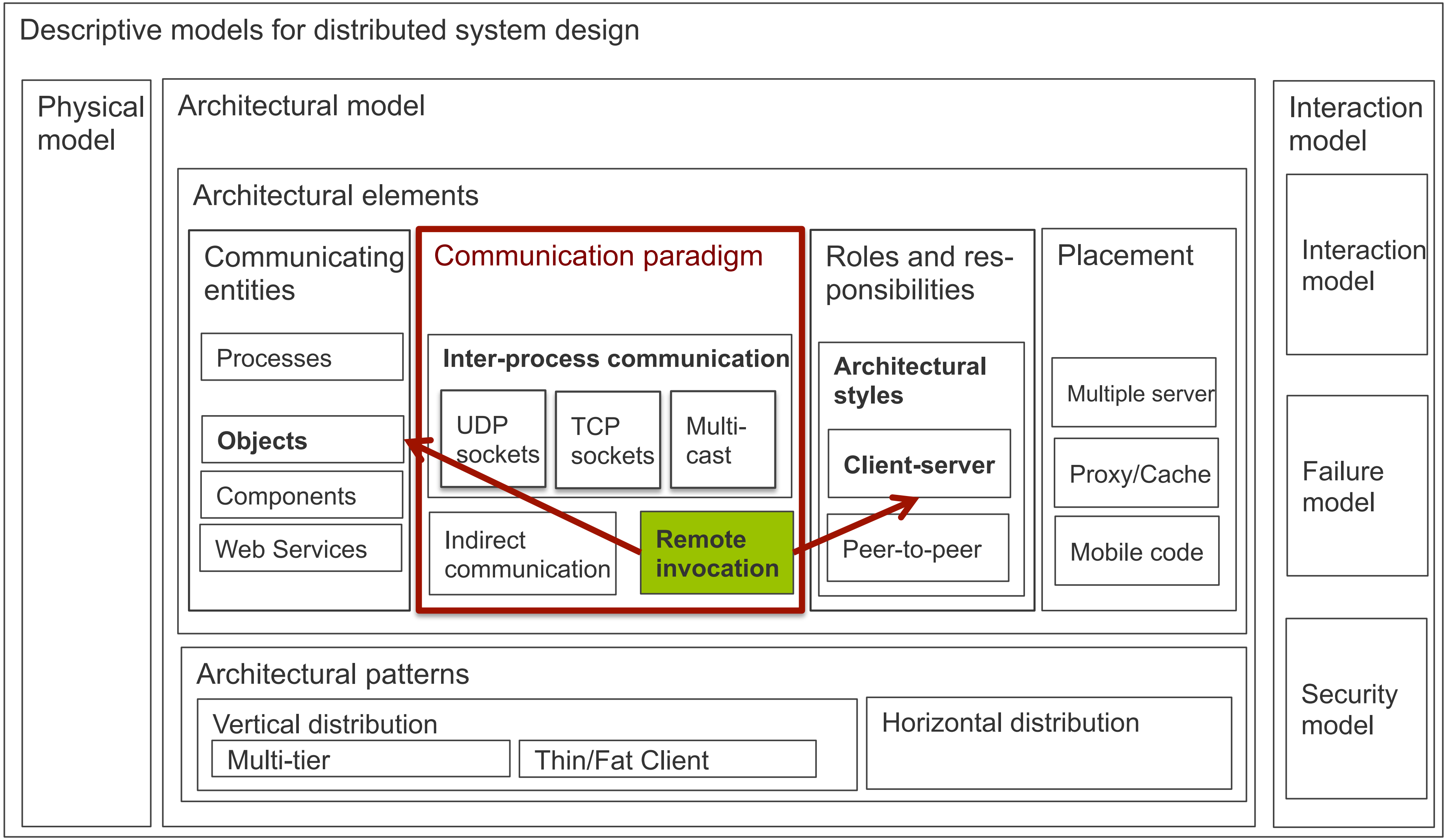
Prof. Dr. Claudia Müller-Birn
Institute for Computer Science, Networked Information Systems

Freie Universität Berlin

Indirect Communication

Netzprogrammierung
(Algorithmen und Programmierung V)

Our topics last week



Our topics today

- Key properties of indirect communication
- Space and time uncoupling

- Group communication
  - Key properties
  - Programming model
  - Implementation issues

- Publish and subscribe systems
  - Key properties
  - Programming model (esp. topic-based, content-based, type-based)
  - Implementation issues (p/s architectures)

- Message queues

Freie Universität Berlin

Indirect Communication

Introduction

Indirect communication is defined as communication between entities in a distributed system through an intermediary with no direct coupling between the sender and the receiver(s).

Key properties

Space uncoupling

The sender does not know or need to know the identity of the receiver(s), and vice versa.

Time uncoupling

The sender and the receiver(s) can have independent lifetimes.

Indirect communication is often used in distributed systems where change is anticipated.

Examples

- Mobile environments where users may rapidly connect to and disconnect from the network
- Managing event feeds in financial systems

Space and time coupling in distributed systems

	Time-coupled	Time-uncoupled
Space coupling	Communication directed towards a given receiver or receivers; receiver(s) must exist in that moment in time	Communication directed towards a given receiver or receivers; sender(s) and receiver(s) can have independent lifetimes
Space uncoupling	Sender does not need to know the identity of the receiver(s); receiver(s) must exist at that moment in time	Sender does not need to know the identity of the receiver(s); sender(s) and receiver(s) can have independent lifetimes

Freie Universität Berlin

Relationship with asynchronous communication

Asynchronous communication

A sender sends a message and then continues (without blocking), and hence there is no need to meet in time with the receiver to communicate

Time uncoupling

The sender and the receiver(s) can have independent existence; for example, the receiver may not exist at the time communication is initiated

Indirect communication

Group communication

Introduction group communication

Group communication offers a service whereby a message is sent to a group and then this message is delivered to all members of the group.

Characteristics

- Sender is not aware of the identities of the receivers
- Represents an abstraction over multicast communication

Possible implementation over IP multicast (or an equivalent overlay network), adding value in terms of

- Managing group membership
- Detecting failures and providing reliability and ordering guarantees

Key areas of application

Reliable dissemination of information to potentially large numbers of clients, including **financial industry**, where institutions require accurate and up-to-date access to a wide variety of information sources

Support for **collaborative applications**, where events must be disseminated to multiple users to preserve a common user view – for example, in multiuser games

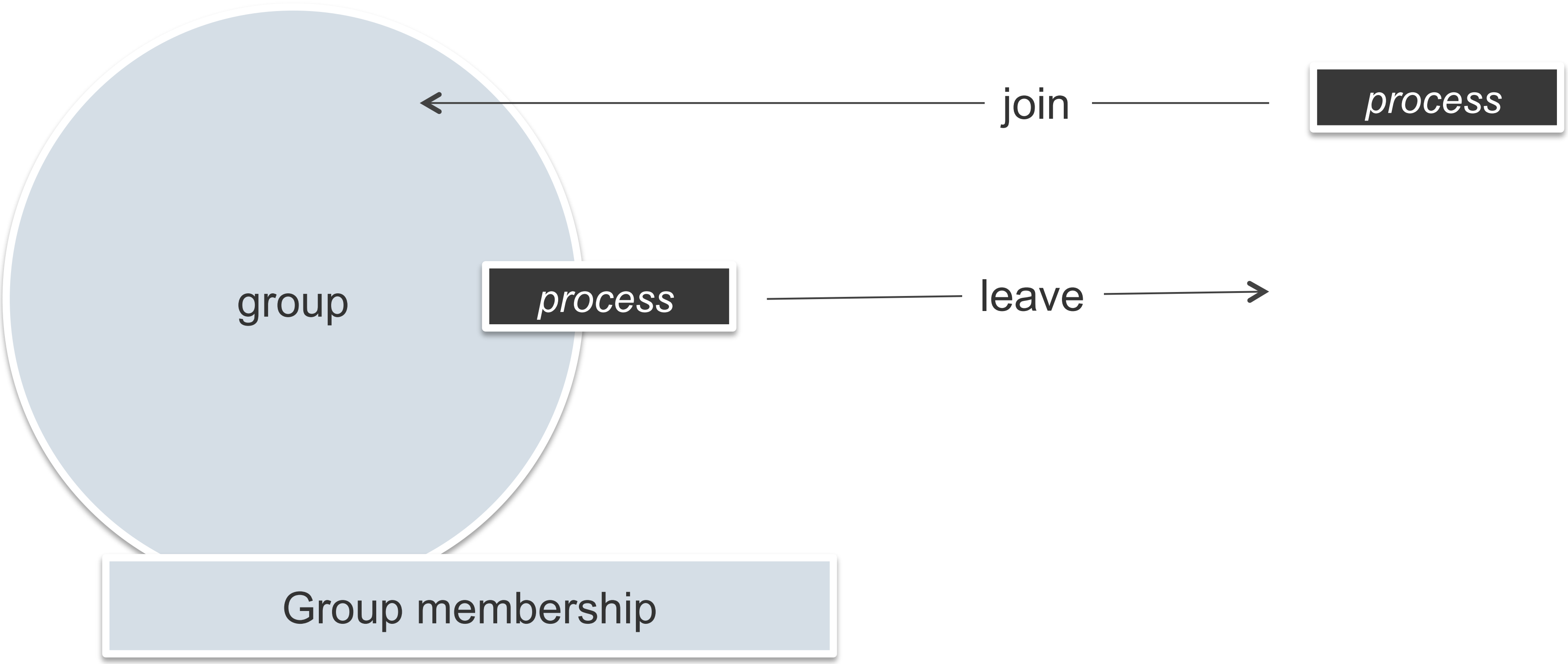
Support for a range of fault-tolerance strategies, including the consistent update of **replicated data** or the implementation of highly available (replicated) servers

Support for **system monitoring and management**, including for example load balancing strategies

Group communication

The programming model

Central concepts



Process groups and object groups

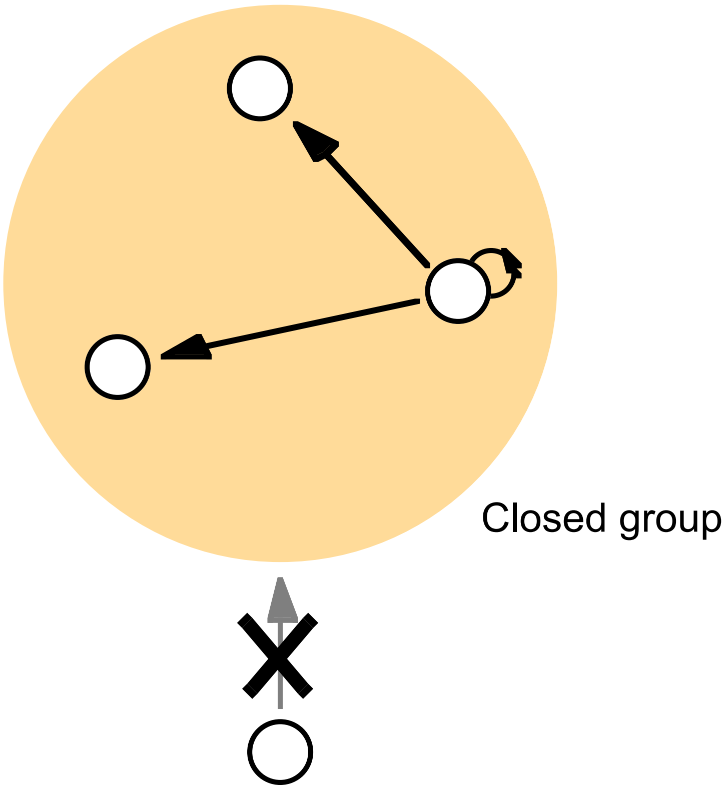
Most group services are suing the concept of **process groups**

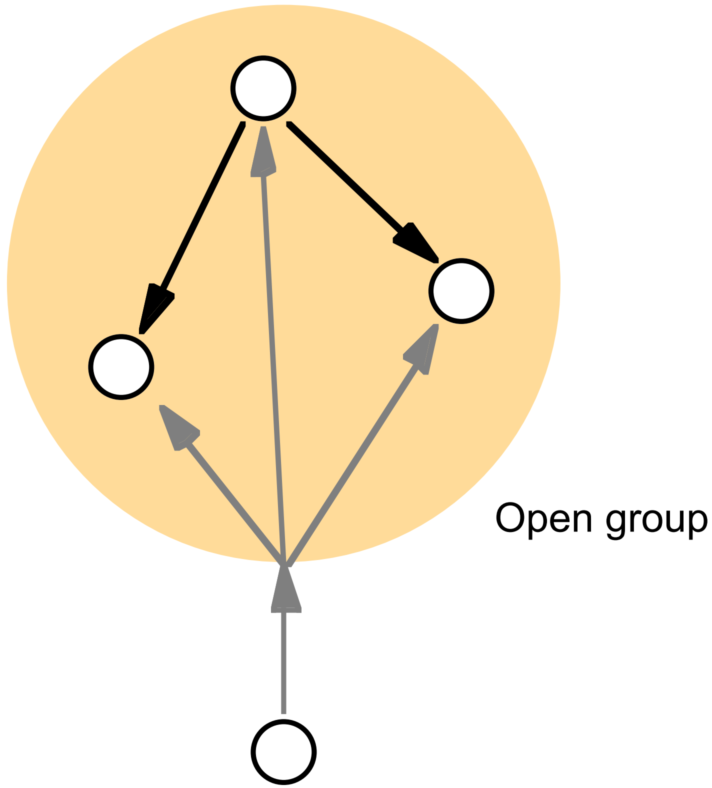
- Communicating entities are processes
- Messages are delivered to processes and no further support for dispatching is provided.
- Messages are typically unstructured byte arrays with no support for marshalling or complex data types

Object groups provide a higher-level approach to group computing

- Collection of objects (normally instances of the same class) that process the same set of invocations concurrently, with each returning responses
- Client objects need to be aware of the replication => they invoke operations on a single, local objects which acts as a proxy for the group
- Proxy uses a group communication systems to send the invocations to the members of the object group

Open versus closed groups

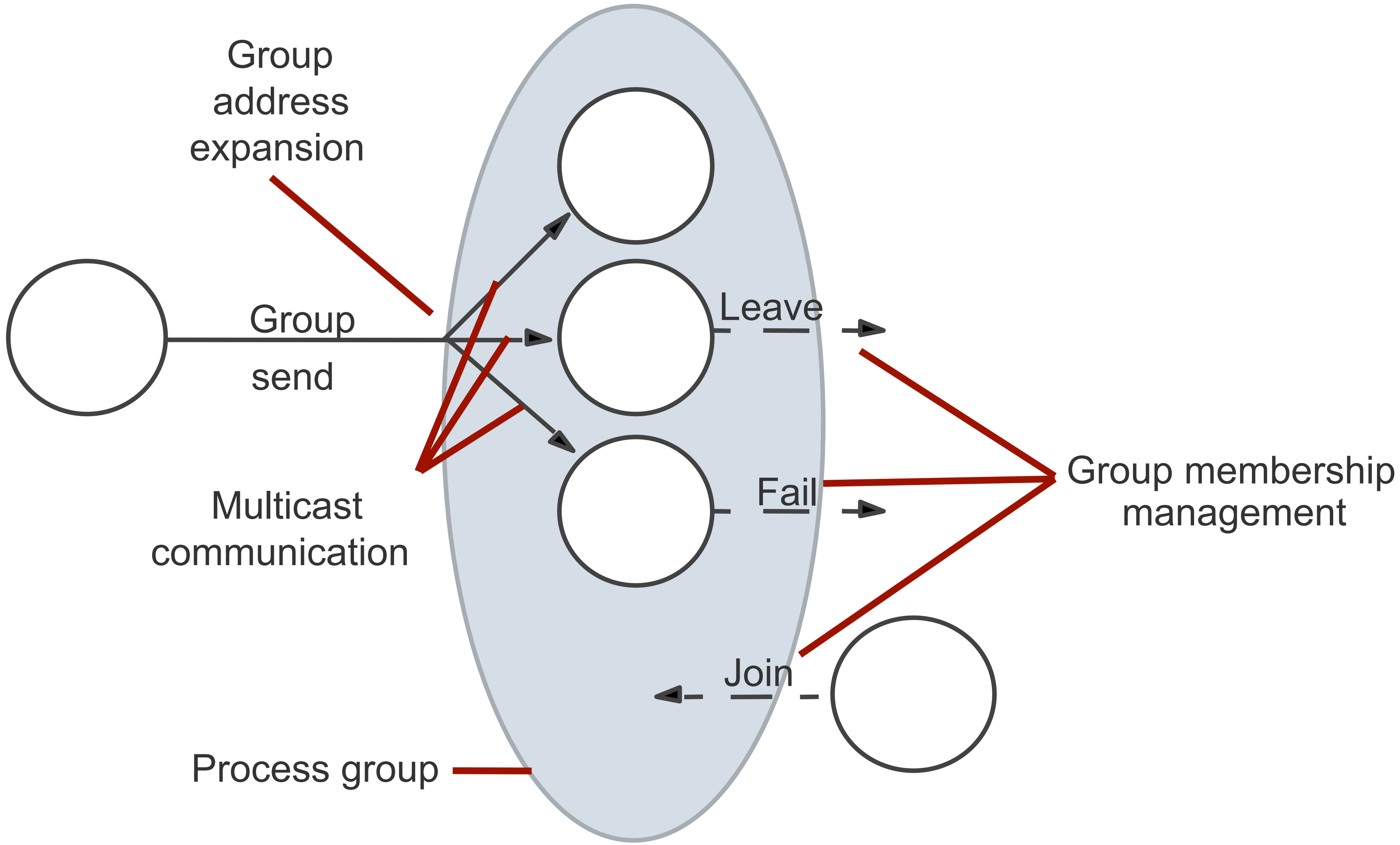




Group communication

Implementation issues

Group membership management



Group membership management

Providing an interface for group membership changes

The membership service provides operations to create and destroy process groups and to add or withdraw a process to or from a group.

Failure detection

The service monitors the group members not only in case they should crash, but also in case they should become unreachable because of a communication failure.

Notifying members of group membership changes

The service notifies the group's members when a process is added, or when a process is excluded.

Performing group address expansion

When a process multicasts a message, it supplies the group identifier rather than a list of processes in the group.

Reliability in multicast

Reliability in one-to-one communication

Integrity: the message received is the same as the one sent, and no messages are delivered twice

Validity: any outgoing message is eventually delivered

Reliable Multicast

Integrity: delivering the messages correctly at most once

Validity: guaranteeing that a message sent will eventually be delivered

(+) Agreement: Stating that if the message is delivered to one process, then it is delivered to all processes in the group

Ordering in multicast

FIFO ordering

First-in-first-out (FIFO) ordering is concerned with preserving the order from the perspective of a sender process, in that if a process sends one message before another, it will be delivered in this order to all processes in the group.

Causal ordering

Causal ordering takes into account causal relationships between the messages, in that if a message happens before another message in the distributed system this is so-called causal relationship will be preserved in the delivery of the associated message at all processes.

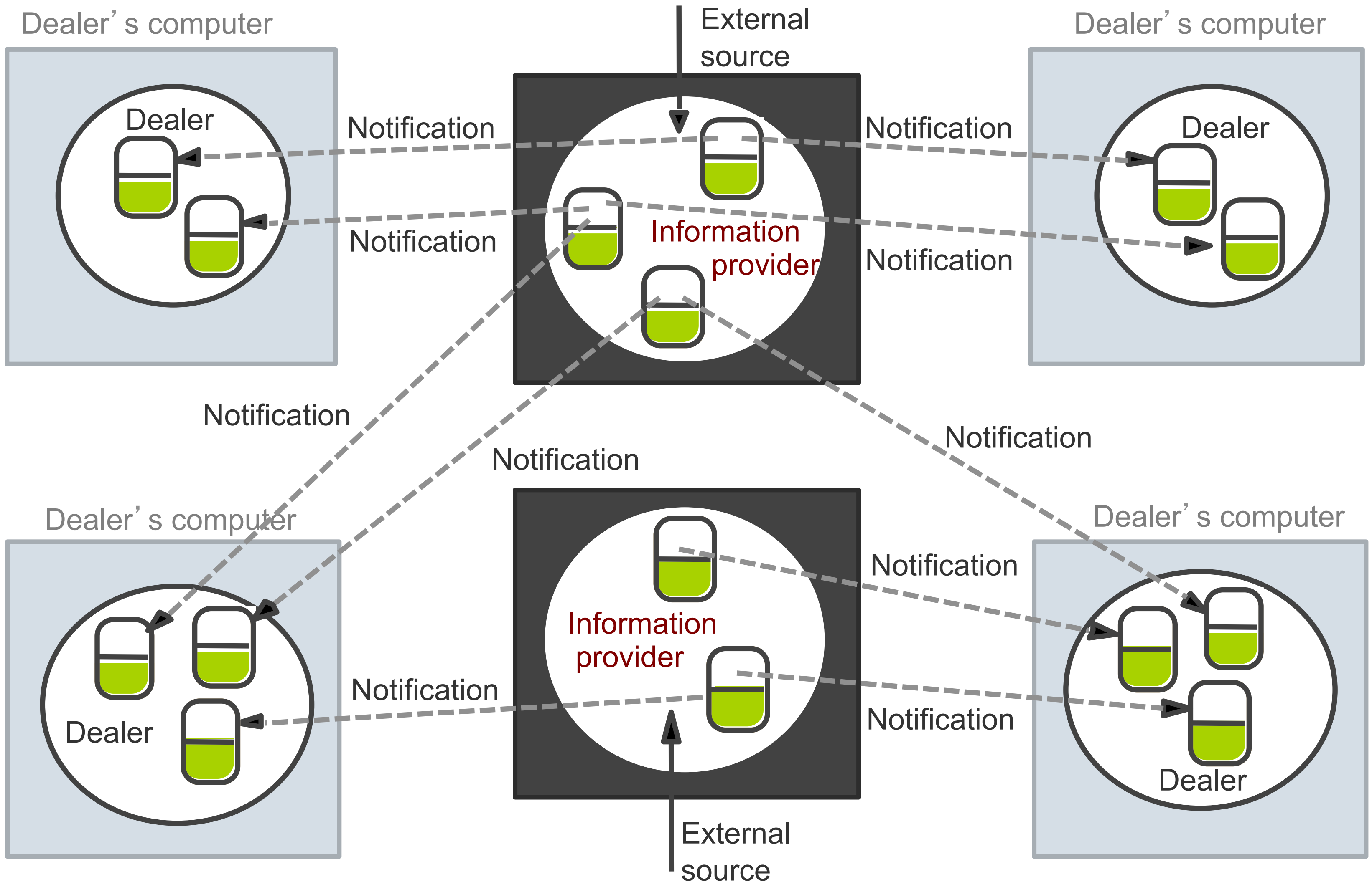
Total ordering

In total ordering, if a message is delivered before another message at one process, then the same order will be preserved at all processes.

Indirect communication

Publish-subscribe systems

Example: Dealing room system

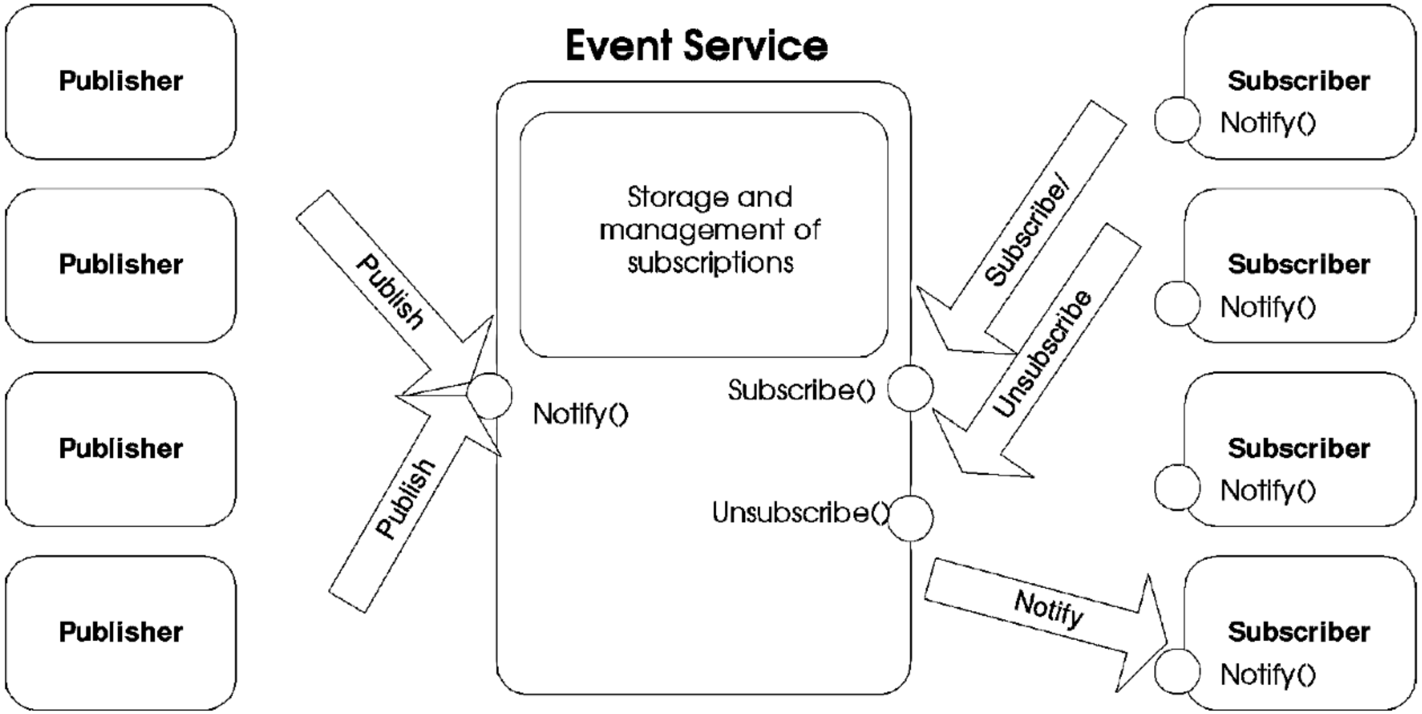


Applications

- Financial information systems
- Other areas of live feeds of real-time data (including RSS feeds)
- Support for cooperative working, where a number of participants need to be informed of events of shared interest
- Support for ubiquitous computing, including the management of events emanating from the ubiquitous infrastructure (e.g., location events)
- A broad set of monitoring applications, including network monitoring in the Internet

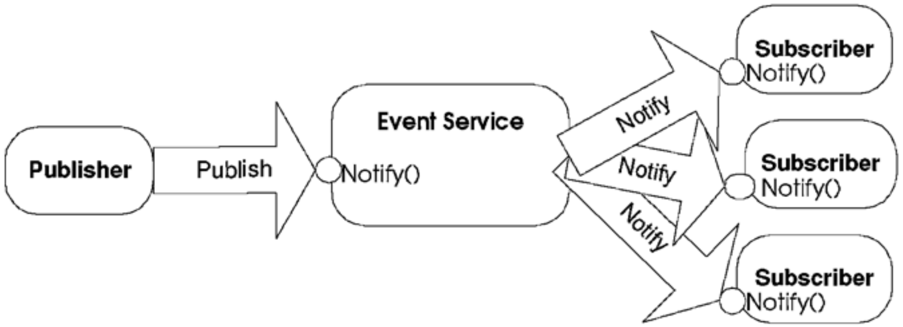
A publish-subscribe system is a system…

…where publishers publish structured events to an event service and subscribers express interest in particular events through subscriptions.

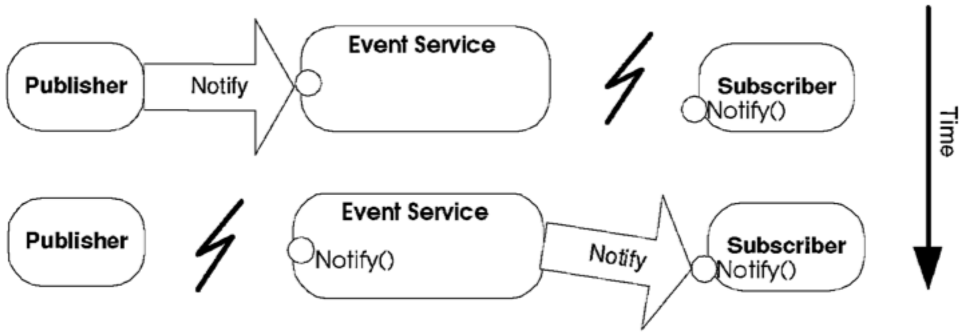


A simple object-based p-s system

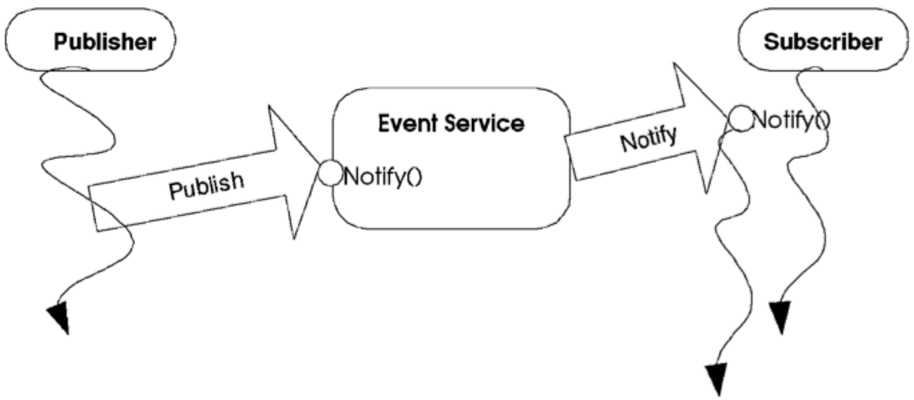
Space, time and sychronization decoupling



Space decoupling



Time decoupling



Synchronization decoupling

Characteristics of publish-subscribe systems

Heterogeneity

When event notifications are used as a means of communication, components in a distributed system that were not designed to interoperate can be made to work together. All that is required is that event-generating objects publish the types of events they offer, and that other objects subscribe to patterns of events and provide an interface for receiving and dealing with the resultant notifications.

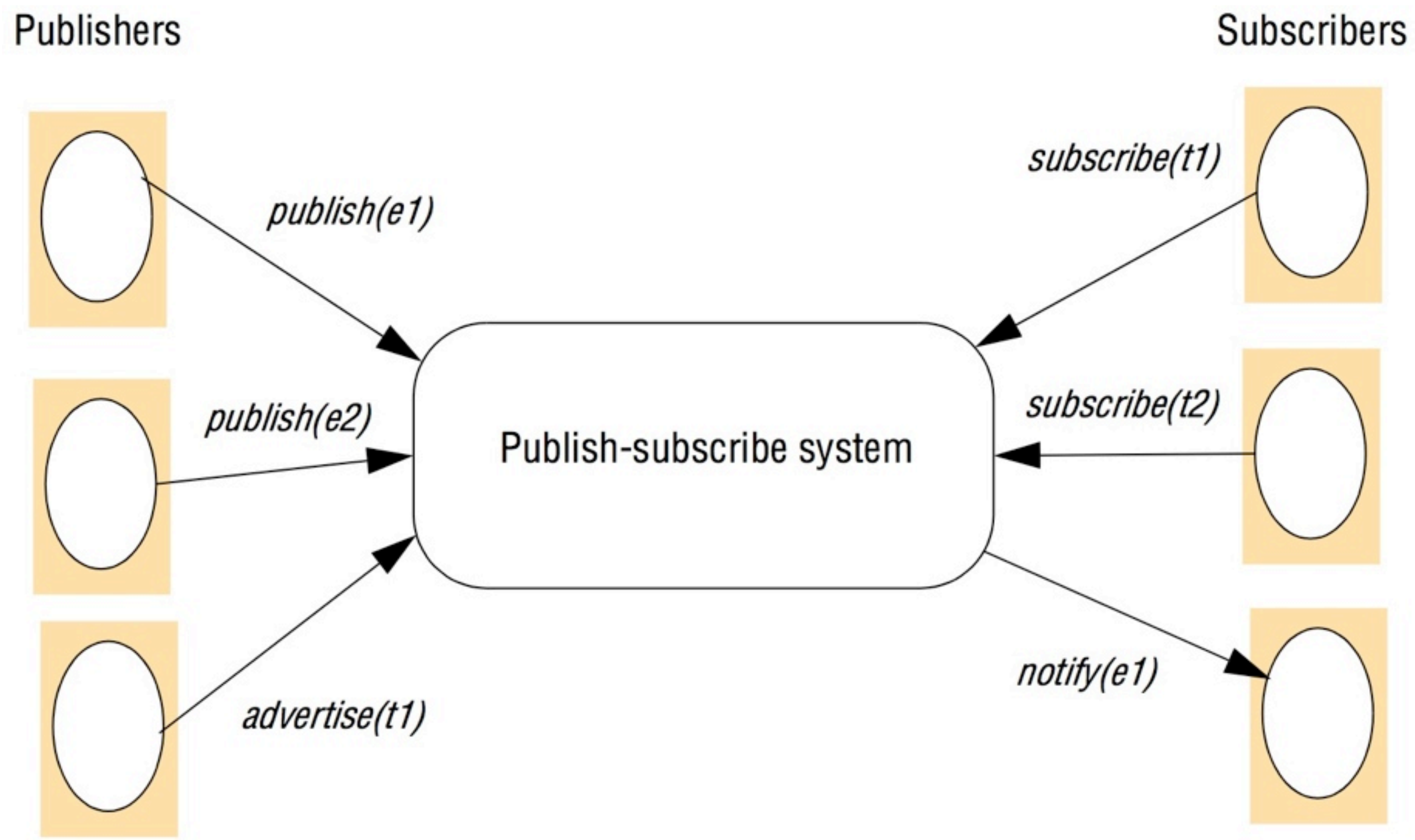
Asynchronicity

Notifications are sent asynchronously by event-generating publishers to all the subscribers that have expressed an interest in them to prevent publishers needing to synchronize with subscribers – publishers and subscribers need to be decoupled.

Publish-subscribe systems

The programming model

The publish-subscribe paradigm



Subscription models of p-s systems

Topic-based

Content-based

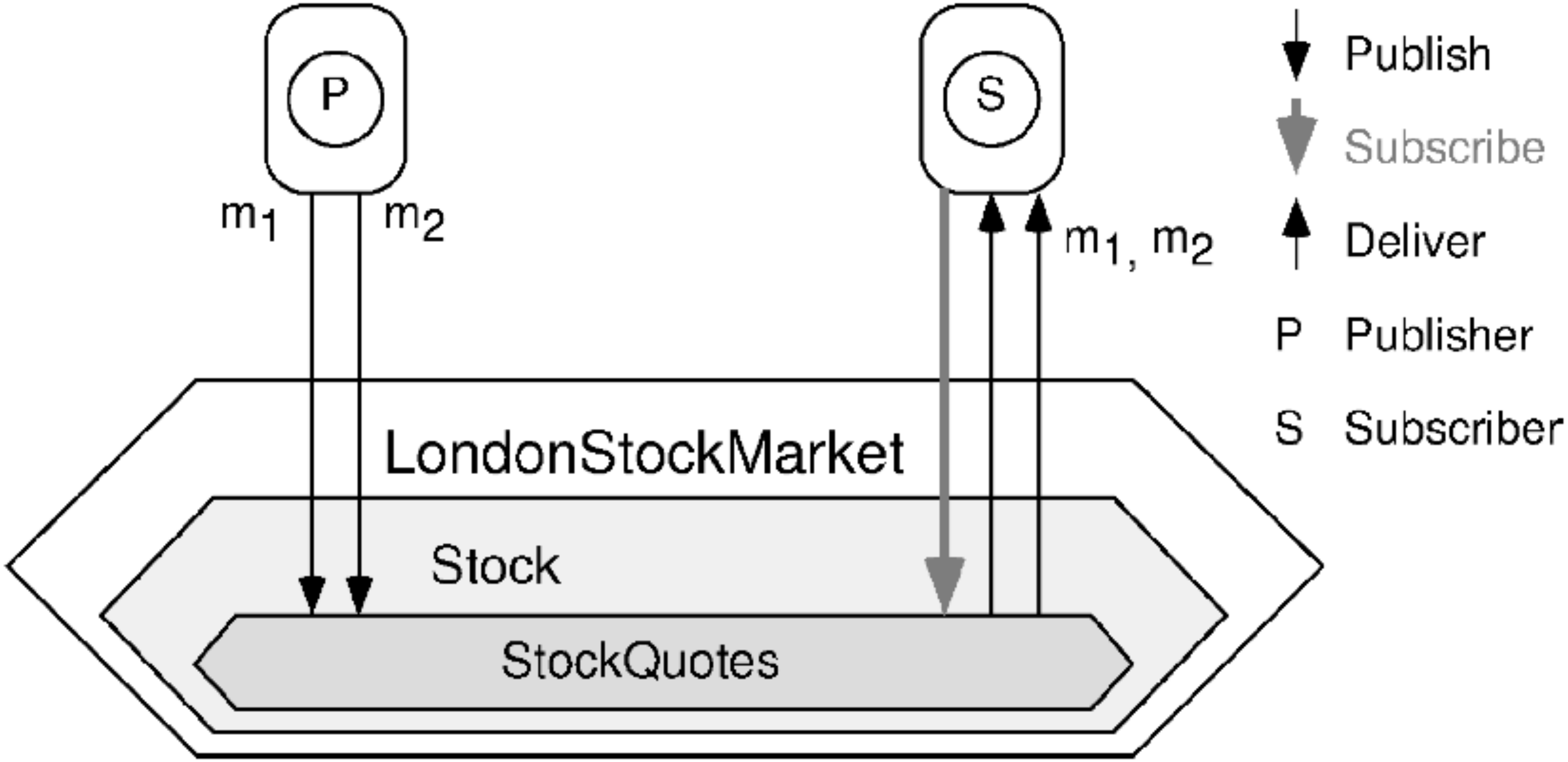
Type-based

Concept-based

XML

Location-awareness

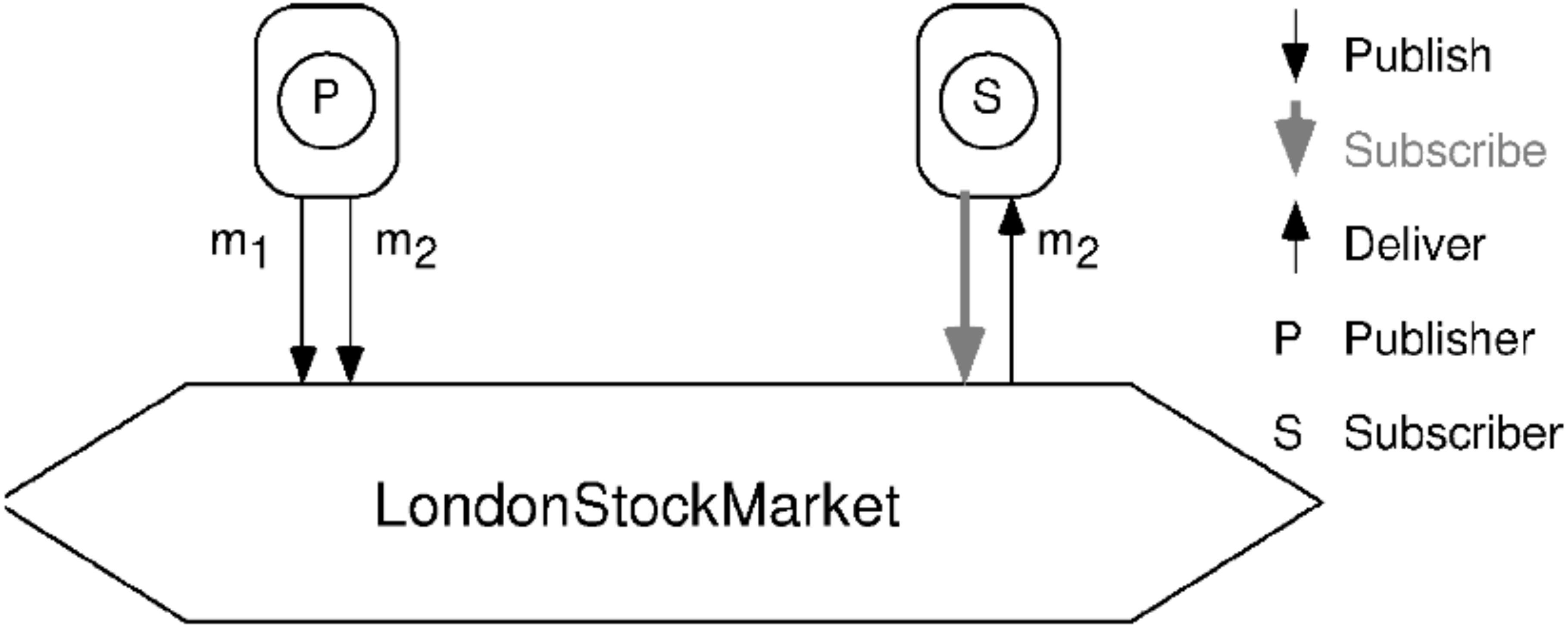
Topic-based publish-subscribe interactions



Topic-based publish-subscribe systems

Notification	Notifications are grouped in topics (or subjects)
Subscriber	Subscriber will receive all notifications related to that particular topic (identified by keywords)
Topics	Topics are structured in a hierarchy
Groups	Subscribing to a topic *T* means a user is becoming a member of group *T*, and publishing an event on topic *T* means becoming a publisher for topic *T*
Communication	Exist a well-defined path (channel) from publisher to all interested subscribers
Subscription	A subscription made to some node in the hierarchy implicitly involves subscriptions to all the subtopics of that node

Content-based publish-subscribe interactions



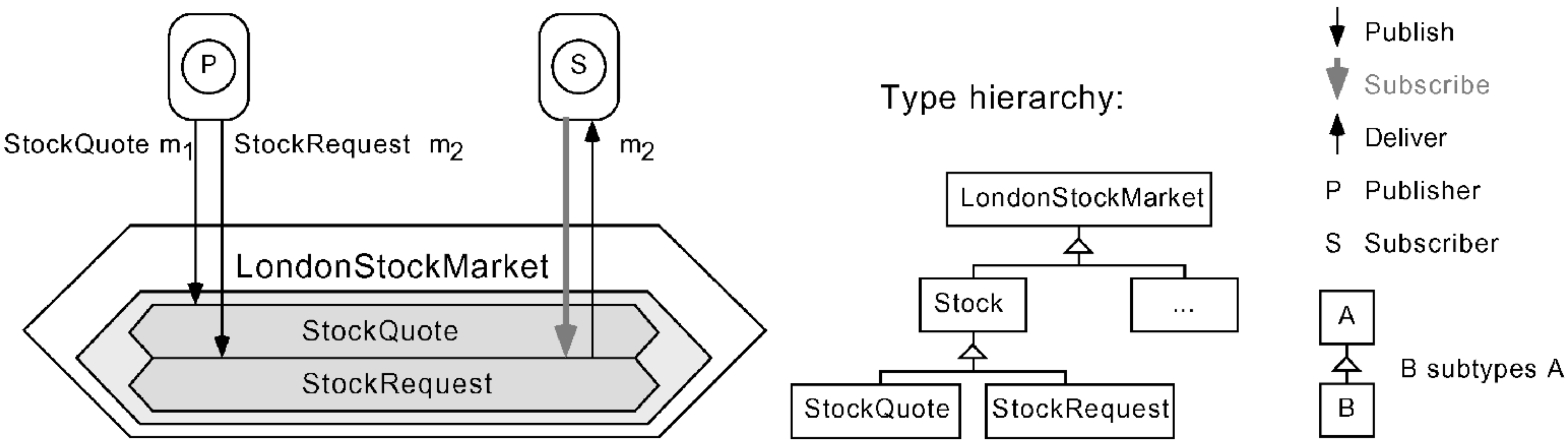
m_1: { ..., company: "Telco", price: 120, ..., ... }

m_2: { ..., company: "Telco", price: 90 , ..., ... }

Content-based publish-subscribe systems

Notification	Each subscriber only receives the notifications that match entirely its individual criteria
Subscriber	Subscribers can announce their individual interests by specifying the properties of the event notifications they are interested in
Topics	Topics filtered according to events and type
Groups	There is no concept of groups as topic are not rearranged into keywords
Communication	Subscription patterns used to identify the events of interest for a given subscriber and propagate events accordingly
Subscription	Subscription scheme based on the actual content of the considered events

Type-based publish-subscribe interactions



Type-based publish-subscribe systems

Notification	Notifications are declared as objects belonging to a specific type, which can thus encapsulate attributes as well as methods
Subscriber	Event subscriber specifies the event type (i.e. topic) it is interested in, and then supplies a filter expression that operates on the attributes provided by this event type.
Topics	Events are objects
Groups	Messages regrouped in a topic usually are of the same type
Communication	Producers publish information on an information bus A subscriber advertises its interest in a type T, which means that it will receive all messages of type T.
Subscription	Declaration of a desired type is the main discriminating attribute.

Freie Universität Berlin

Advantages and disadvantages of p/s systems based on subscription model

	Topic-based	Content-based	Type-based
Advantage	Routing is simple through multicast group to peers that match subscription topics	If notification does not match any subscription it is not sent	Flexible decentralization of implementation, scalable by use of remote content filtering (e.g., locally at the subscriber)
Dis-advantage	Limited expressiveness; Inefficient use of bandwidth	Expressive, but higher runtime overhead requires complex protocols/ implementation to determine the subscriber	Many events need to be pruned for performance reasons

Additional subscription models

Concept-based

Allow to describe event schema at a higher level of abstraction by using ontologies, that provide a knowledge base for an unambiguous interpretation of the event structure, by using metadata and mapping functions

XML-based

Supports a semi-structured data model, typically based on XML documents

Provides natural advantages such as interoperability, independence from implementation and extensibility

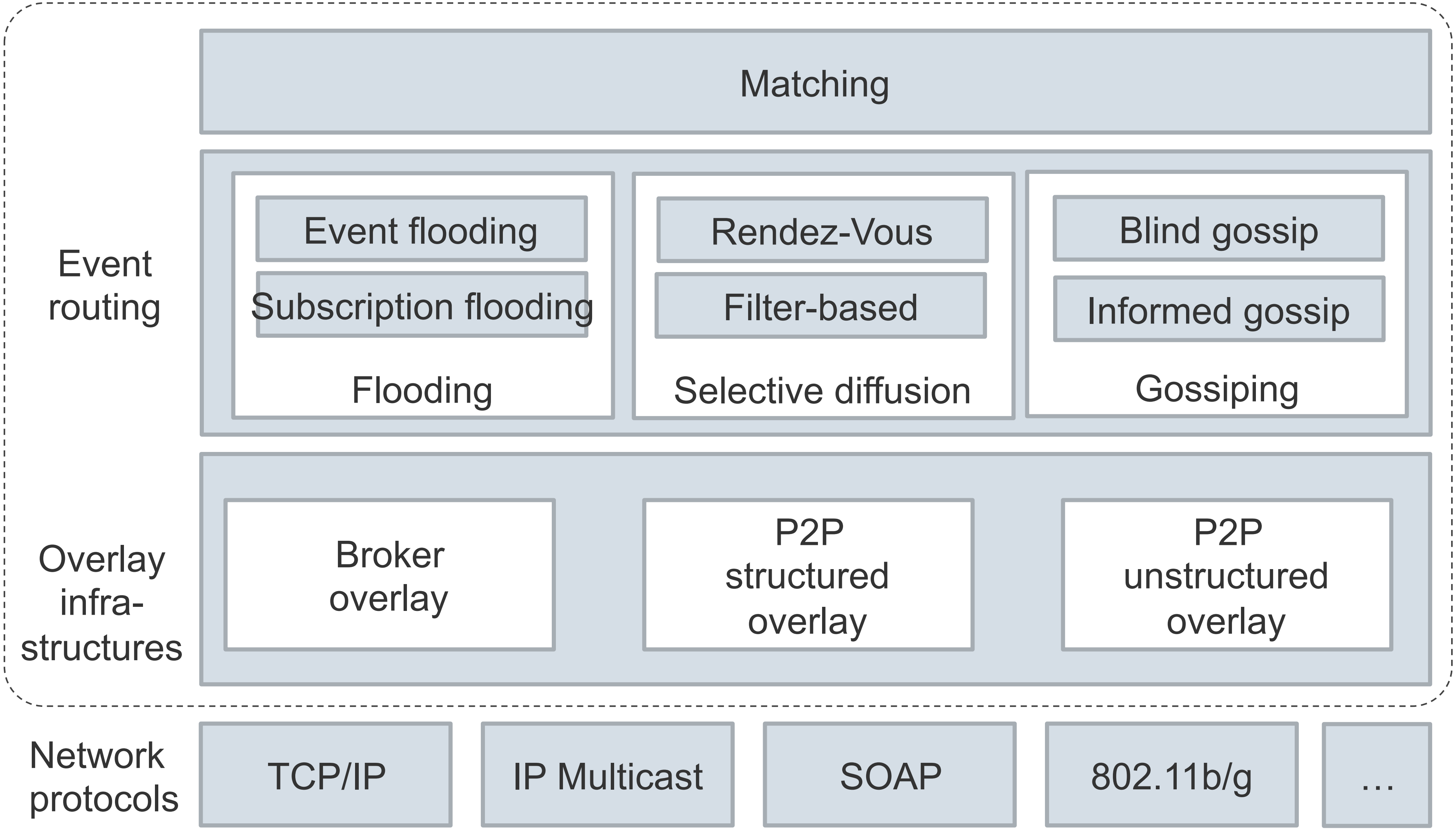
Location-awareness

Used in mobile environments typically require the support for location-aware subscriptions

Publish-subscribe systems

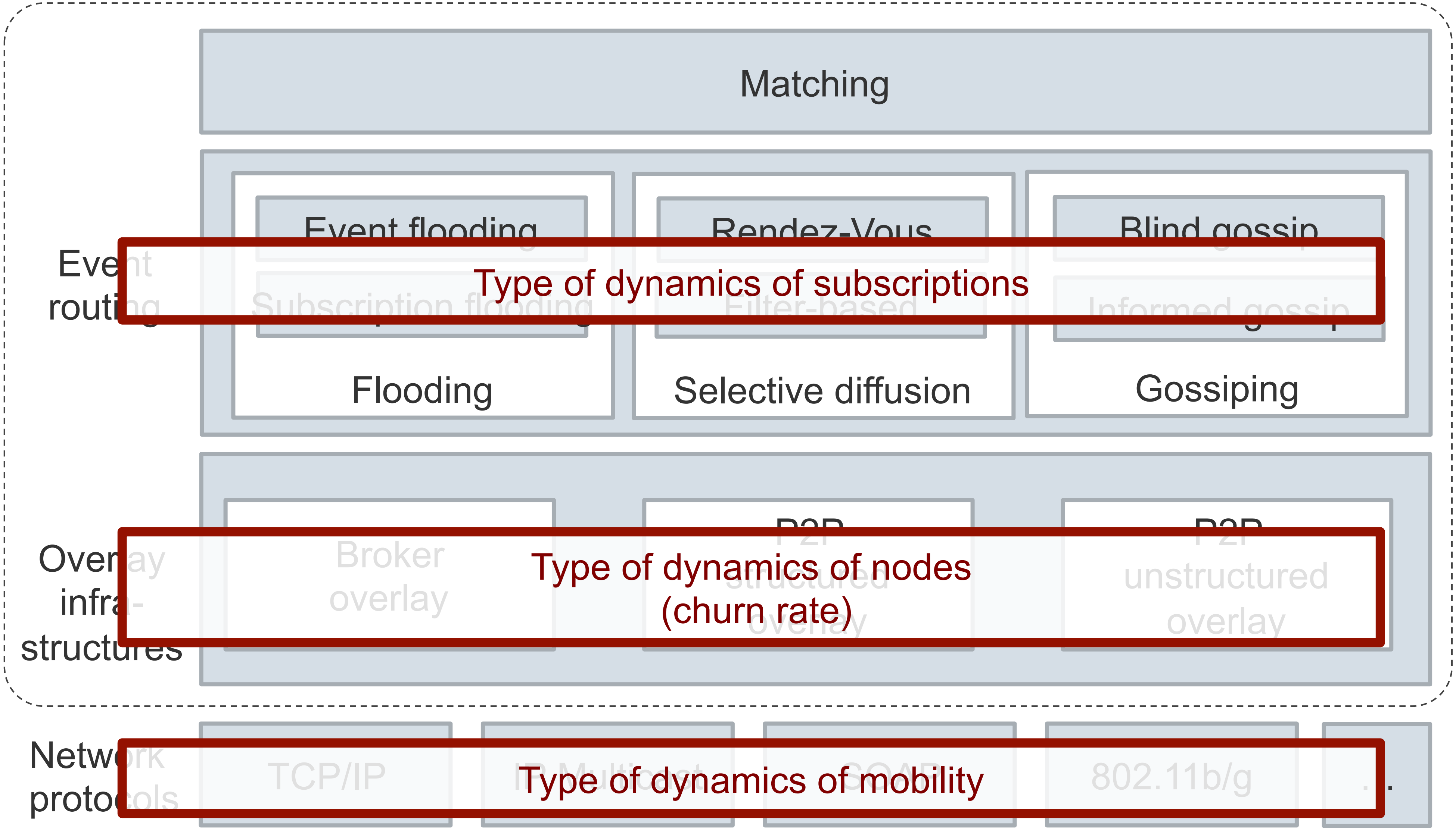
Implementation issues

The architecture of publish-subscribe systems



Freie Universität Berlin

The architecture of publish-subscribe systems



Broker overlay

P/s system is implemented as a set of independent, communicating servers (= broker).

Brokers form an application-level overlay and typically communicate through an underlying transport protocol.

Clients can access the system through any broker and in general each broker stores only a subset of all the subscriptions in the system.

The broker network is implemented as an application-level overlay: connections are pure abstractions as links are not required to represent permanent, long-lived connections.

The topology is assumed to be managed by an administrator. A broker overlay is inherently static.

Peer-to-peer structured overlay

Is a self-organized application-level network composed by a set of nodes forming a structured graph over a virtual key space where each key of the virtual space is mapped to a node.

The structure imposed to the graph permits efficient discovery of data items and this, in turns, allows to realize efficient unicast or multicast communication facility among the nodes.

A structured overlay infrastructure ensures that a correspondence always exist between any address and an active node in the system despite churn (the continuous process of arrivals and departures of nodes of the overlay) and node failures.

A structured overlay allows to better handle dynamic aspects of the systems such as faults and node joins.

Peer-to-peer unstructured overlays

The overlay networks strive to organize nodes in one flat or hierarchical small diameter network (like a random graph) despite churn and node failures.

Differently from broker overlays, nodes in these overlays are not necessarily supposed to be dedicated server but can include workstations, laptops, mobile devices and so on, acting both as clients and as part of the pub/sub system.

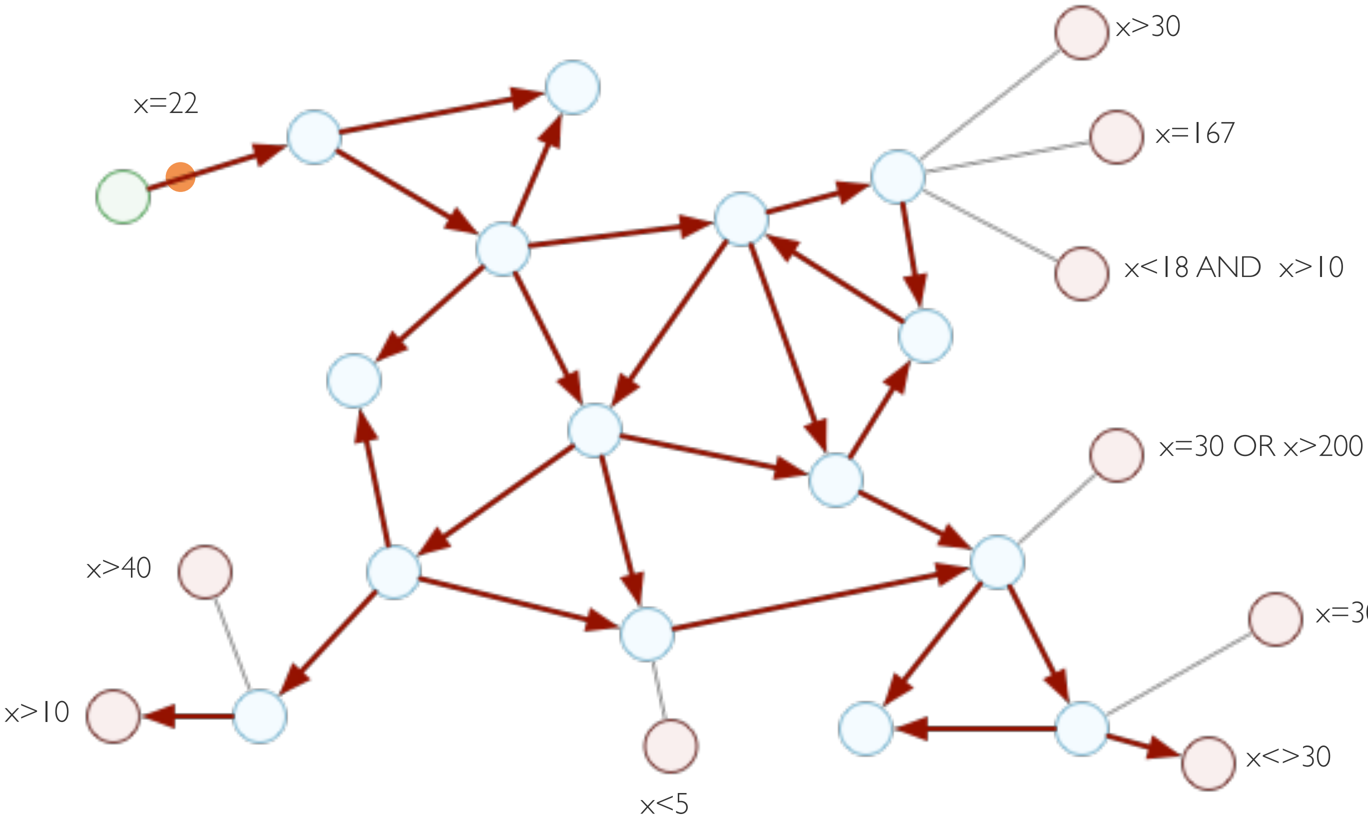
Moreover, the topology of the overlay is obviously unmanaged.

Unstructured overlays use flooding, gossiping or random walks on the overlay graph to diffuse and to retrieve information associated with the nodes.

Implementation issues

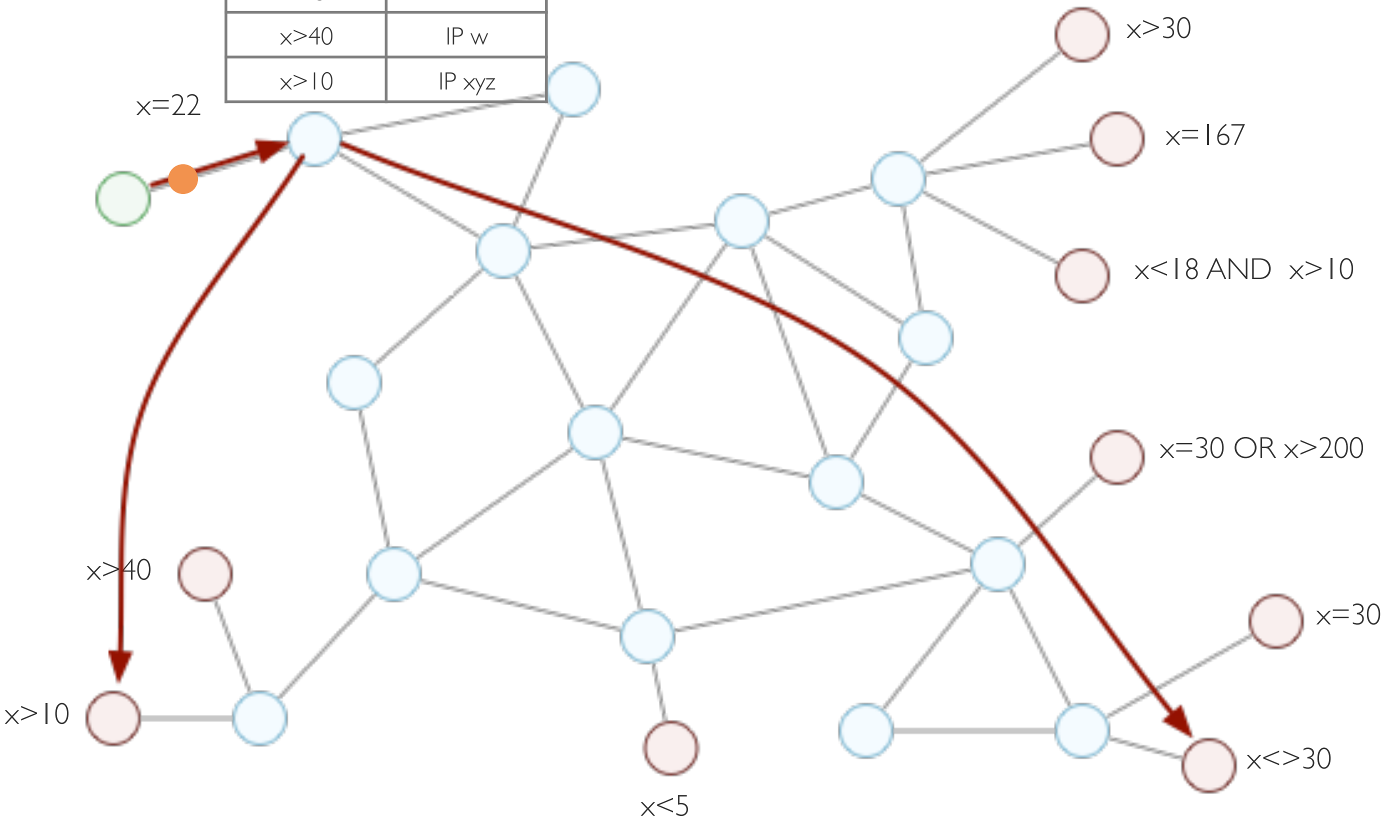
Event Routing

Event flooding

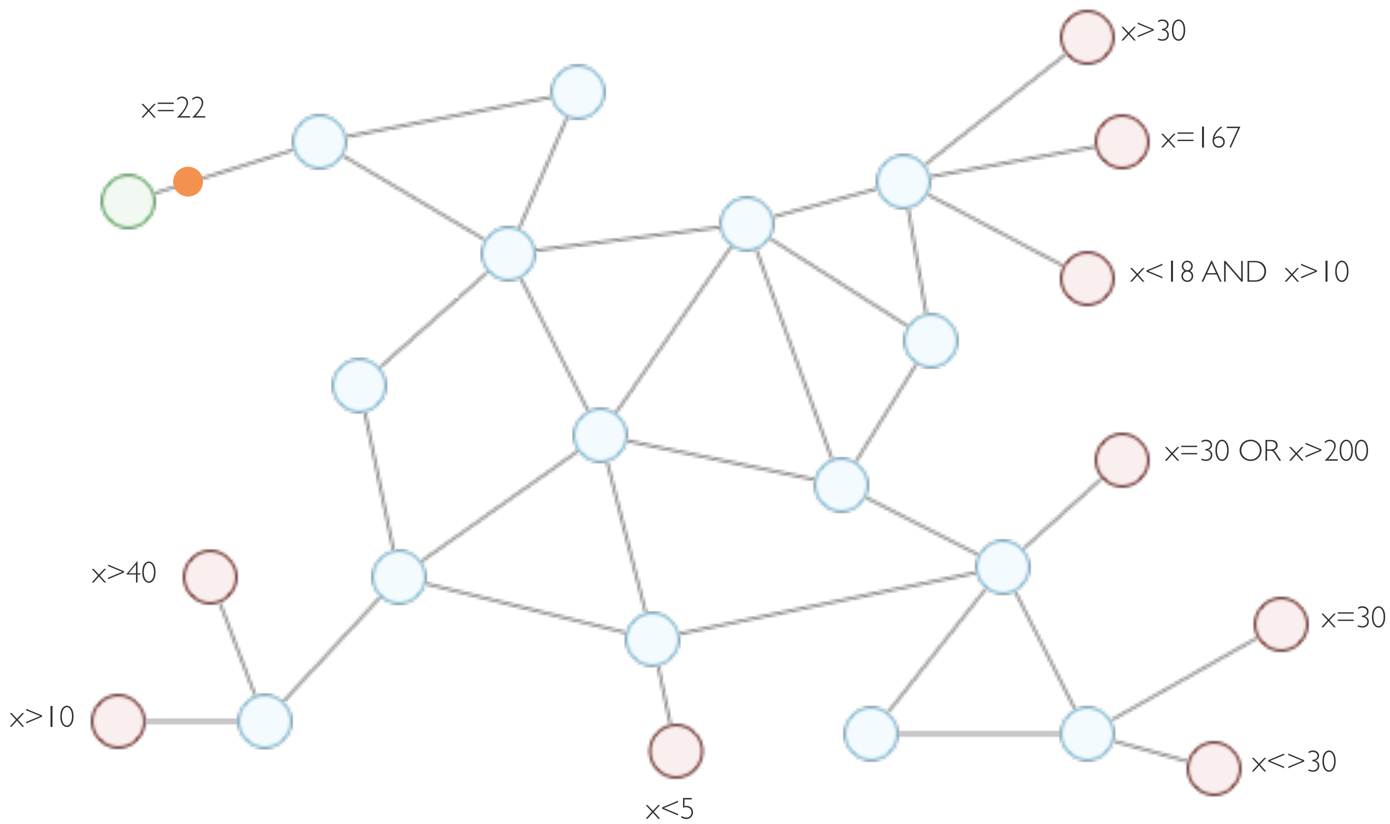


Subscription flooding

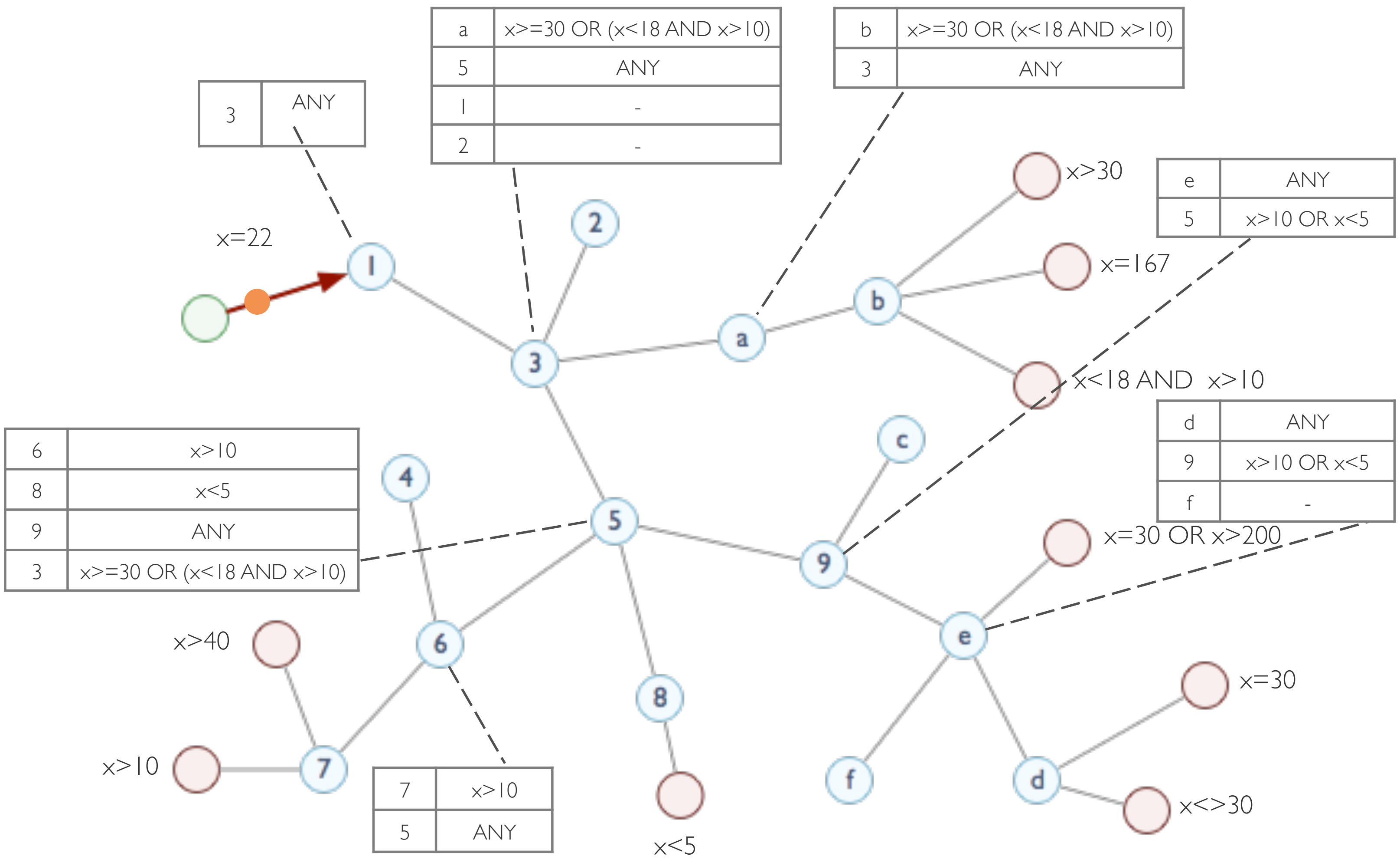
x>30	IP x
x<>30	IP y
x<5	IP z
x>40	IP w
x>10	IP xyz



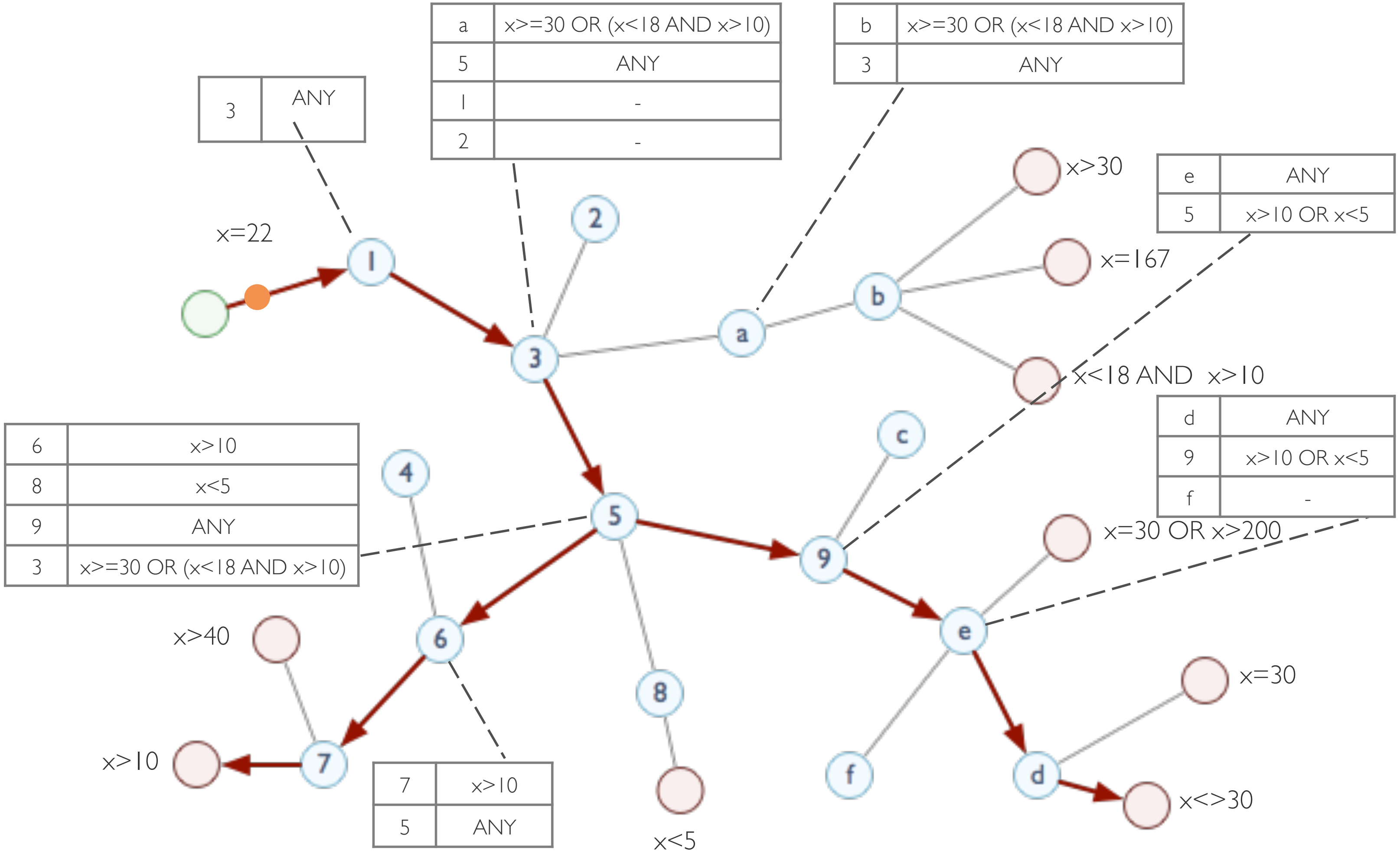
Filter-based routing



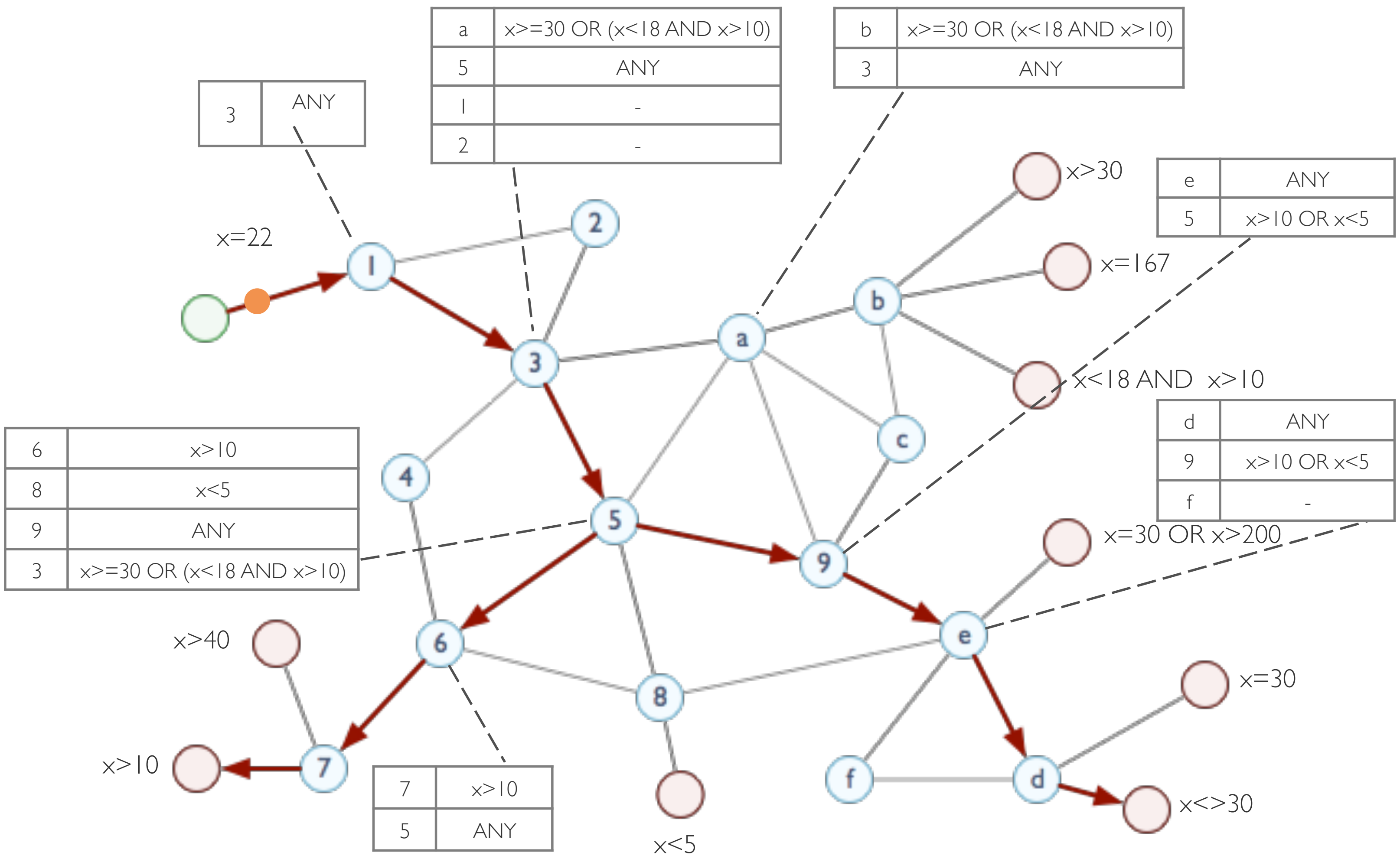
Filter-based routing (cont.)



Filter-based routing (cont.)



Filter-based routing (cont.)



Rendez-Vous routing

It is based on two functions, namely *SN* and *EN*, used to associate respectively subscriptions and events to brokers in the system.

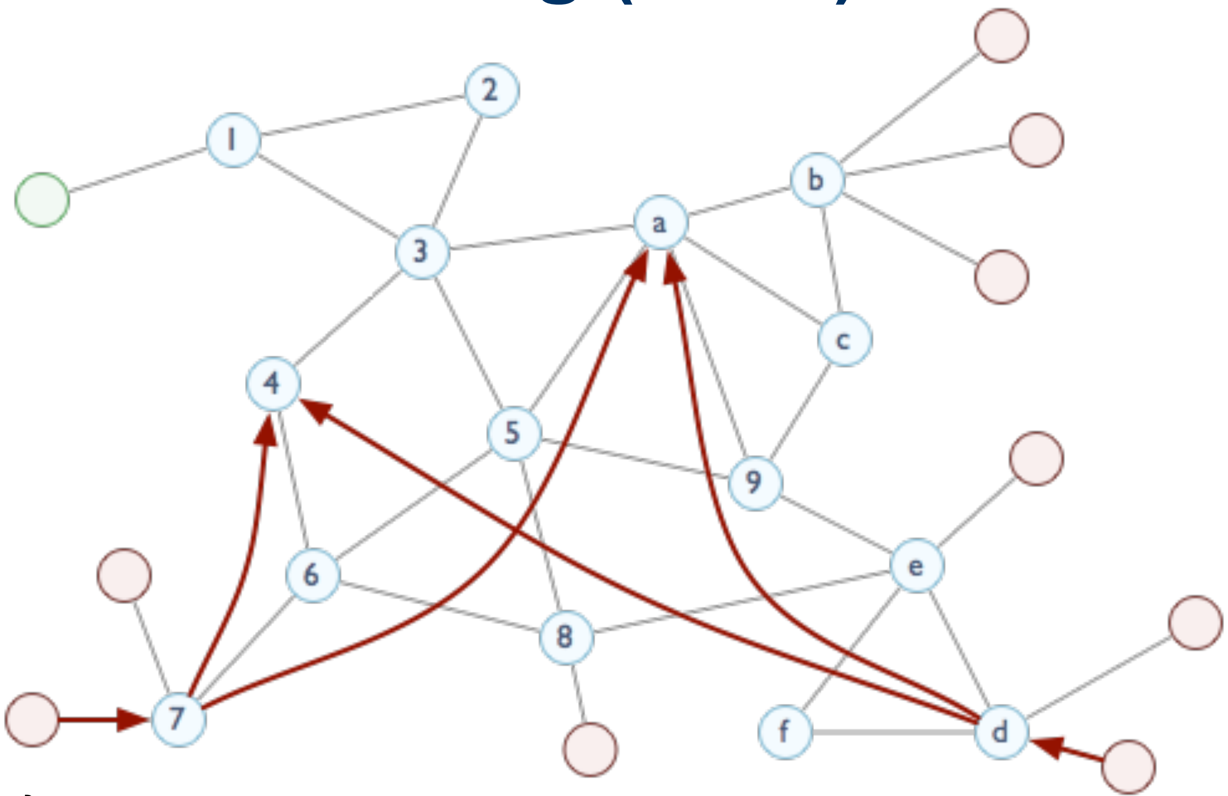
Given a subscription *s*, *SN(s)* returns a set of nodes which are responsible for storing *s* and forwarding received events matching *s* to all those subscribers that subscribed it.

Given an event *e*, *EN(e)* returns a set of nodes which must receive *e* to match it against the subscriptions they store.

Event routing is a two-phases process: first an event *e* is sent to all brokers returned by *EN(e)*, then those brokers match it against the subscriptions they store and notify the corresponding subscribers.

This approach works only if for each subscription *s* and event *e,* such that *e* matches *s*, the intersection between *EN(e)* and *SN(s)* is not empty (*mapping intersection rule*).

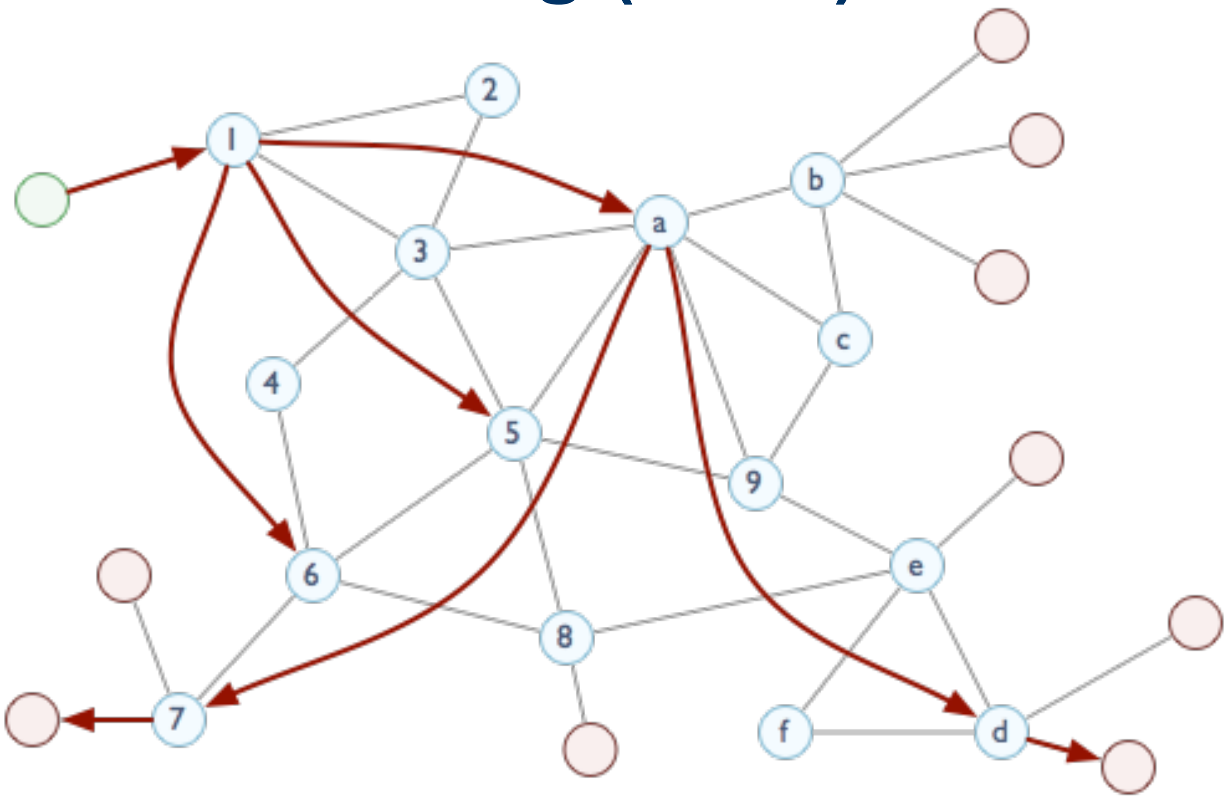
Rendez-Vous routing (cont.)



SN(S) = {4,a}

Phase 1: two nodes issue the same subscription *S*.

Rendez-Vous routing (cont.)



EN(e) = {5,6,a}

Broker a is the rendez-vous point between event *e* and subscription *S*.

Phase 2: an event *e* matching *S* is routed toward the rendez-vous node where it is matched against *S*.

Indirect communication
Message queues

Characteristics

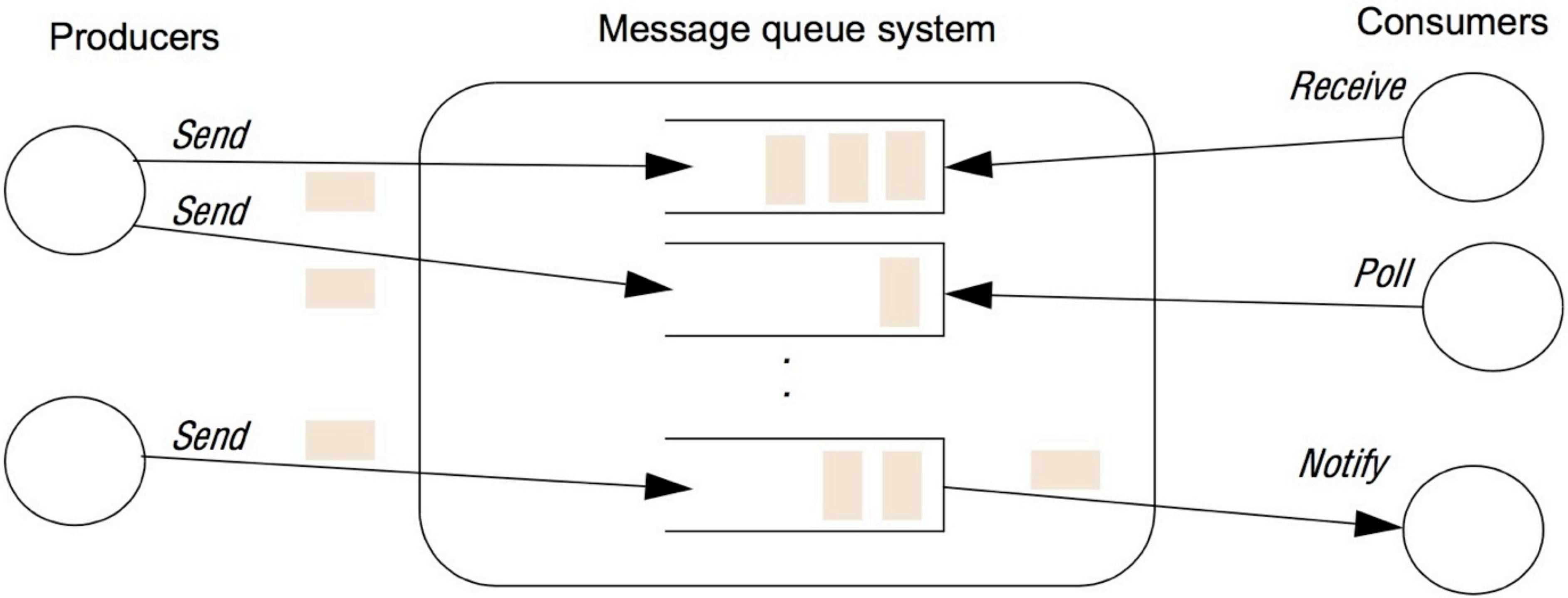
Whereas groups and publish/subscribe systems provide a one-to-many style of communication, messages queues provide a point-to-point service using the concept of a message queue as an indirection.

Message queues are also referred to as Message-Oriented Middleware.

Commercial middleware such as

- IBM WebSphere MQ
- Microsoft MSMQ
- Oracle Stream Advanced Queuing (AQ)

Programming model



Indirect Communication

Summary

So, what have we learned today?

Characteristics of indirect communication (Space and Time decoupling)

Group communication

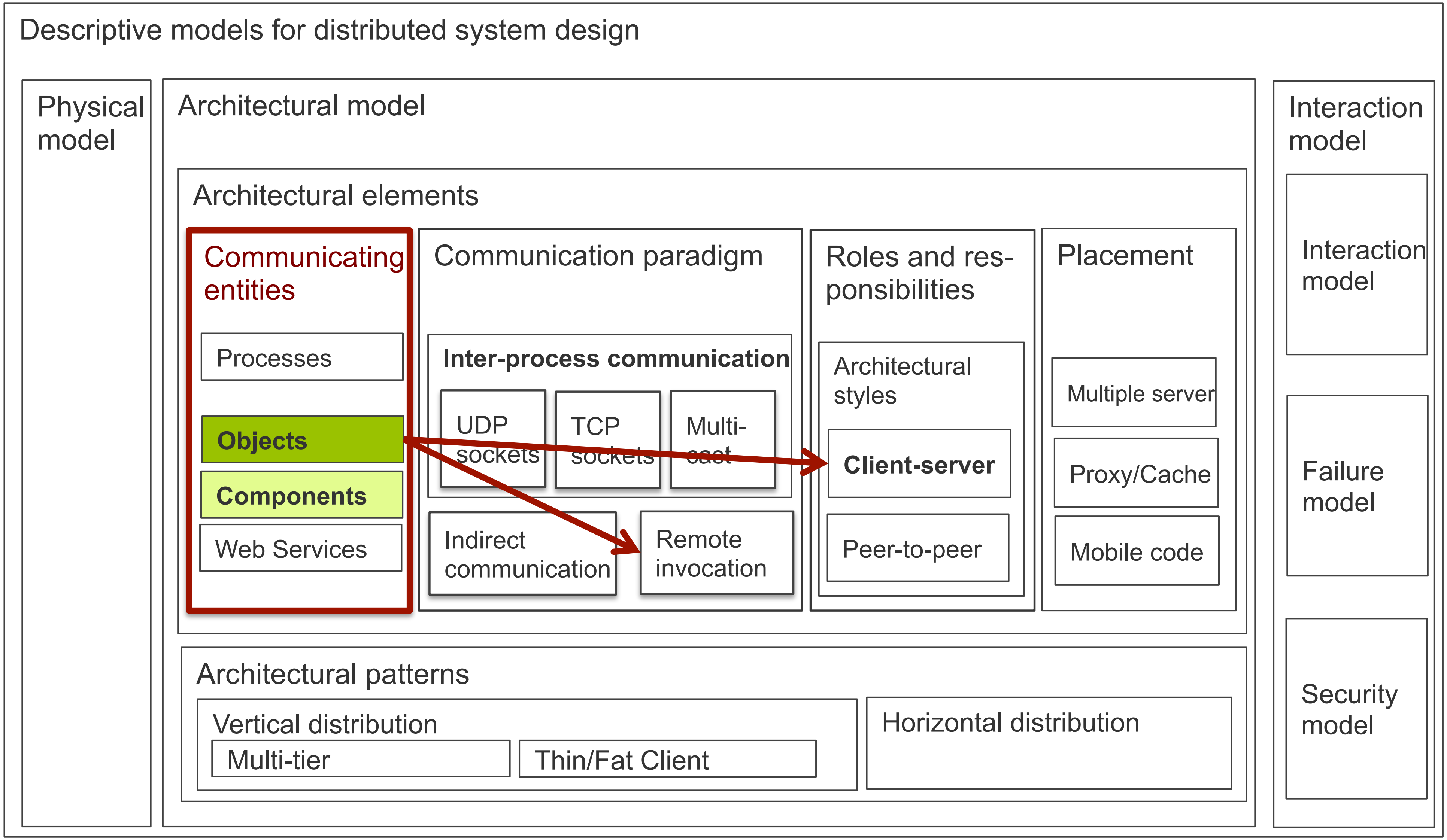
- Applications
- Central concepts and types of groups
- Importance of group membership management

Publish-subscribe systems

- Space, Time and synchronization decoupling
- Subscription models of p/s systems
- The architecture of publish-subscribe systems (overlay infrastructures and event routing)

Message queues

Our topics next week



Next class

Distributed Objects and Components (Case Study: CORBA)

References

George Coulouris, Jean Dollimore, Tim Kindberg: *Distributed Systems: Concepts and Design*. 5th edition, Addison Wesley, 2011.

Patrick Th. Eugster, Pascal A. Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. 2003. The many faces of publish/subscribe. *ACM Comput. Surv.* 35, 2 (June 2003), 114-131. DOI=10.1145/857076.857078 http://doi.acm.org/10.1145/857076.857078

Baldoni, R. & Virgillito, A., 2005. Distributed event routing in publish/subscribe communication systems: a survey. DIS Universita di Roma" La Sapienza" Tech Rep. Available at:
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.106.1108&rep=rep1&type=pdf.

Selected slides from "Distributed Event Routing in Publish/Subscribe Systems", Roberto Baldoni, Sapienza University of Rome, 2009