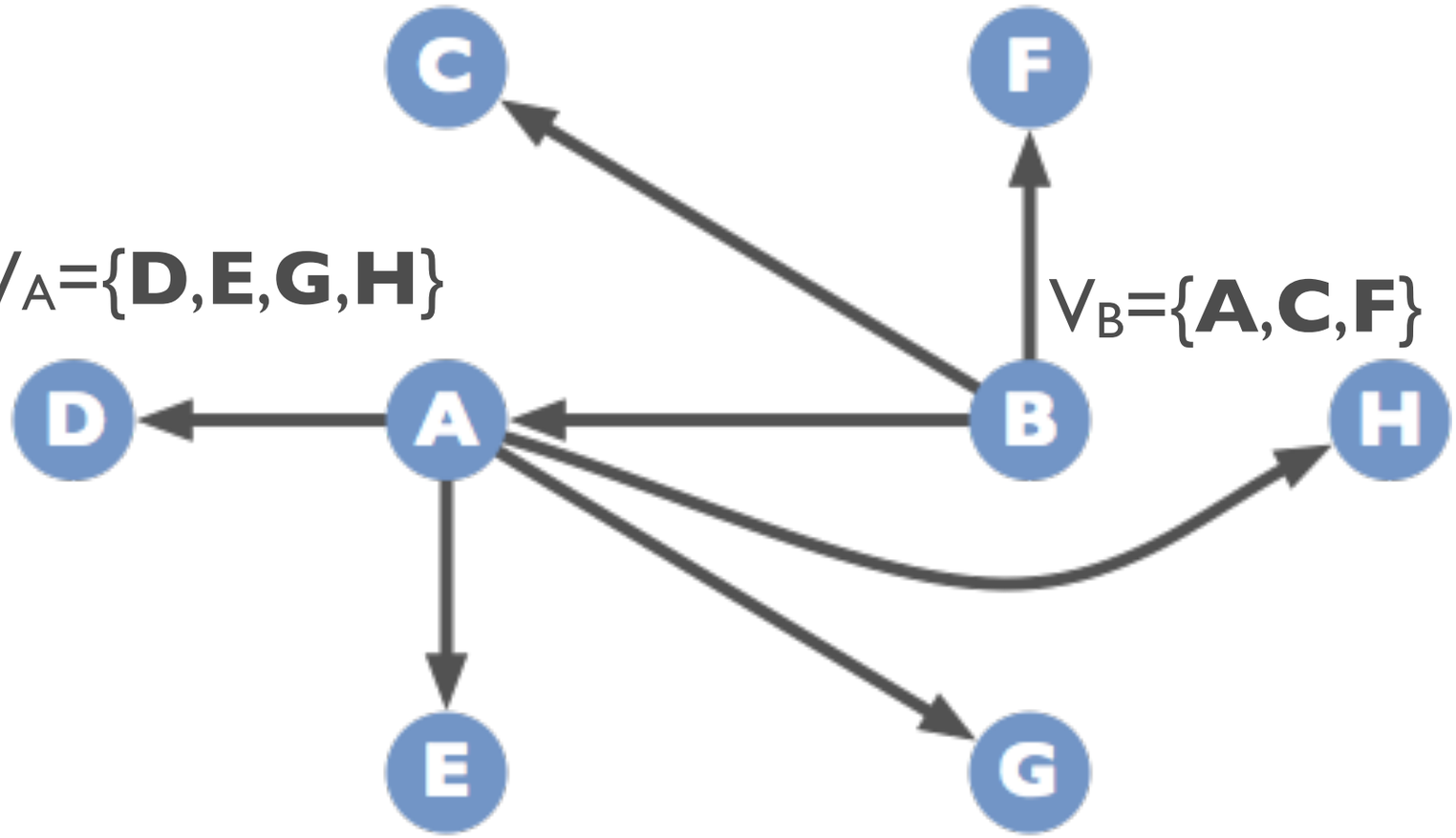
Cyclon

An example:



Cyclon

An example:



Cyclon: Shuffling

- No peer becomes disconnected
  - pointers move, so peers change from being neighbour of one peer to being the neighbour of another peer
- If n_p initiates a shuffle to n_q, then after the shuffle
  - n_p becomes a neighbour of n_q
  - n_q is no longer neighbour of n_p
  - Edges reverse direction

Cyclon: Node joins/leaves

- A joining node n_p
  - contacts an existing node n_q
  - performs c random walks of the expected average path length from n_q
- Property of random graphs
  - A random walk of length at lest equal to the average path length is guaranteed to end at a *random* node irrespectively of the starting node
- For each random walk, the target node n_r performs a shuffle of length l with n_p
  - n_r gets n_p in its view (age 0)
  - n_p gets the replaced entry of n_r
- The contacted node during a shuffle is removed if it does not respond
  - Such a node has higher age and therefore if it fails it will be selected and removed

Cyclon: Data Storage/Retrieval

- **store(x)**: a node *p* stores *i* copies of an information x on nodes chosen at random.
  - Node addresses are obtained though the peer sampling
- **get(x)**: each node forward randomly the request to x nodes chosen at random.
  - Node addresses are obtained though the peer sampling
  - A node containing the searched item returns it.
  - Search is stopped using a TTL value.

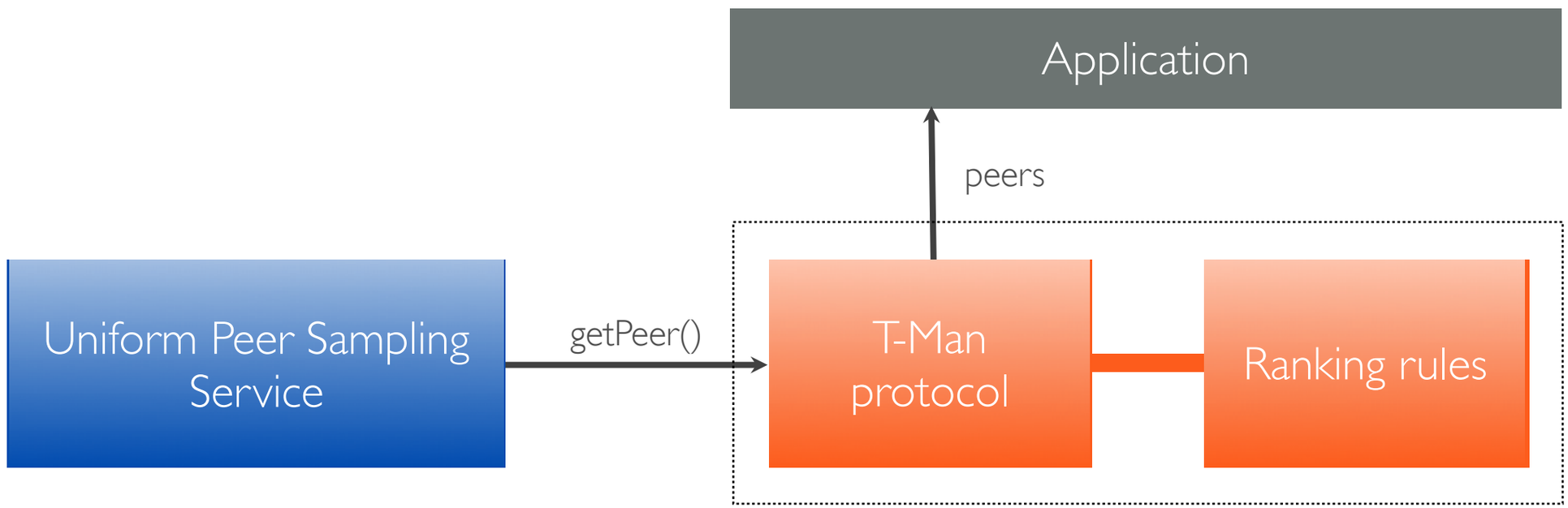
Cyclon - Peer Sampling Service

Cyclon provides a ***Peer Sampling Service***

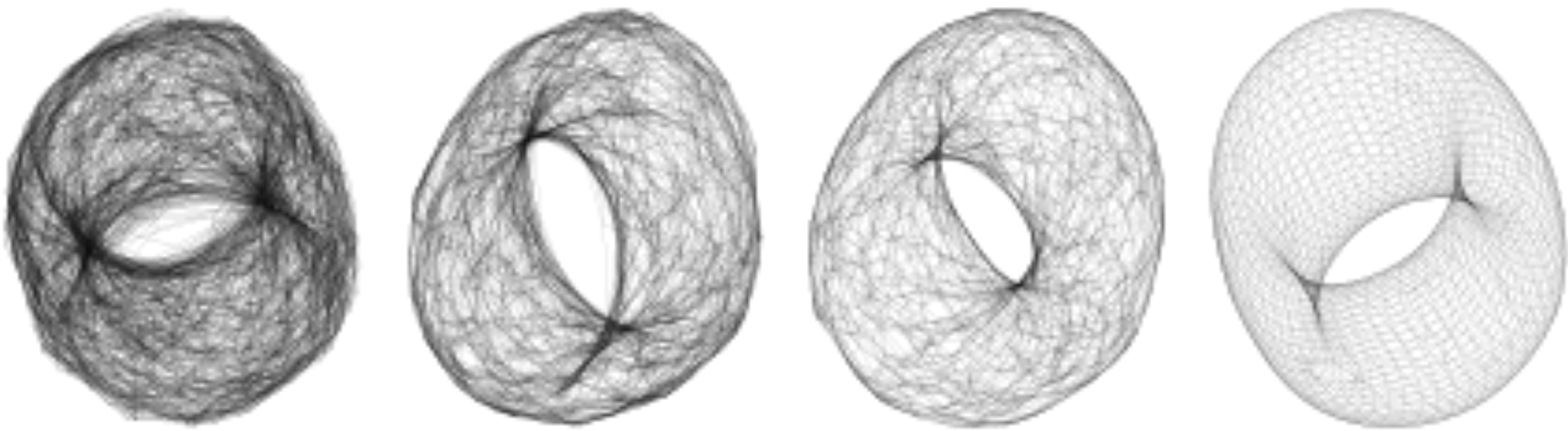
- This service can be used to obtain the address of one of the nodes belonging to the system.
- A relevant class of Peer Sampling Services are the ***Uniform*** Peer Sampling Services
  - Uniform Peer Sampling Service provides uniform random samples of nodes currently active in the system.
- The peer sampling service is a building block of several distributed algorithms

Cyclon - Peer Sampling - Structured overlay management

- T-Man: Gossip-based Overlay Topology Management [Jelasity, Babaoglu - ESOA 05]
- Overlay networks building and maintenance, i.e. how to manage application-level links in order to
  - keep the network connected
  - maintain a desired topology
- A single protocol for many topologies
- A ranking rule decides the topology



Cyclon: Structured overlay management



Time

Agenda

- Introduction
- P2P as middleware
- The churn phenomena
- Gnutella: a famous example
- What is an overlay network
- Peer Sampling
- Structured Vs unstructured overlay network:
  - Cyclon: an unstructured P2P system
  - Chord/DHT: a structured overlay network
  - P2P in domotics environment
  - Kelips: an hybrid system
- Search in P2P network

DHT: Distributed Hash Table

A *Hash table* is a data structure that collects values and that guarantee small reading/writing time.

Values are divided in *buckets* by mean of a function *h()* that associates each value with the relative bucket.

D*istributed hash tables* (DHT) extend this concept in a distributed scenario:

- allow to store pairs *(key, value)*.
- each node stores only a subset of the whole set of keys (its bucket).
- In order to know the node storing a value, it is sufficient to compute *h(key)* where (key, value) is a pair of the DHT.
- They must manage efficiently the addition/removal of nodes

DHT: Distributed Hash Table

A DHT provides two primitives:

put(k,v) : stores the value v into the node responsible for the key k

get(k) : requests to the node, responsible for key k, the value v corresponding to k

Nodes, keys and values are managed such as strings of bits:

NodeID : string of *n* bits univocally identifying a node in the system.

key : string of *n* bit univocally identifying a value in the system

value : string of bytes without a predefined length

DHT: Distributed Hash Table

There are a lot of service that can be implemented using a DHT:

- File sharing
- Storage
- Database
- Name Directory
- Chat
- Publish/subscribe Systems
- Distributed Cache
- Streaming audio/video
- ...

DHT: Distributed Hash Table

... but what kind of characteristics must have a DHT?

- Keys should be uniformly distributed on every node in the system (load balancing).
- Each node must maintain only informations about a (small) subset of the nodes in the system.
- Each request should be efficiently forwarded among nodes.
- Adding or removing a node should involve only a small number of operations.

DHT: Distributed Hash Table

Several implementations of DHT:

- Chord [MIT]
- Pastry [Microsoft Research UK, Rice University]
- Tapestry [UC Berkeley]
- Content Addressable Network (CAN) [UC Berkeley]
- SkipNet [Microsoft Research US, University of Washington]
- Kademlia [New York University]
- Viceroy [Israele, UC Berkeley]
- P-Grid [EPFL Ginevra]
- ...

Chord

Node ID: unique identifier of a node composed by N bits (usually obtained applying hashing functions on IP address of the node)

Key: identifier of N bits obtained computing hashing function on a sequence of bytes

Value: a sequence of bytes

Operations:

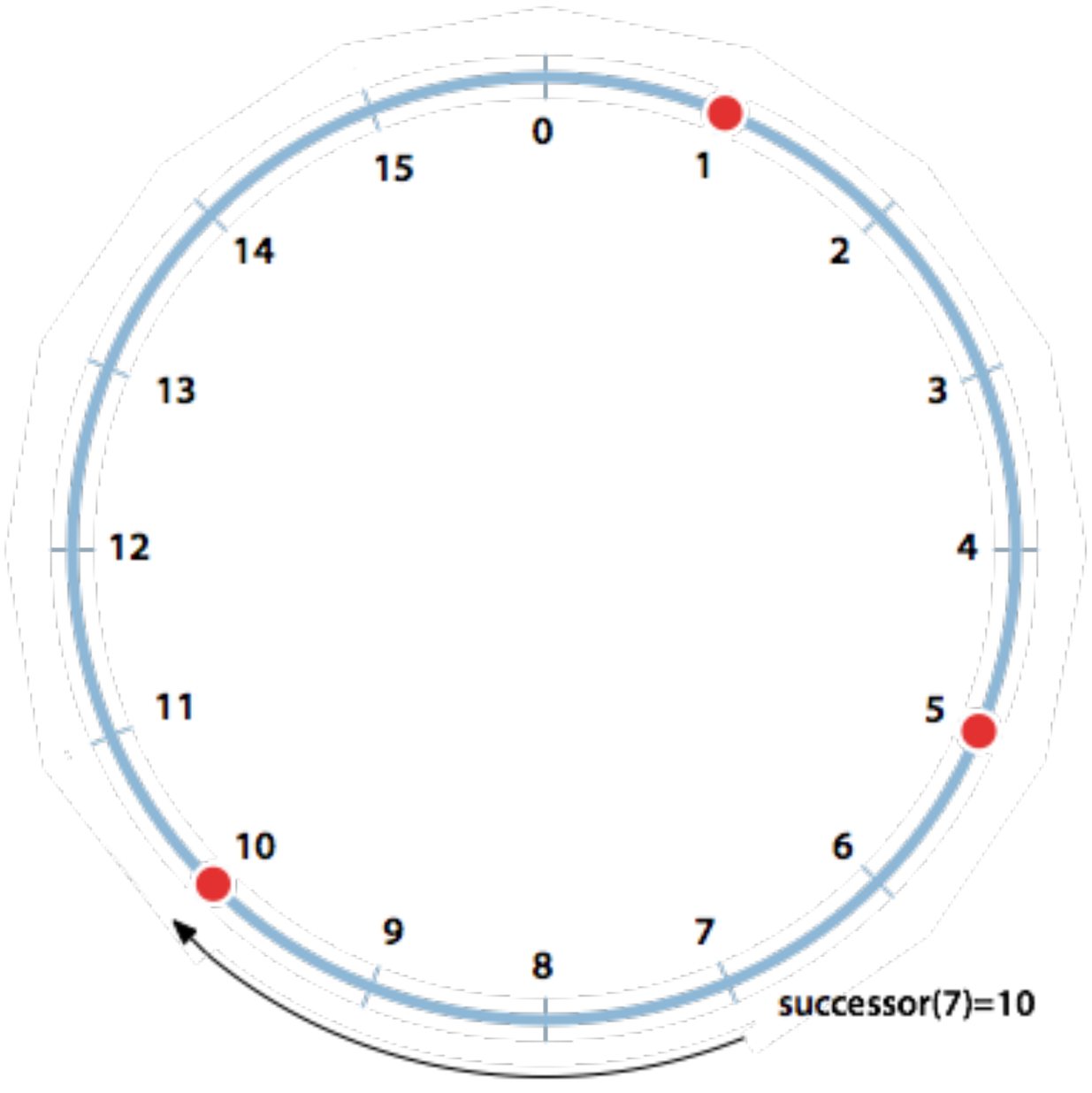
- insert(key, value)
- lookup(key)
- update(key, new_value)
- join(n)
- leave( )

Chord

Nodes are organized inside a "logical ring" composed by keys of N bits.

Each node is responsible for every key that comes before it in the ring (it is defined the function *successor(k)*).

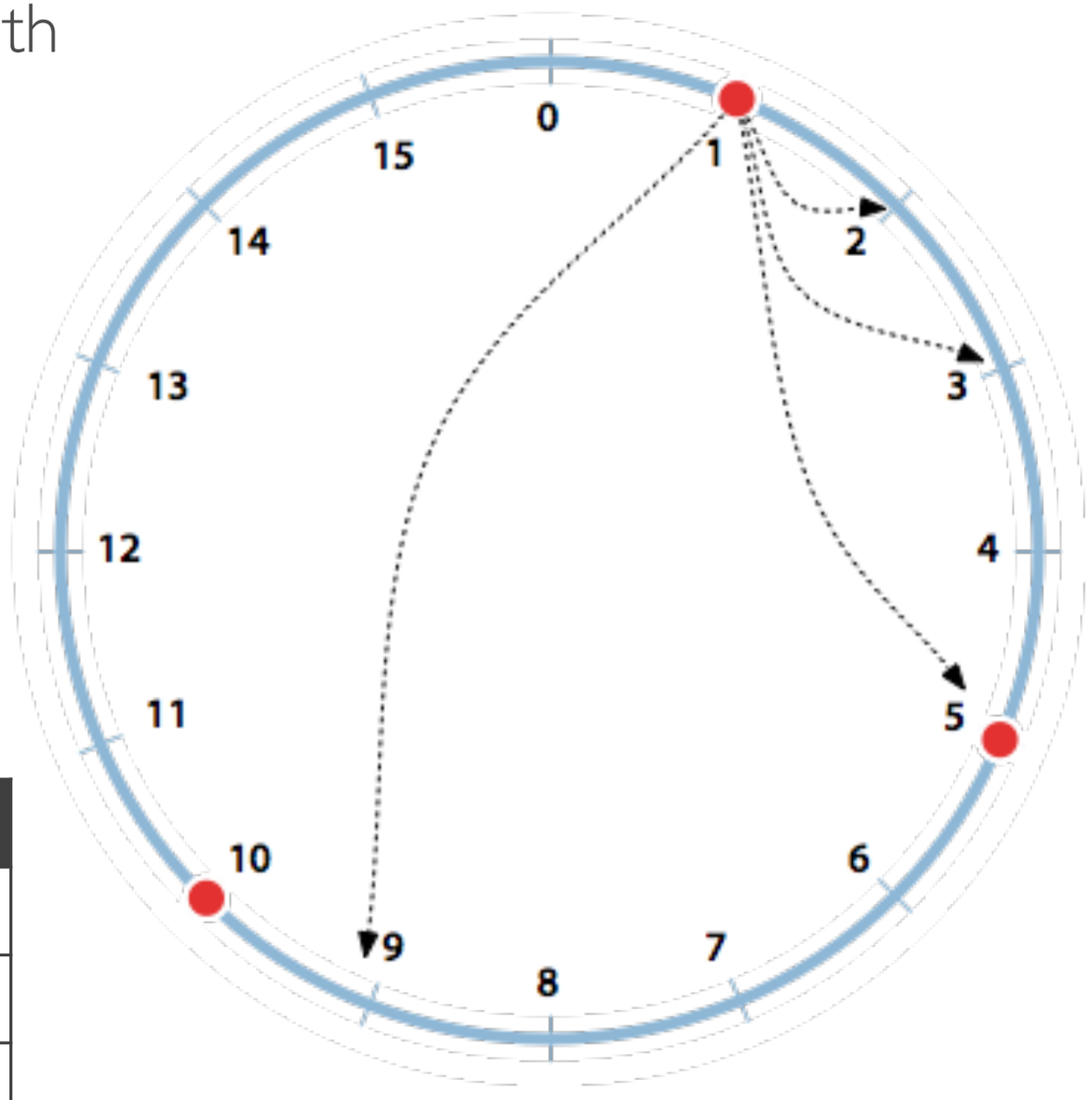
The hash function *h()* guarantees an uniform distribution of the keys and nodes in the ring.



Chord

Each node maintains a routing table (***finger table***) with O(log N) rows.

i-th row in the table belonging to node identified by *X*, contains node returned by *successor(X + 2^(i-1))*



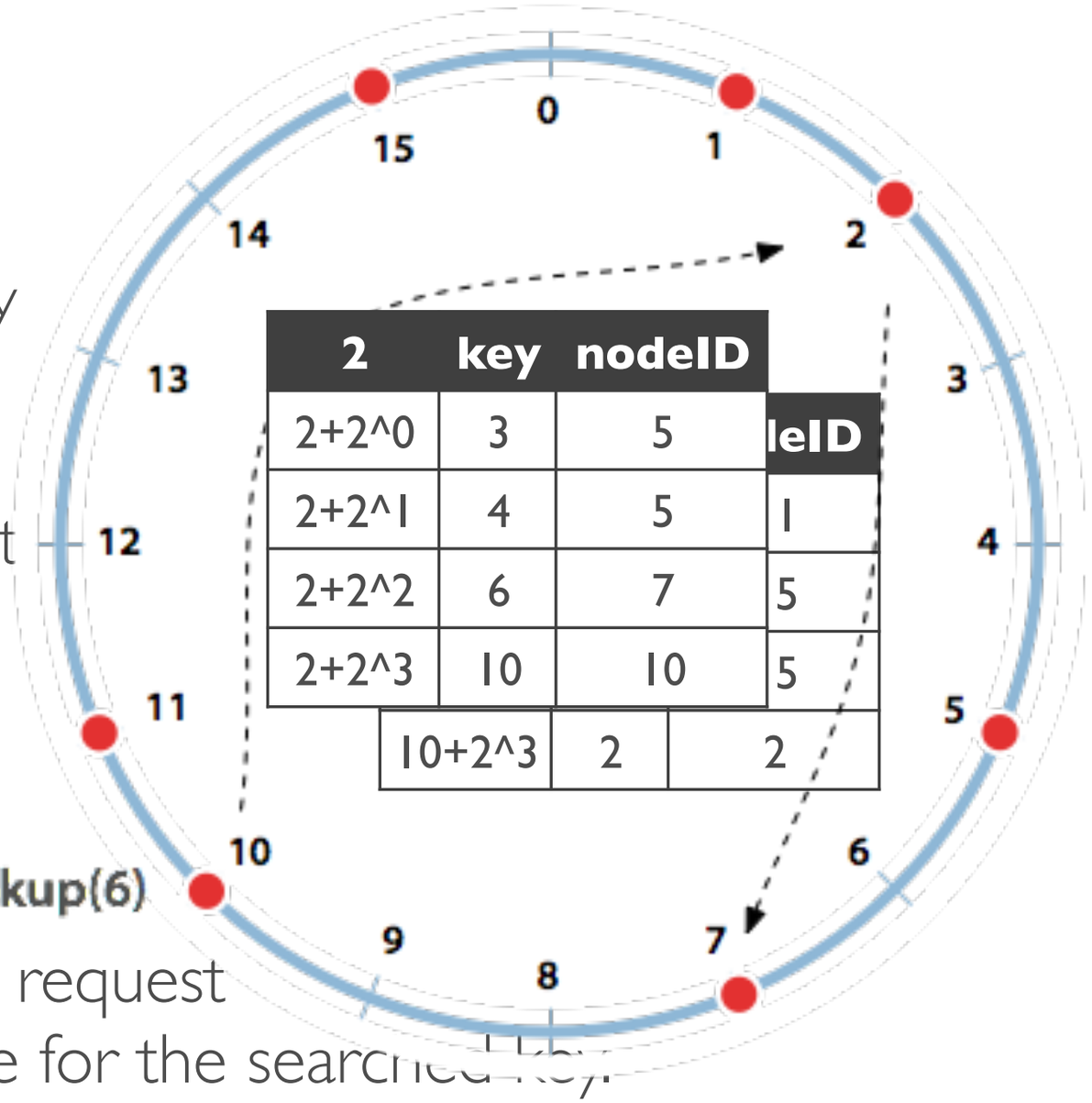
I	key	nodeID
I+2^0	2	5
I+2^I	3	5
I+2^2	5	5
I+2^3	9	I0

Chord

Searching a key *k*=6...
(***lookup***)

Each node selects
by mean of its finger table
the farthest finger whose key
comes before (or it is
equal) the requested
key and forwards the request
corresponding node

In this way
There is the certainty that,
in a finite number of steps,
the key lookup ends and the request
reaches the node responsible for the searched key.



Chord

Management of join/leave

Two properties must be maintained:

- Finger tables have to be correct
- Each key k has to be managed by the node successor(k)

Join(n):

A. Predecessor and fingers of n are initialized
B. Predecessors and fingers of other nodes in the network will be be updated to take into account
C. All keys, whose successor is become **n**, will be moved in **n**

A. Initialization of the predecessor and fingers:

Computing the finger table is trivial:

For each row in the finger table the node asks to each other node in the network to compute the successor(n+2^(i-1)).

In order to compute the predecessor:

1. n asks to a node to compute n'=successor(n)
2. n asks to n' who is its predecessor
3. the predecessor of n' becomes the predecessor of n

B. Update of predecessors and fingers of other nodes in the network:

Hyp. After the join of n, n should become the i-th finger of a node p.

This can happen if and only if:

- p precedes n of at least 2^(i-1) keys
- the node m identified by the i-th finger of p satisfies m=successor(n)

The first node that is able to satisfy these conditions is the immediate predecessor of (n-2^(i-1)).

For each finger i, the algorithm covers the ring counterclockwise starting from the key n-2^(i-1) updating the i-th finger of each node p' for which n=successor(p'+2^(i-1)).

C. All key whose n is become the successor will be moved in n:

This operation can be completed with a simple inspection of the keys managed by successor(n).

The algorithm used by nodes to leave the network is similar.

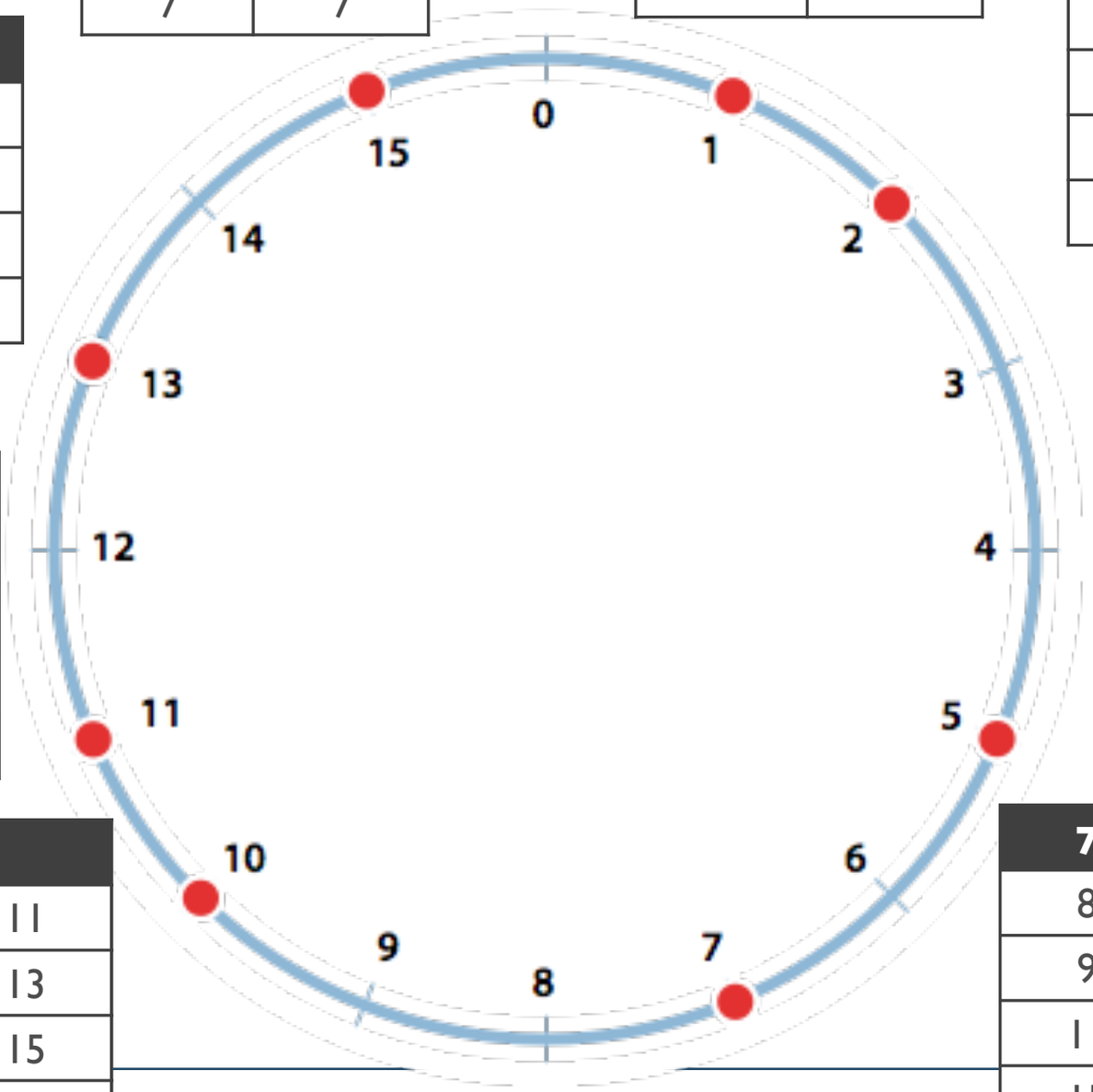
Chord

15	
0	1
1	1
3	5
7	7

1	
2	2
3	5
5	5
9	10

13	
14	15
15	15
1	1
5	5

2	
3	5
4	5
6	7
10	10



11	
12	13
13	13
15	15
3	5

5	
6	7
7	7
9	10
13	13

10	
11	11
12	13
14	15
2	2

7	
8	10
9	10
11	11
15	15

Chord

In a system with N nodes and K keys:

- There is high probability that a node will be responsible of K/N keys.
- Each node maintains informations about O(log N) nodes
- There is high probability that a search is completed in O(log N) steps.
- When a node executes a join/leave only O(1/N) keys must be moved and, to complete the operation, will be produced O(log^2 N) messages.

Agenda

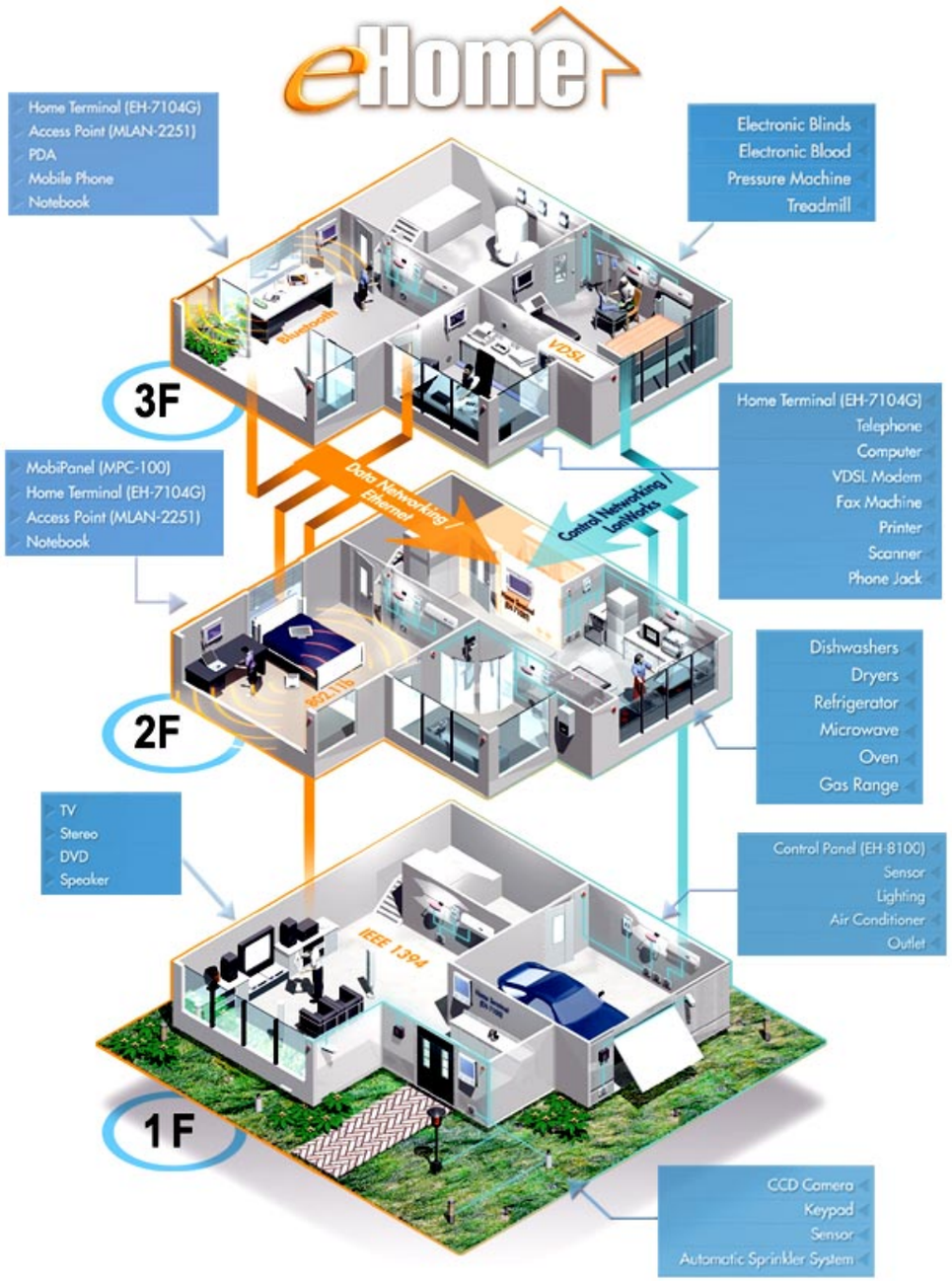
- Introduction
- P2P as middleware
- The churn phenomena
- Gnutella: a famous example
- What is an overlay network
- Peer Sampling
- Structured Vs unstructured overlay network:
  - Cyclon: an unstructured P2P system
  - Chord/DHT: a structured overlay network
  - P2P in domotics environment
  - Kelips: an hybrid system
- Search in P2P network

P2P in domotics environment

Example of complex applications: storing context data in a distributed home automation system.

Future home automation systems:

- Lot of devices
- Powerful devices
- Complete decentralization
- Complex composite services



P2P in domotics environment

Context: current state of the environment where the system is running.

Physical context

- room temperature
- light intensity
- Water flow from the tap

Device context

- the light bulb is on
- AC fan is running at medium speed

User context

- Alice is in the kitchen close to the fridge
- Bob prefers to answer incoming phone calls from his mobile

System context

- A software component is running correctly or not

P2P in domotics environment

The availability of up-to-date context is paramount to:

- offer automatically orchestrated complex services
- offer these services in a personalized way taking into account user preferences
- adapt the service execution to the current environment status
- tolerate and adapt to malfunctions

Managing context means:

- Data collection
- Context storage
- Context search and retrieval

P2P in domotics environment

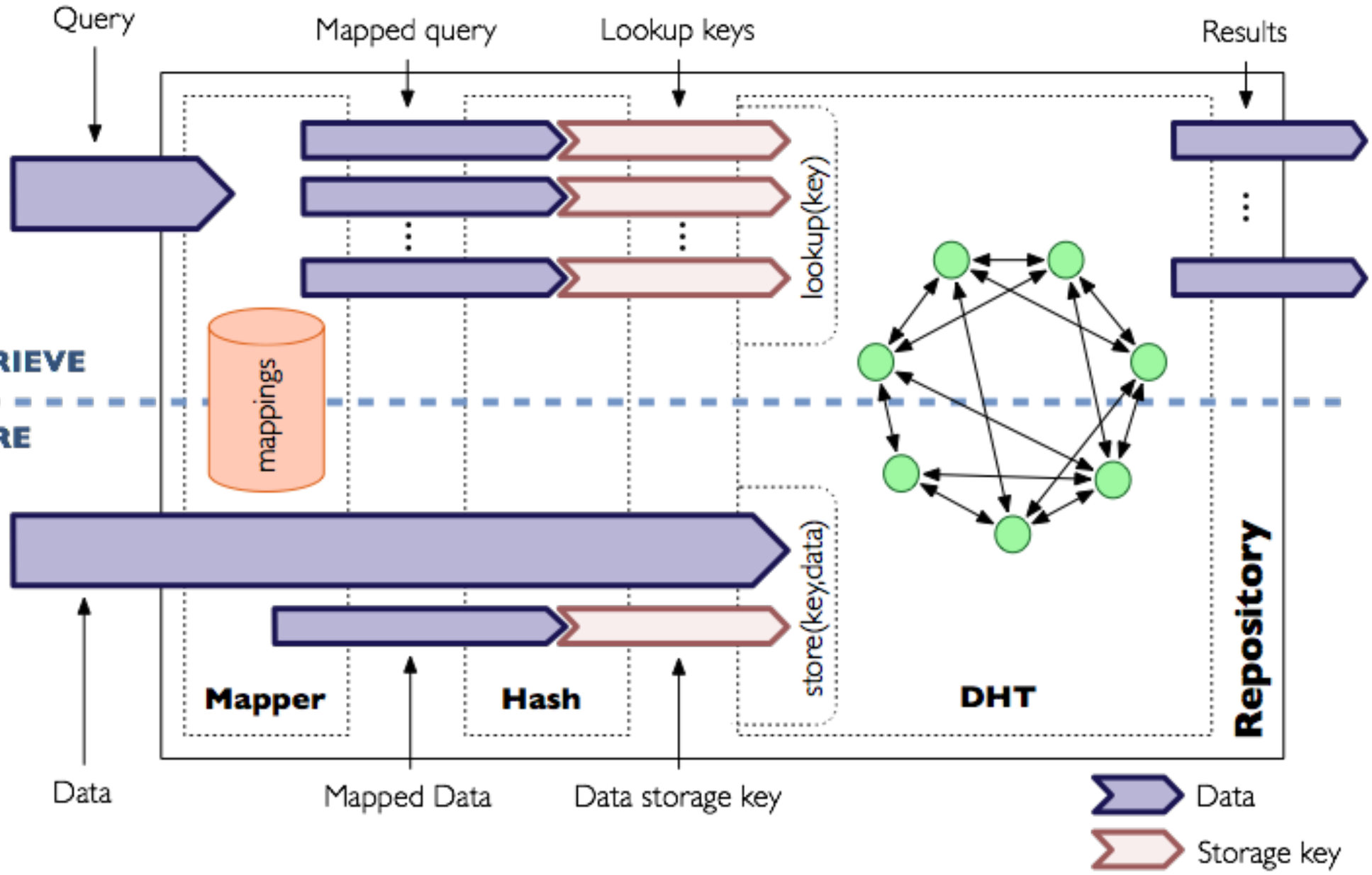
The DHT only offers a simple lookup(key)→data primitive

How can we express complex queries like

> *"Retrieve all data from devices that sensed a temperature greater than 21°C and that are located in the kitchen"*

We introduce a "mapper" component that decouples the interaction among devices/software components and the DHT

P2P in domotics environment



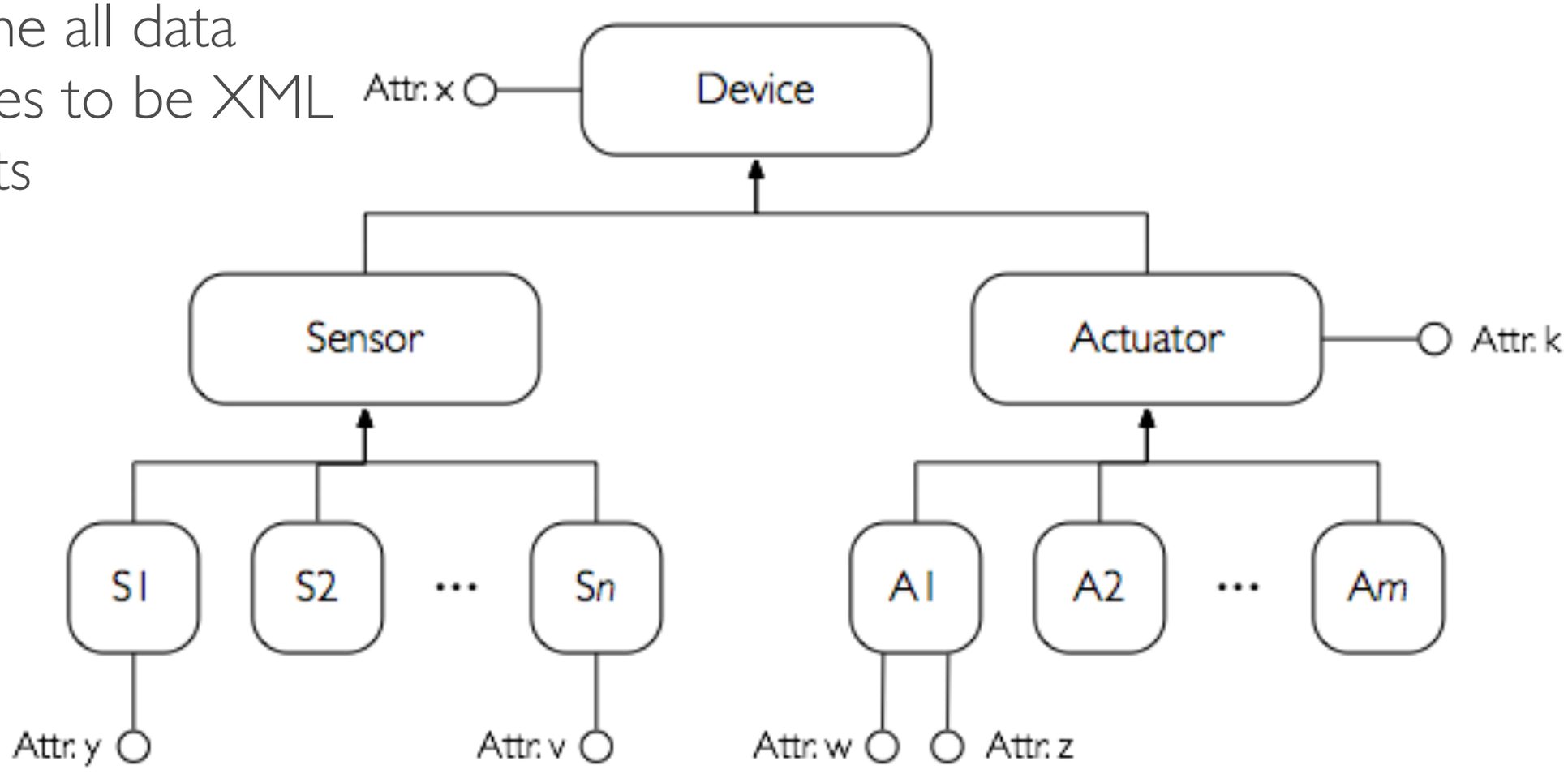
P2P in domotics environment

In order to map queries and data to keys in a meaningful way both must adhere to a common schema

The schema can be represented through a hierarchical structure

Elements are characterized by attributes

We assume all data and queries to be XML documents



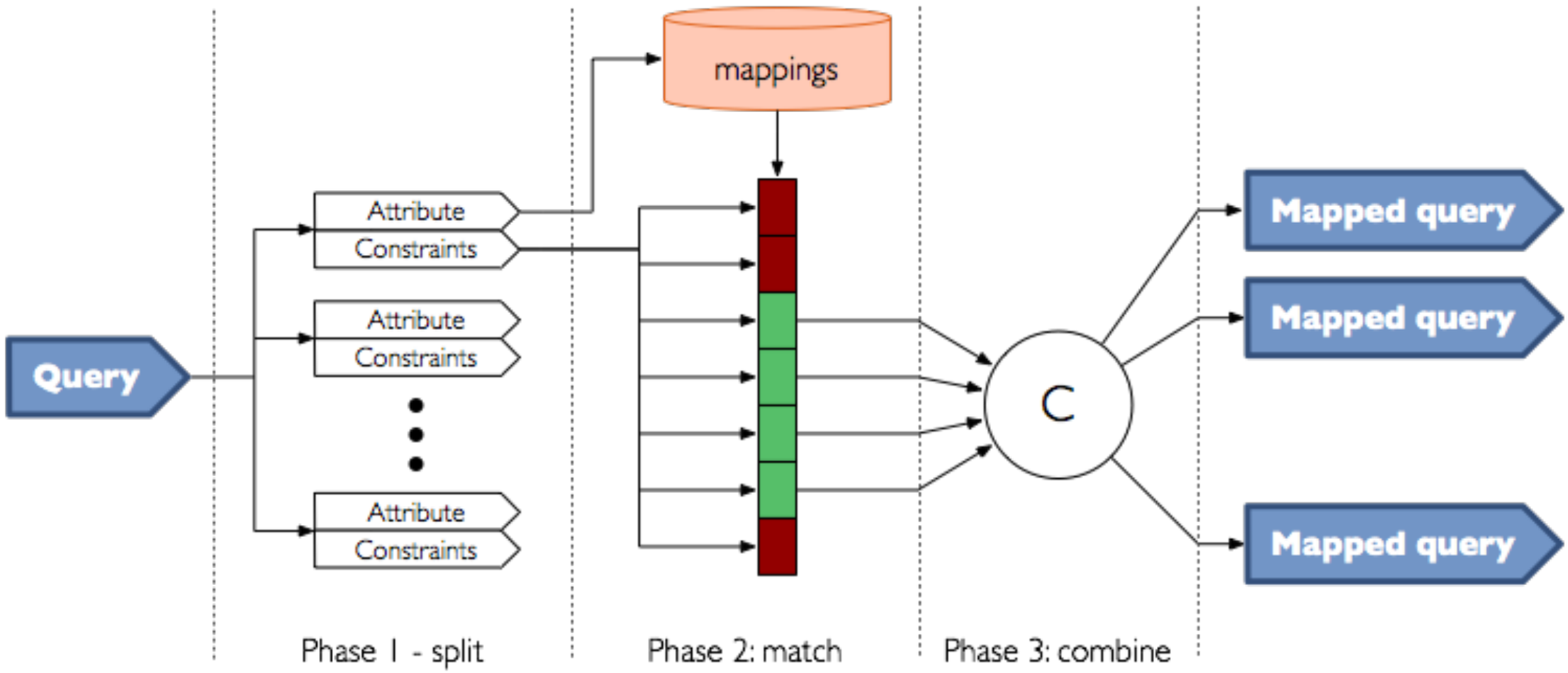
P2P in domotics environment

Given a shared schema:

- A piece of context data produced by a device is a set of attributes with attached values
- A query is similar but instead of values it can contain constraints on one or more attributes

P2P in domotics environment

When a query is issued to the repository it goes through three phases in the mapper



P2P in domotics environment

Mappings must be defined for every attribute associated to every element of the schema.

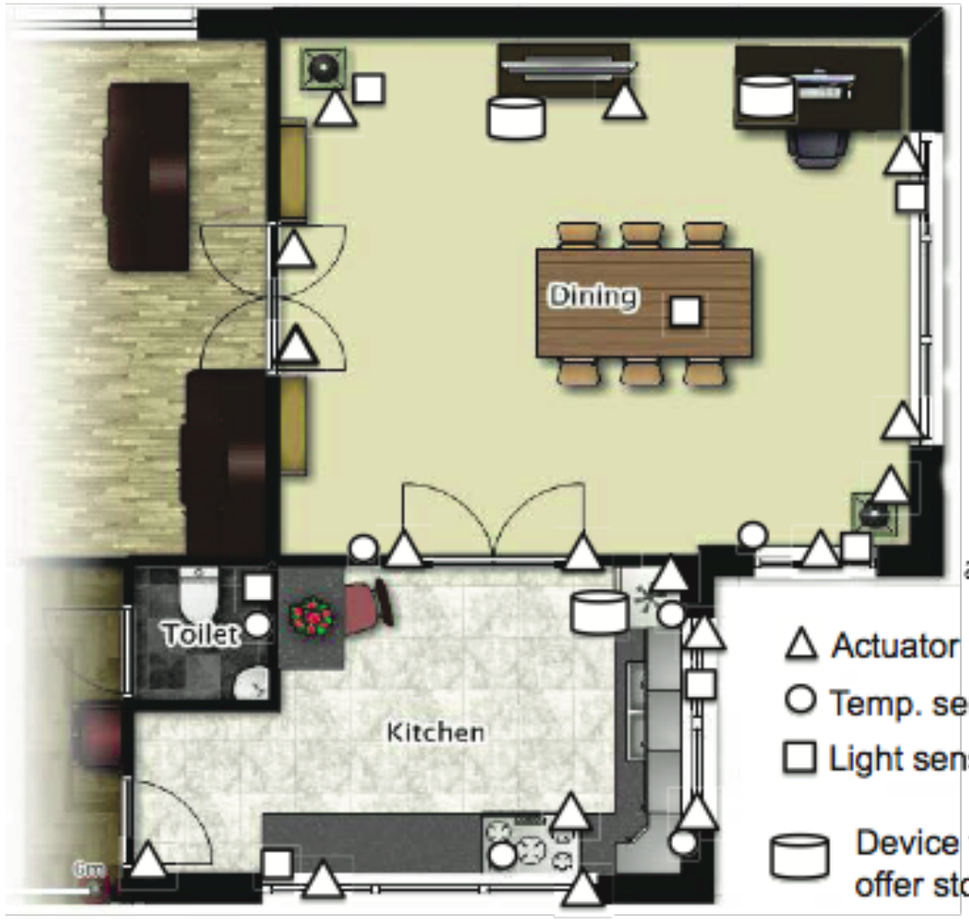
A mapping is built by partitioning the set of all valid values for an attribute in subsets and electing a representative value for each of them

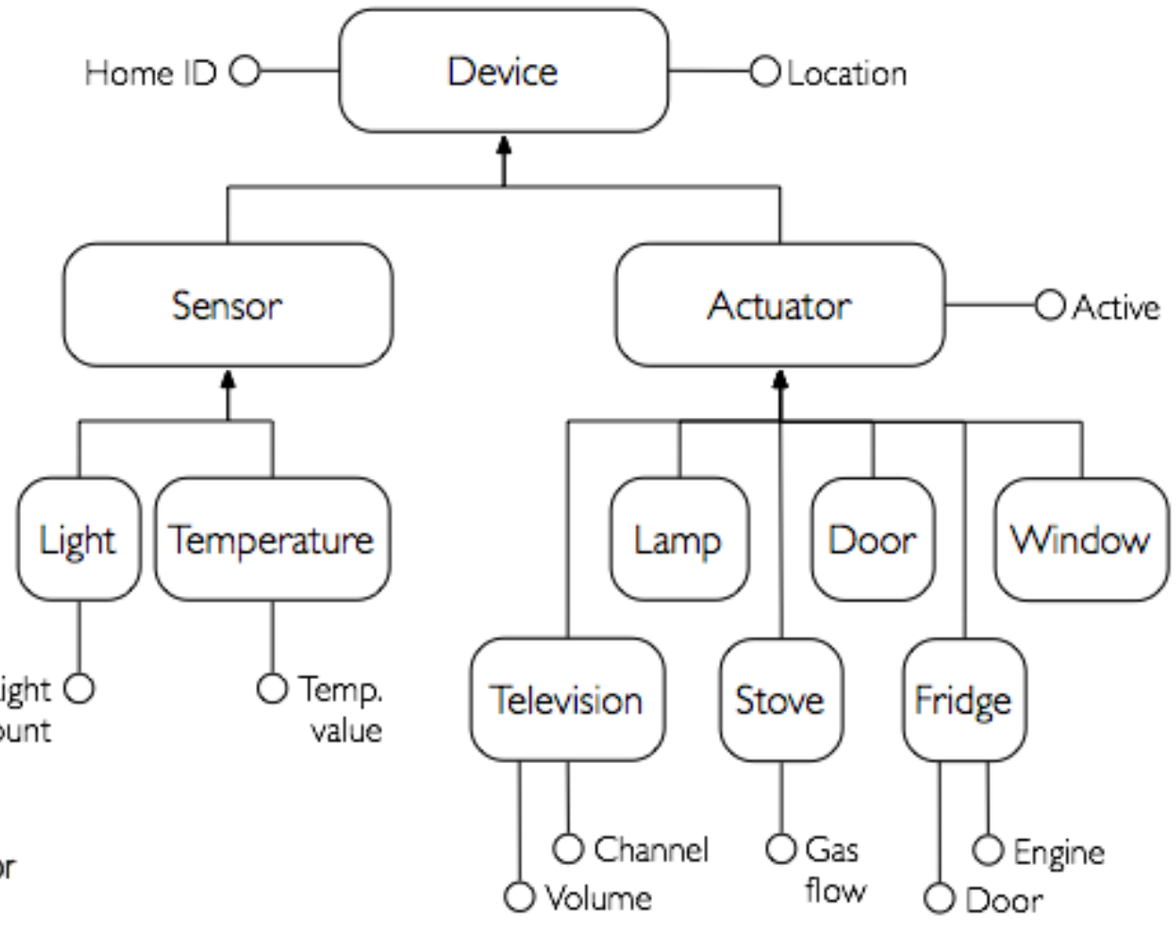
Mapping is straightforward for bounded numerical values:

/Device/Sensor/Temperature/current_temperature ⇒ [–10,0,10,20,30,40,50]

P2P in domotics environment

A practical example: consider some rooms in a house equipped with smart devices and our system





P2P in domotics environment

Mappings must be defined for temperature and light sensor readings:

- Temperature values are numbers ranging between 10 and 40 (expressed as Celsius deg.)

  /Device/Sensor/Temperature/Temp value ⇒ [10, 15, 20, 25, 30, 35]

- Light intensity values are expressed like four broad categories (dark, soft, normal and strong light)

  /Device/Sensor/Light/Light amount ⇒ [0, 1000, 5000, 20000]

- An enumeration attribute like Location has a predefined set of acceptable values ➠ 1-to-1 mapping is fine

P2P in domotics environment

A temperature sensor could produce a piece of data like this:

```
<Device>
  <Home_ID datatype=”integer”>1</Home ID>
  <Location datatype=”enumeration”>Kitchen</Location>
  <Sensor>    <Temperature>
      <Temp_value datatype=”float”>26.5</Temp value>    </Temperature>
  </Sensor></Device>
```

P2P in domotics environment

During mapping:

- attribute Home_ID is mapped to its 1-to-1 value
- attribute Location is mapped to the corresponding value in the mapping "Kitchen" due to the 1-to-1 mapping
- attribute Temp_value (26.5) is mapped to intervals 25-30

P2P in domotics environment

The mapped data is:

```
<Device>
  <Home_ID datatype=”integer”>1</Home ID>
  <Location datatype=”enumeration”>Kitchen</Location>
  <Sensor>    <Temperature>
      <Temp_value datatype=”float”>25</Temp value>    </Temperature>
  </Sensor></Device>
```

P2P in domotics environment

A software component needs to interrogate the repository and obtain data from all temperature sensors in the house whose last reading reported a temperature value greater than 25.5°C

<Device>
<Home_ID datatype="integer">1</Home ID> <Location datatype="enumeration">*</Location> <Sensor> <Temperature> <Temp_value datatype="float"> >25 </Temp value> </Temperature> </Sensor></Device>

P2P in domotics environment

During mapping:

- attribute Home_ID is mapped to its 1-to-1 value
- attribute Location, due to the * wildcard, is mapped to all locations defined in the mapping (Kitchen, Living Room, etc.)
- attribute Temp_value, due to the constraint >25 is mapped to intervals 25-30, 30-35 and 35-40

All these mapped values are combined in 9 different mapped queries.

P2P in domotics environment

One of the 9 mapped queries is the following:

```
<Device>
  <Home_ID datatype=”integer”>1</Home ID>
  <Location datatype=”enumeration”>Kitchen</Location>
  <Sensor>    <Temperature>
      <Temp_value datatype=”float”>25</Temp value>    </Temperature>
  </Sensor></Device>
```

Agenda

- Introduction
- P2P as middleware
- The churn phenomena
- Gnutella: a famous example
- What is an overlay network
- Peer Sampling
- Structured Vs unstructured overlay network:
  - Cyclon: an unstructured P2P system
  - Chord/DHT: a structured overlay network
  - P2P in domotics environment
  - Kelips: an hybrid system
- Search in P2P network

Kelips

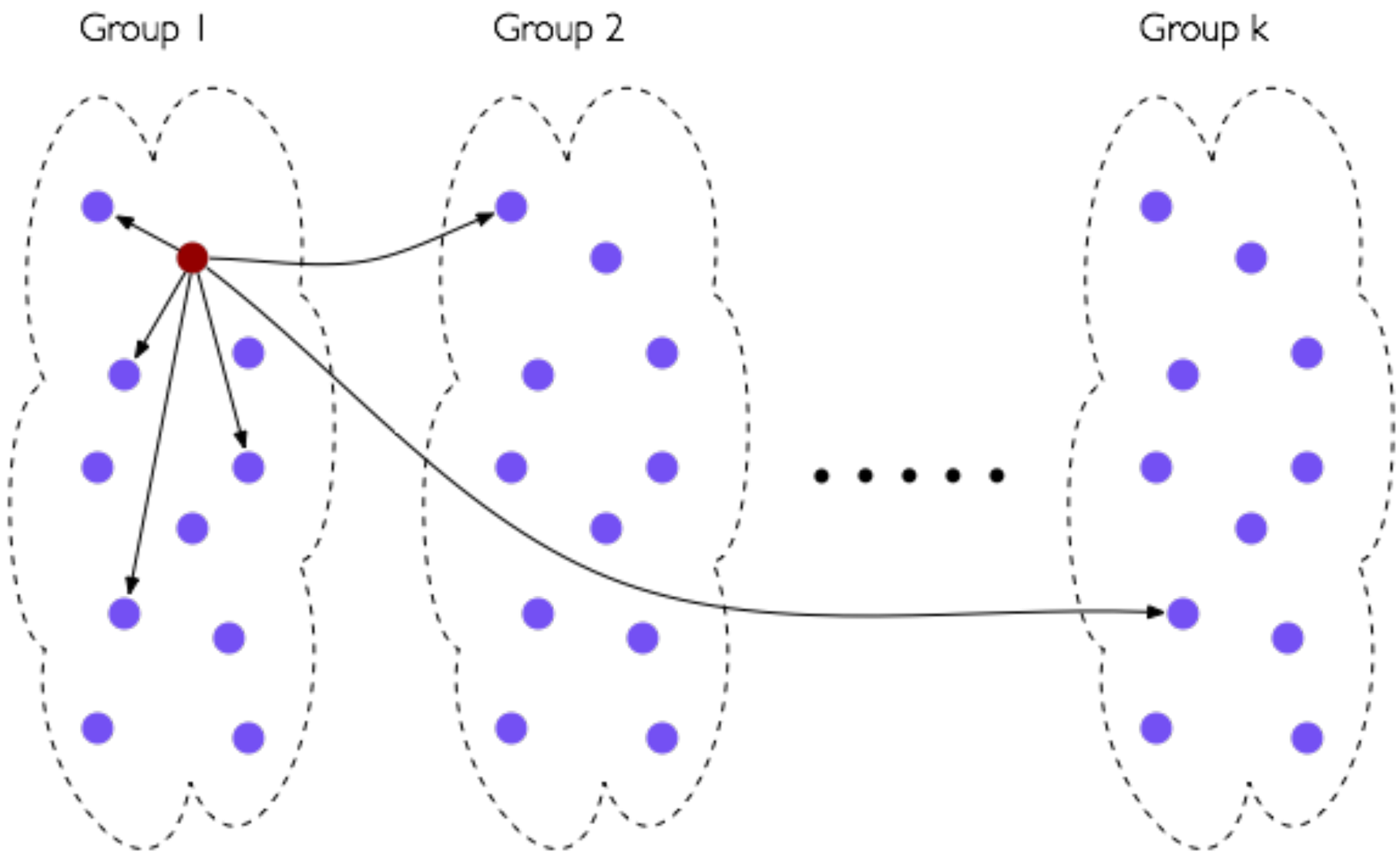
Indranil Gupta, Ken Birman, Prakash Linga, Al Demers, Robbert van Renesse, "Kelips: Building an Efficient and Stable P2P DHT Through Increased Memory and Background Overhead", Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)

Kelips is a distributed algorithm implementing a hybrid structured/ unstructured scheme with store/lookup capabilities:

- Every node is mapped through a hash function to one of several groups.
- Every object, mapped to one of the groups, is managed by one of the nodes belonging to that group.
- Every node manages three data structures:
  - Local view: the set of nodes belonging to it same group.
  - Contacts: a subset of nodes belonging to other groups.
  - Filetuples: an index of objects managed by nodes in its same group.
- Data structure content is maintained up-to-date through a continuous gossip stream of information among nodes.

Kelips

An example:



Kelips

Object search:

- If the node group and searched object group match, the node just looks into filetuples.
- Otherwise it looks into contacts for a node belonging to the same group of the searched object and forward it the request.
- To improve robustness versus stale information in data structures, the node can forward randomly the request through a (short) random walk.

Agenda

- Introduction
- P2P as middleware
- The churn phenomena
- Gnutella: a famous example
- What is an overlay network
- Peer Sampling
- Structured Vs unstructured overlay network:
  - Cyclon: an unstructured P2P system
  - Chord/DHT: a structured overlay network
  - P2P in domotics environment
  - Kelips: an hybrid system
- Search in P2P network

Search in P2P systems - Summary of Peer-to-peer systems

- What is a peer-to-peer system ?
  - A set of hosts (nodes) that communicate and exchange information to implement a service/application
  - Usually all the hosts have the same role in the system
  - Results stem from collaboration among nodes
- Common characteristics:
  - Large scale
    - number of nodes
    - global load
  - Nodes unreliability
    - Faults
    - Churn
    - Selfish/Byzantine behaviours
  - Lack of central administration => self-administration

Search in P2P systems

- **Problem**: the user must access some data stored somewhere in the system
- The search mechanism, given a query, returns data satisfying the query or identifiers of nodes where these data are stored.
- Challenges:
  - Nodes are unreliable: they can leave the system at any time, due to crash o leave
  - The scale of the system is large
    - searched data can be stored on one single node among millions
    - millions of users also mean a large number of concurrent searches

Search in P2P systems

Expressiveness

- The way the system let user specify queries.

Examples:

- **Key lookup**: each resource is identified with a "key" (identifier). A search specifies a key. The mechanism returns data identified by the searched key.
- **Keywords query**: a query contains a set of searched keywords (e.g. Q="low-cost flight Paris". The mechanism returns data containing the searched keywords. Results of keyword searches can be sometimes ranked and filtered by relevance.
- **Aggregate**: aggregate functions (Sum, Count, Max, etc.) can be used to obtain properties extracted from the whole data collection (e.g. How many documents contains the world "p2p"?)
- **SQL**: Current researches on supporting SQL in P2P systems are preliminary.

Search in P2P systems

- **Stop condition**
  - Defines the "completeness" of the answer expected from the system
    - every result
    - the first result found
    - a subset of the results with a certain size

Search in P2P systems - Non-functional requirements

Quality of Service

User-perceived qualities.

- ***Accuracy*** - the result set does not contain data that do not match the query
- ***Result set size***
  - *Completeness* - the result set contains all data that satisfies the search criterion
  - *Satisfaction* - the first *n* results are provided to the user
  - A single result is sufficient
- ***Responsiveness***
  - Time to obtain the first result
  - Time to obtain the entire set of results
- ***Validity***
  - the output of the query belogs to an unpredictable set of processes

More specific QoS metrics can depend on applications.

Search in P2P systems - Non-functional requirements

- **Efficiency**
  - Reduced resource usage to answer a search
    - Bandwidth
    - Memory
    - CPU
    - Battery
    - ...
- **Robustness**
  - failures and dynamics should not impact service efficiency and QoS

Search in P2P systems - Design choices

- **System topology**
  - Tree, ring, torus, mesh, butterflies, random graphs, k-regular graphs, etc.
- **Data positioning**
  - ***Source***
    - Data is left on the node that inserted it in the system
  - ***Structure***
    - Data (or a pointer) is moved to a specific node
  - ***Replicated (with structure)***
  - ***Replicated (random)***

Search in P2P systems - Design choices

- **Routing**
  - How queries are routed inside the system to reach those nodes where they are positively evaluated
  - **Blind search approach**
    - No hint about the position of searched data
    - ***Flooding-based techniques***
      - simple
      - robust
      - expensive
    - ***Random walks***

Search in P2P systems - Design choices

- **Routing**
  - How queries are routed inside the system to reach those nodes where they are positively evaluated
  - **Informed search approach**
    - Information about data positioning is first distributed in the system and then exploited to improve search mechanism performance
    - ***Exact search***
      - like in DHTs
    - ***Hint-based***
      - Approximate information about data location

Search in P2P systems - Design choices

- **Routing**
  - How queries are routed inside the system to reach those nodes where they are positively evaluated
  - **Aggregation**
    - No hint about the position of searched data
    - ***Estimation***
      - Sample and Collide
    - ***Real Aggregation***
      - like in a database

Search in P2P systems - blind search

Breadth First Search (BFS) / Flooding

- No specific data positioning strategy
- The query is routed to all the nodes in the network
- No limits on expressiveness
- Huge overhead => low efficiency
- Robust
- Returns complete results

Search in P2P systems - blind search

Limited horizon

- No specific data positioning strategy
- The query is flooded in the network
- The search scope is limited through Time-To-Live.
- No limits on expressiveness
- Robust
- Completeness of the result set cannot be guaranteed
- Goal: reduce query forwarding
- [Gnutella, www.stanford.edu/class/cs244b/gnutella_protocol_0.4.pdf]

Search in P2P systems - blind search

Iterative deepening

- Similar to limited horizon
- Various runs with increasing TTL values until results are found
- Suited to applications where a subset of results is needed
- Goal: reduce query forwarding in most cases while still providing result completeness
- [Yang, Garcia-Molina - "Improving Search in Peer-to-Peer Networks", ICDCS '02]

Search in P2P systems - blind search - cache based

Uniform index caching (UIC)

- Equivalent to limited horizon but...
- Each node maintains in a cache the last *n* query answers that it forwarded
- Queries contained in the cache are answered quickly, without the need of forwarding them.
- Problems related to cache management
- Sensible to topology changes
- Goal: improve responsiveness and reduce query forwarding
- [Markatos - "Tracing a large-scale Peer to Peer System: an hour in the life of Gnutella", IEEE/ACM International Symposium on Cluster Computing and the Grid, 2002.]

Search in P2P systems - blind search

Random walks

- Simple: each node forwards the query to one of its neighbors
- Stop condition: depends on the number of desired results
- Needs STRONG data replication or caching to deliver acceptable robustness and responsiveness levels.
- Goal: reduce query forwarding while quickly adapting to topology changes

Search in P2P systems - Informed search

DHTs

- Nodes constituting the system are organized in order to realize a distributed implementation of an hash table
  - each node is responsible for maintaining a specific subset of data
  - an intelligent query routing mechanism route the query to the destination node in a logarithmic number of steps
  - the mapping between data and nodes is realized through a globally known consistent hash function
- Elegant and efficient solution for exact-match queries => limited expressiveness
- Some solutions are sensible to churn

- [I. Stoica, R. Morris, D. Karger, M. F. Kaashoek and H. Balakrishnan, Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications, Proceedings of ACM SIGCOMM, 2001]
- [A. Rowstron and P. Druschel, Pastry: Scalable, Decentralized Object Location and Routing for Large-Scale Peer-to-Peer Systems, Proceedings of International Conference on Distributed Systems Platforms (Middleware), 2001]
- [S. Ratnasamy, P. Francis, M. Handley, R. Karp and S. Shenker, A scalable content-addressable network, Proceedings of ACM SIGCOMM, 2001]
- [P. Maymounkov and D. Mazires. Kademlia: A peer-to-peer information system based on the xor metric. In Peer-to-Peer Systems: First InternationalWorkshop, IPTPS 2002]

Search in P2P systems - Informed search

Hybrid systems - Kelips

- DHT-like infrastructure with mixed topologies:
  - high level - structured
  - low level - random
- Based on gossip algorithms (probabilistic approach)
- Self-stabilizing properties
- More resistant to churn/failures
- Constant cost for searches
- [K. Birman et al., 2007]

Search in P2P systems - Informed search

Probabilistic routing tables

- Routing tables contain approximated information about data positioning
- Information is spread from the data source and recorded in routing tables using exponentially decaying bloom filters
- The farer a node is from the data source, the more approximated the information stored in its routing table is
- [Kumar, Xu, Zegura - "Efficient and Scalable Query Routing for Unstructured Peer-to-Peer Networks", INFOCOM '05]

Search in P2P systems - Aggregation

Estimation

- Estimation try to reach the final result observing a sub-set of the whole P2P network
- Common query: counting, average, sum, max, min
- Sample and Collide method
- Approximated result
- [Massoulié, Le Merrer, Kermarrec, Ganesh - "Peer Counting and sampling in overlay networks: random walk methods", PODC '06]

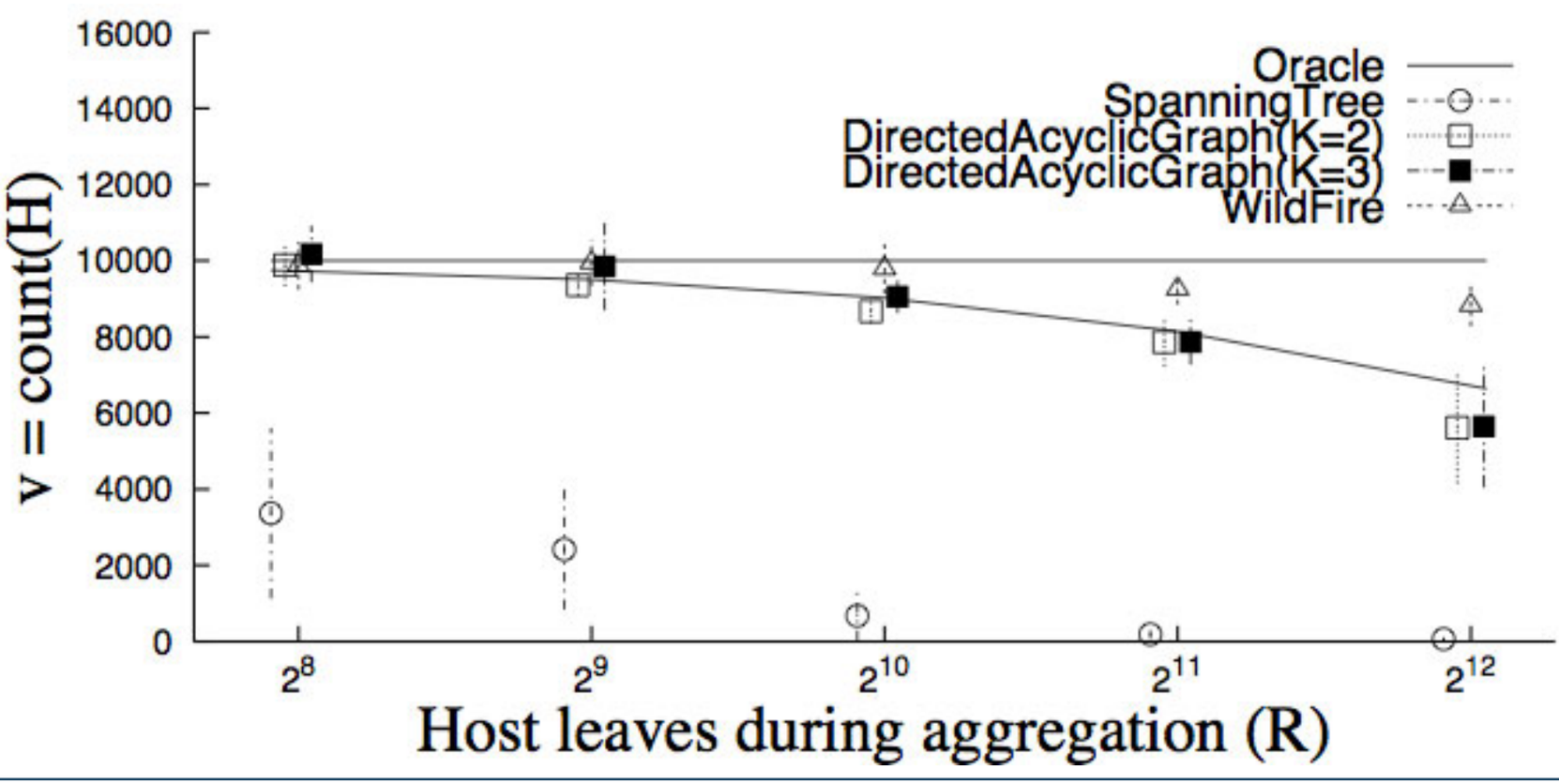
Search in P2P systems - Aggregation

Real Aggregation

- A real aggregation is based on two steps: **broadcast** and **convergecast**
  - during the **broadcast** phase all of the nodes are informed about the query to compute
  - during the **convergecast** phase each result of the query is routed through an aggregation schema till the root
- The root of the aggregation schema computes the final aggregation
- Common query: potentially everything
- Biased by the churn
- **Exact result**
- [Bawa, Gionis, GArcia-Molina, Motwani - "The price of validity in dynamic networks", Journal of Computer and System Sciences vol. 73 '07]

Search in P2P systems - Aggregation - Validity Issue

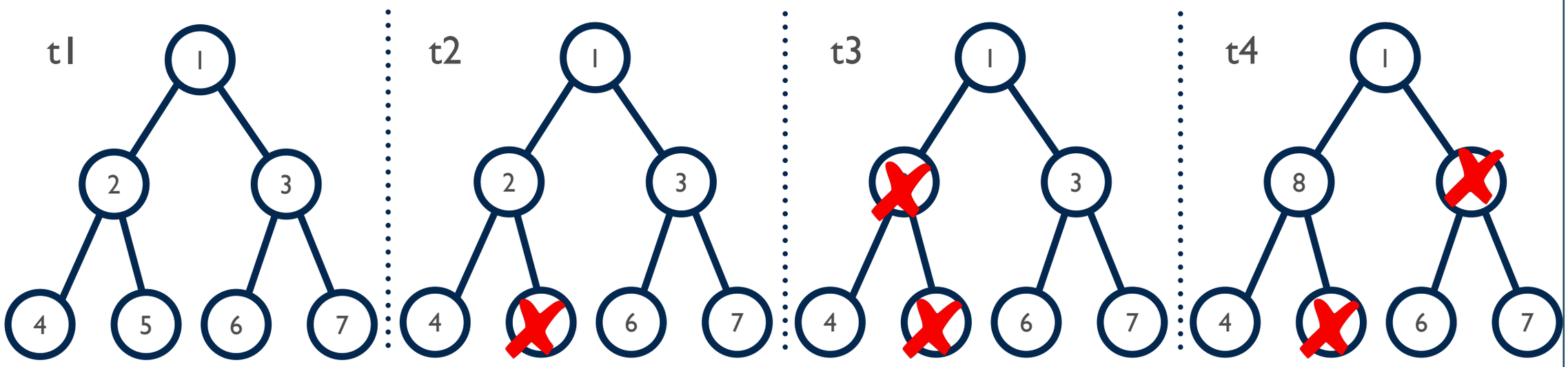
- An aggregate result is exact in absence of churn
- If the churn is present the P2P network on which the query is invoked can change during the execution
- The final result could belong to one of the different configurations or to a composition of them
  - **issue: the validity of the final result couldbe compromized!**



Search in P2P systems - Aggregation - Validity Issue

- Let ***q*** be a query, ***T*** be the query time interval and ***v(q)*** the result of the query q; ***v(q)*** can respect one of the following Validity Property:
  - **Snapshot Validity**: it means that ***v(q)*** belongs to an existing configuration of the network in T, i.e. it contains exactly all of the nodes present in the snapshot
  - **Interval Validity**: it means that ***v(q)*** has to take into account (i) at least all of the nodes present in the network for the entire interval T and (ii) at most all of the nodes present in the network at least for a while in T
  - **Single Site Validity**: it means that ***v(q)*** has to take into account at least all of the nodes that are connected to the root through a valid path for the entire interval T
- if **v(q)** contains more than once the contribution of a single process it does not respect validity properties. It is an error.
- **v(q)** can also respect not even one of the presented properties

Search in P2P systems - Aggregation - Validity Issue - Example



Query **q** starts in t1 and end in t4 with **v(q)**

1. v(q) = {1} is single site valid
2. v(q) = {1,4,6,7} is interval valid
3. v(q) = {1,2,3,4,5,6,7,8} is interval valid
4. v(q) = {1,2,3,4,5,6,7} is snapshot valid
5. v(q) = {1,3,4,6,7} is snapshot valid

Adriano Cerocchi
cerocchi@dis.uniroma1.it
www.dis.uniroma1.it/~cerocchi

MIDLAB Middleware Laboratory