Publish-Subscribe Systems

George Coulouris, Jean Dollimore, Tim Kindberg and Gordon Blair , "**Distributed Systems Concepts and Design**" Edition 5, © Addison-Wesley 2012

Motivations for Pub/Sub model

- Traditional Client/Server communication model
  - (Employs RPC, message queue, shared memory etc..)
  - Synchronous, tightly-coupled request invocations.
  - Very restrictive for distributed applications, especially for WAN and mobile environments.
  - When nodes/links fail, system is affected. Fault Tolerance must be built in to support this.

- Require a more flexible and de-coupled communication style that offers anonymous and asynchronous mechanisms.

What is a Publish/Subscribe System?

- **Distributed event-based system**
- Distributed Pub/Sub System is a communication paradigm that allows freedom in the distributed system by the decoupling of communication entities in terms of time, space and synchronization.
- An event service system that is asynchronous, anonymous and loosely-coupled.
- Ability to quickly adapt in a dynamic environment.

Key components of Pub/Sub System

- Publishers : Publishers generate event data and publishes them.
- Subscribers : Subscribers submit their subscriptions and process the events received
- P/S service: It's the mediator/broker that filters and routes events from publishers to interested subscribers.
- Task of the publish/subscribe system is to match subscriptions against published events and ensure the correct delivery of event notifications.
- A given event will be delivered to potentially many Subscribers.
- Publish-subscribe is fundamentally a one to-many communications paradigm.

Publish/Subscribe System

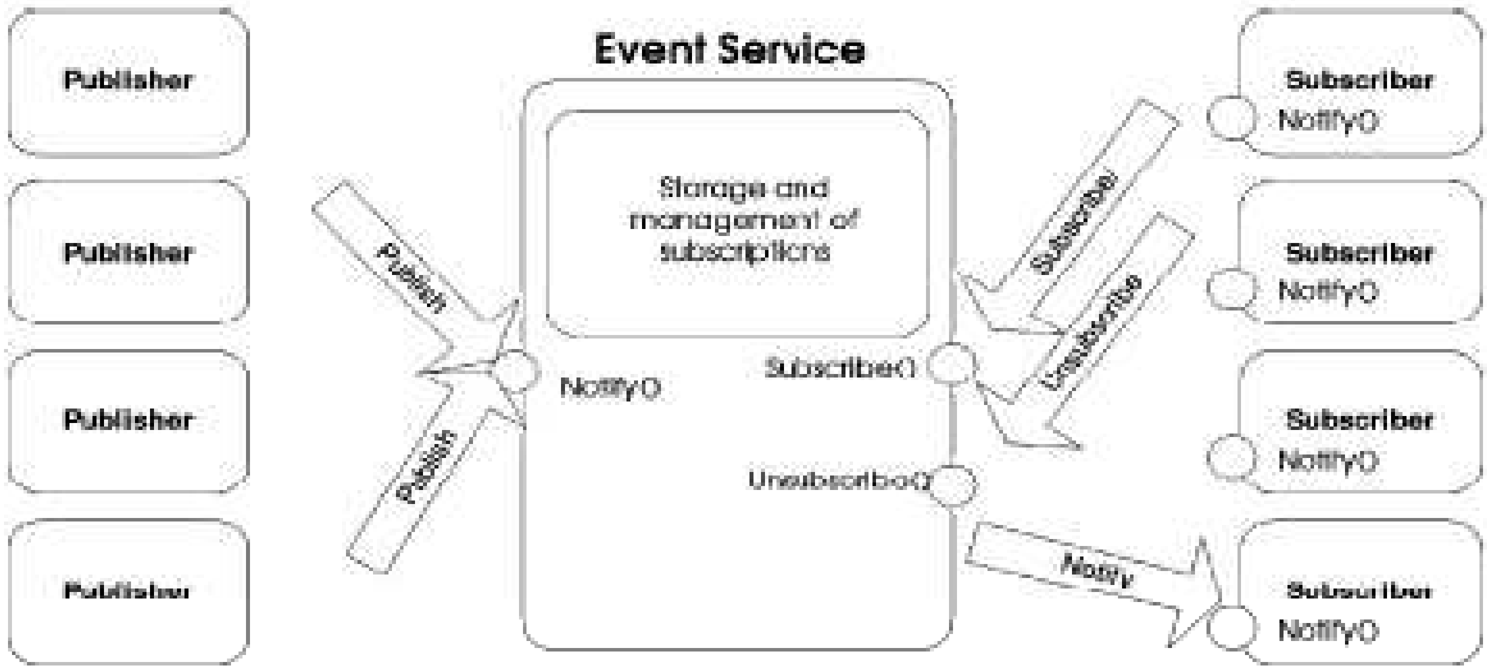


Fig. 1. A simple object-based publish/subscribe system.

Applications

- Financial information systems;
- Other areas with 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 (for example, location events);
- A broad set of monitoring applications, including network monitoring in the Internet.

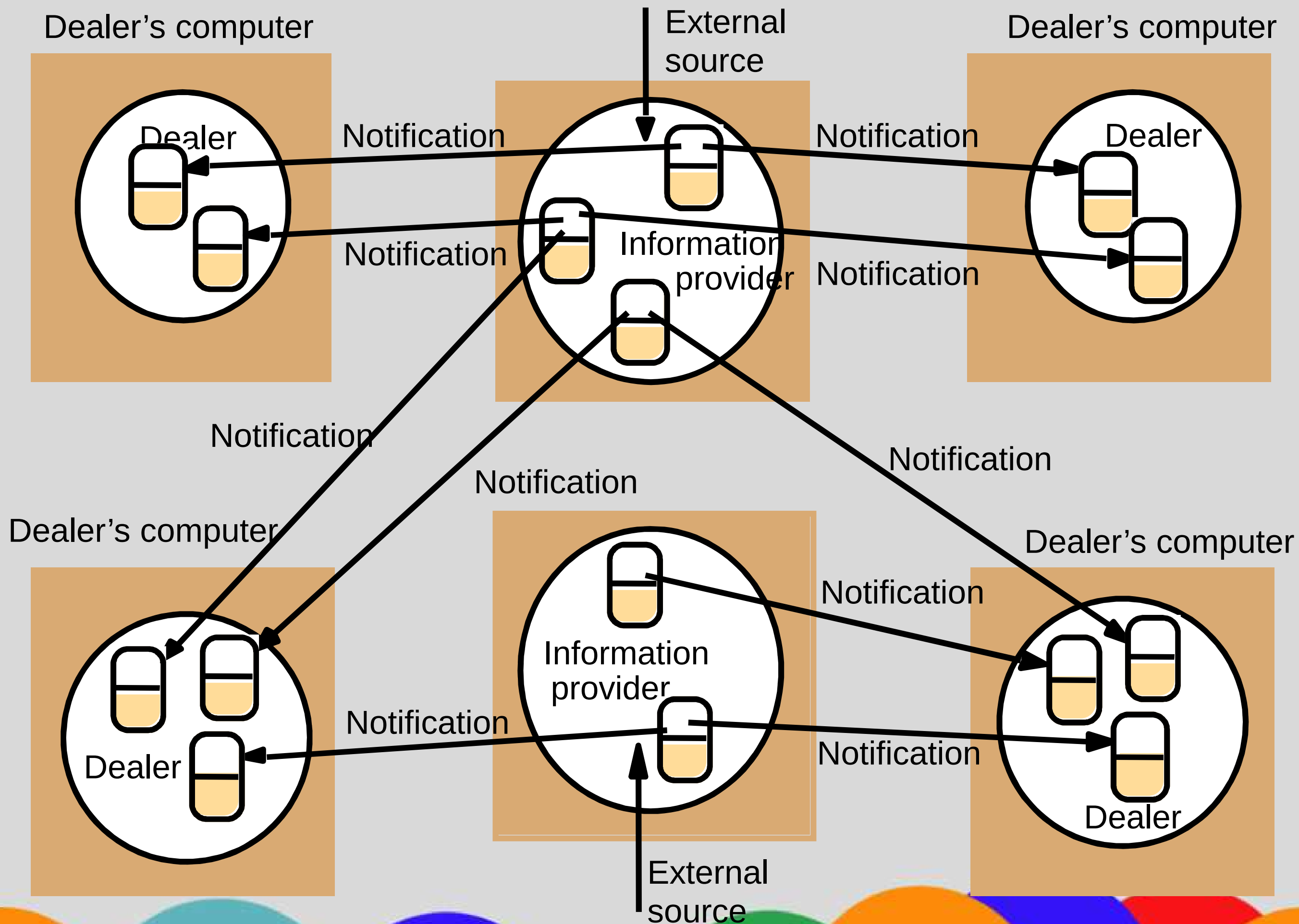
Dealing room system

A dealing room system could be implemented by processes with two different tasks: To know market updates on Stock exchange.

•An **information provider process** continuously receives new trading information from a single external source.
•Each of the updates is regarded as an event.
•The information provider publishes such events to the publish-subscribe system for delivery to all of the dealers who have expressed an interest in the corresponding stock.
•There will be a separate information provider process for each external source.

•A **dealer process** creates a subscription representing each named stock that the user asks to have displayed.
•Each subscription expresses an interest in events related to a given stock at the relevant information provider.
•It then receives all the information sent to it in notifications and displays it to the user.

Dealing room system



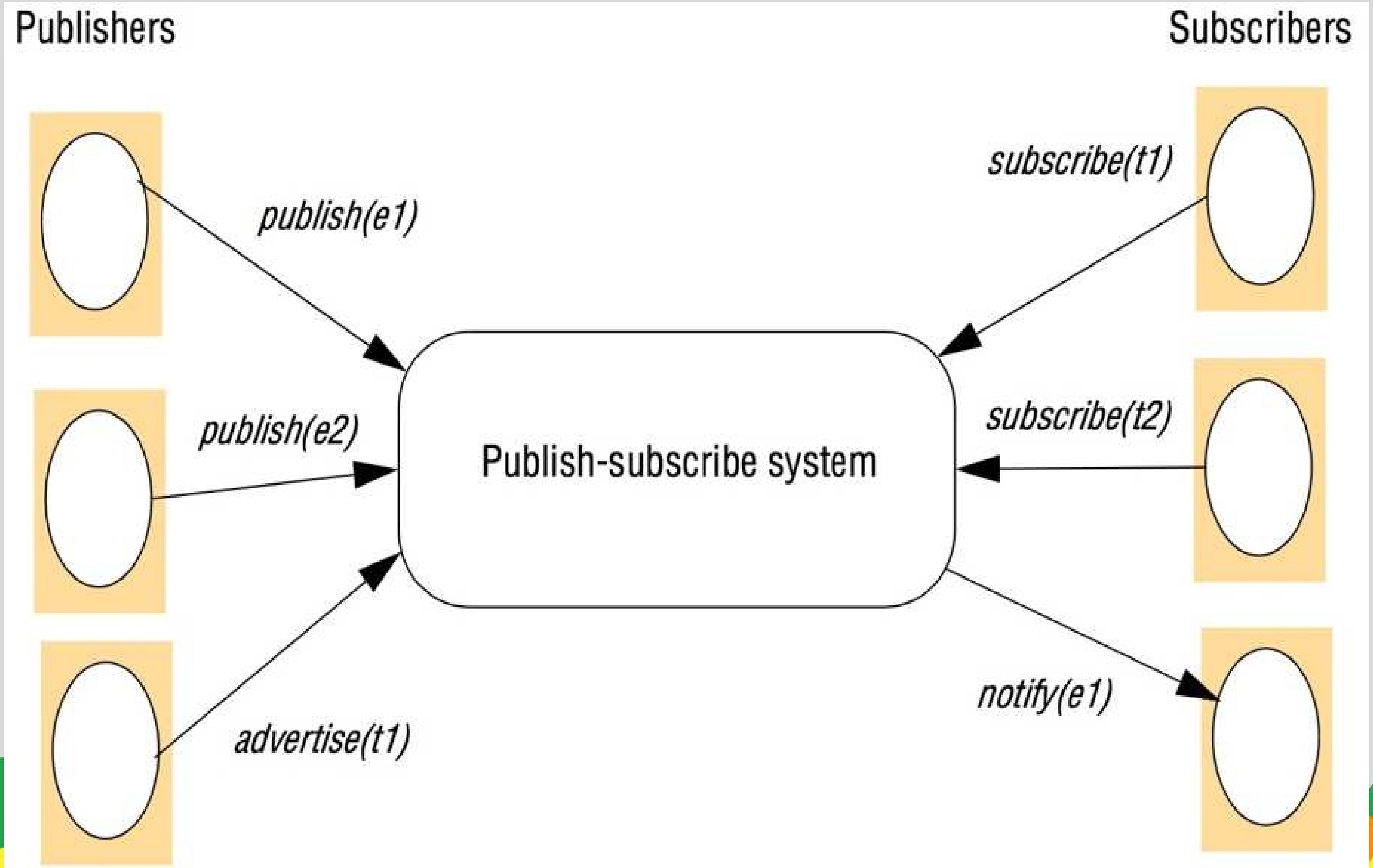
Characteristics of publish-subscribe systems

Publish-subscribe systems have two main characteristics:

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.

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.

The publish-subscribe paradigm



Publish-subscribe systems

The expressiveness of publish-subscribe systems is determined by the subscription (filter) model, with a number of schemes defined and considered in increasing order of sophistication:

Channel-based : publishers publish events to named channels and subscribers then subscribe to one of these named channels to receive all events sent to that channel.

Topic-based (also referred to as subject-based) : each notification is expressed in terms of a number of fields, with one field denoting the topic. Subscriptions are then defined in terms of the topic of interest.

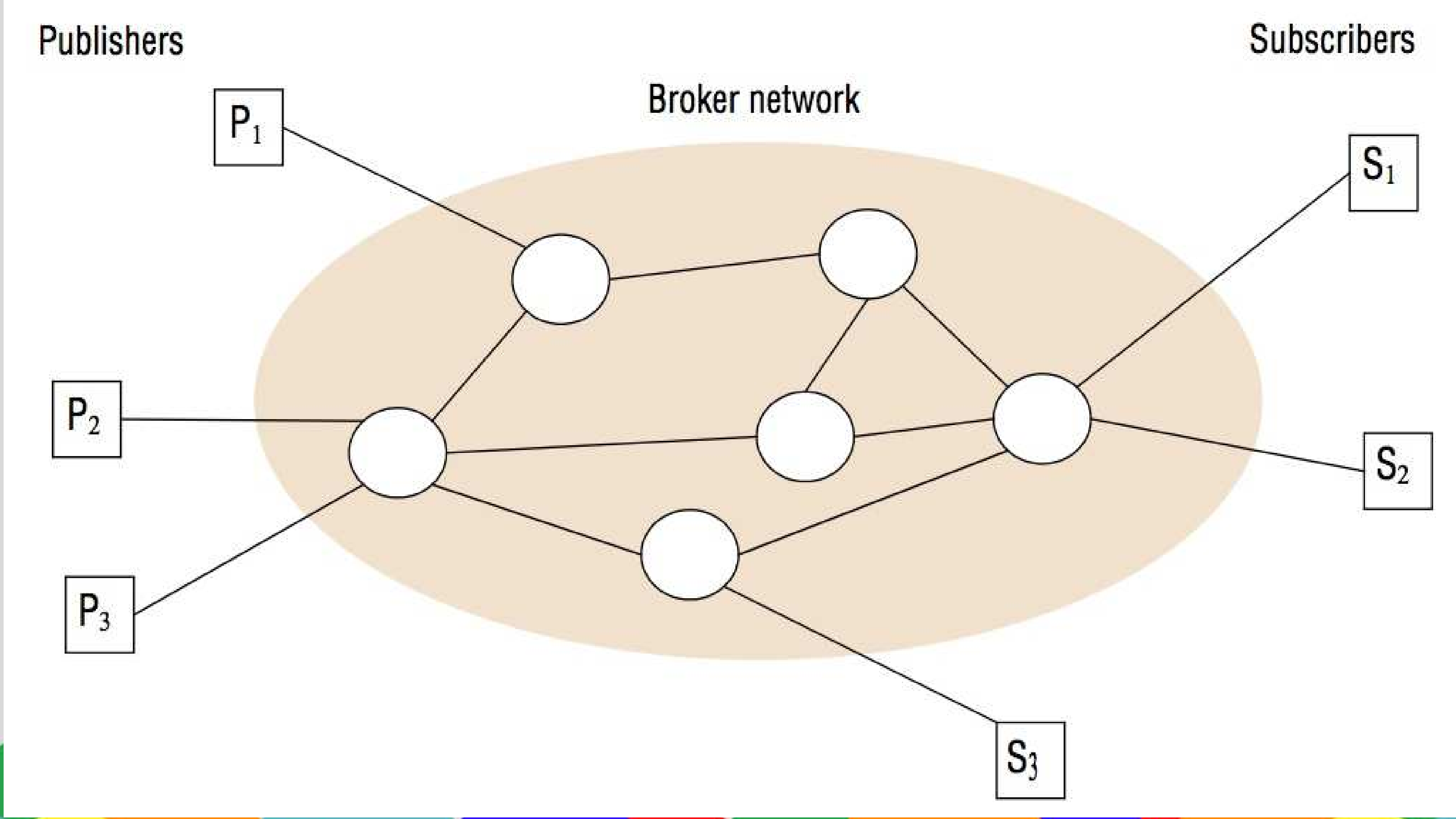
Publish-subscribe systems

Content-based : generalization of topic-based approaches allowing the expression of subscriptions over a range of fields in an event notification.
Type-based :intrinsically linked with object-based approaches where objects have a specified type. In type-based approaches, subscriptions are defined in terms of types of events and matching is defined in terms of types or subtypes of the given filter.

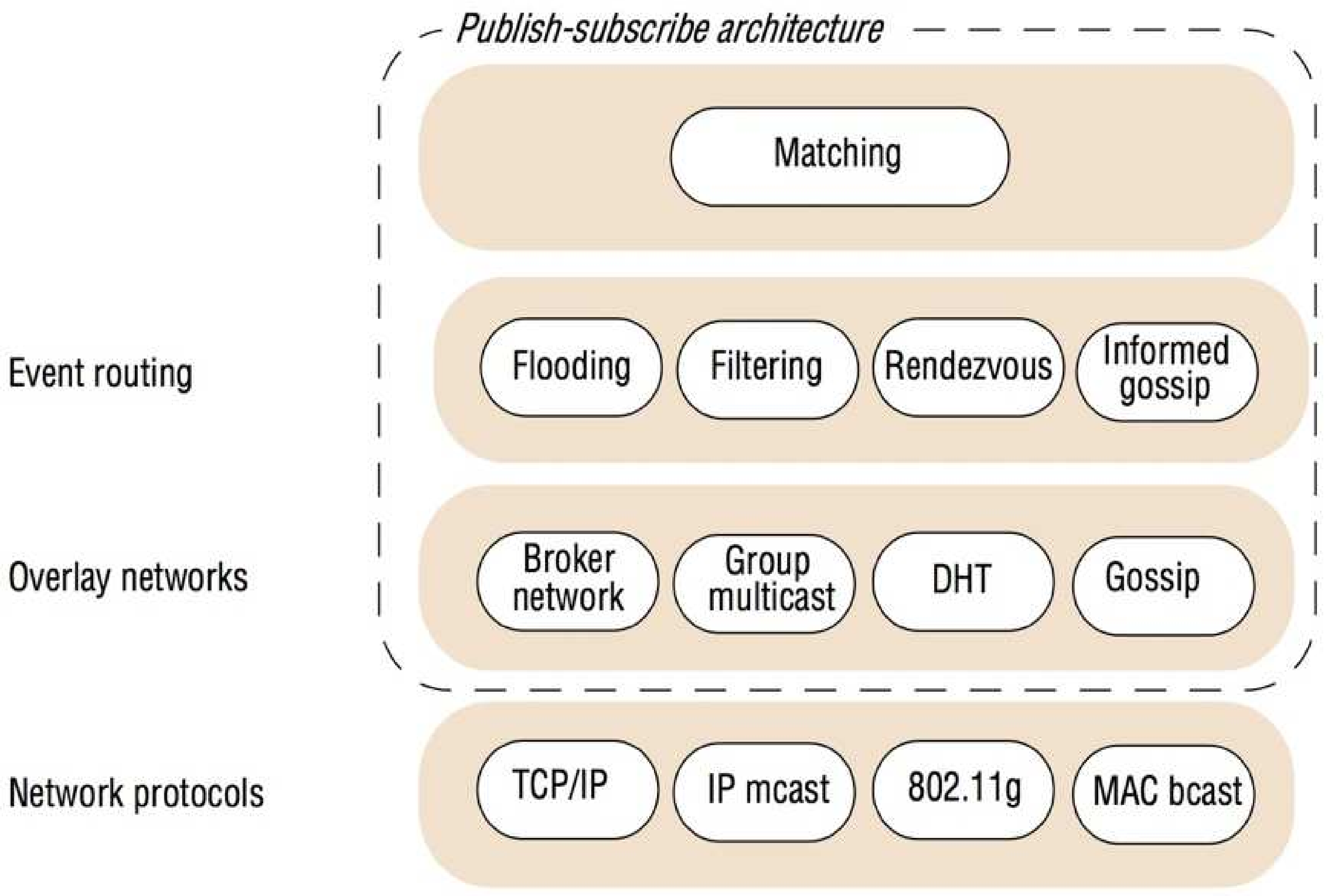
Implementation Issues

- **Centralized Implementation** : A single node with a server on that node acts as an event broker.
  - Publishers then publish events to this broker, subscribers send subscription to the broker and receive notifications in return.
  - Broker is Single point of failure
- **Distributed Implementation:** Centralized broker is replaced by network of brokers that cooperate to offer desired functionality.
- **Peer –to-peer Implementation**:
  - Each node can be publisher, subscriber or broker.
  - Subscribers subscribe to publishers directly and publishers notify subscribers directly. Therefore they must maintain knowledge of each other.

A network of brokers



The architecture of publish-subscribe systems



Architecture

- Event routing performs the task of ensuring that event notifications are routed as efficiently as possible to appropriate subscribers.
- The overlay infrastructure supports this by setting up appropriate networks of brokers or peer-to-peer structures.
- **Content-based routing (CBR**), with the goal being to exploit content information to efficiently route events to their required destination

Flooding

- In Flooding: Sending an event notification to all nodes in the network and then carrying out the appropriate matching at the subscriber end.
- As an alternative, flooding can be used to send subscriptions back to all possible publishers, with the matching carried out at the publishing end.
- Matched events sent directly to the relevant subscribers using point-to-point communication.

Filtering-based routing

- Brokers forward notifications through the network only where there is a path to a valid subscriber.
- Each node must maintain a neighbors list containing a list of all connected neighbors in the network of brokers.
- A subscription list containing a list of all directly connected subscribers serviced by this node, and a routing table.
- Routing table maintains a list of neighbors and valid subscriptions for that pathway.
- It requires an implementation of matching on each node in the network of brokers

Filtering-based routing

upon receive *publish(event e)* ***from*** *node x*
    matchlist := match(e, subscriptions)
    send notify(e) to matchlist;
    fwdlist := match(e, routing);
    send *publish(e) to fwdlist - x;*
upon receive *subscribe(subscription s)* ***from*** *node x*
    if *x is client* ***then***
        add x to subscriptions;
    else *add(x, s) to routing;*
    send *subscribe(s) to neighbours - x;*

Filtering-based routing

- When a broker receives a publish request from a given node,
  - It must pass this notification to all connected nodes where there is a corresponding matching subscription.
  - Decide where to propagate this event through the network of Brokers.
- Matching the event against the subscription list and then forwarding the event to all the nodes with matching subscriptions (the *matchlist) (line 2,3)*
- Matching the event against the routing table and forwarding only to the paths that lead to a subscription (the *fwdlist). (line 4,5)*
- Brokers must also deal with incoming subscription events. If the subscription event is from an immediately connected subscriber, then this subscription must be entered in the subscriptions table. (Line 7,8)
- Otherwise, the broker is an intermediary node; this node now knows that a pathway exists towards this subscription and hence an appropriate entry is added to the routing table (line 9).
- In both cases, this subscription event is then passed to all neighbors apart from the originating node (line 10).

Advertisements

- The pure filtering-based approach described above can generate a lot of traffic due to propagation of subscriptions.
- In advertisements this burden can be reduced by propagating the advertisements towards subscribers in a similar (actually symmetrical) way to the propagation of subscriptions.

Rendezvous-based Routing

- This view the set of all possible events as an event space and to partition responsibility for this event space between the set of brokers in the Network
- This approach defines rendezvous nodes, which are broker nodes responsible for a given subset of the event space

Rendezvous-based Routing

- First, SN(s) takes a given subscription, s, and returns one or more rendezvous nodes that take responsibility for that subscription.
- Each such rendezvous node maintains a subscription list and forwards all matching events to the set of subscribing nodes

Rendezvous-based Routing

- Second, when an event e is published, the function EN(e) also returns one or more rendezvous nodes, this time responsible for matching e against subscriptions in the system
- Both SN(s) and EN(e) return more than one node if reliability is a concern.
- Note also that this approach only works if the intersection of EN(e) and SN(s) is non-empty for a given e that matches s

Rendezvous-based routing

```
upon receive publish(event e) from node x at node i
    rvlist := EN(e);
    if i in rvlist then begin
        matchlist :=match(e, subscriptions);
        send notify(e) to matchlist;
    end
    send publish(e) to rvlist - i;
upon receive subscribe(subscription s) from node x at node i
    rvlist := SN(s);
    if i in rvlist then
        add s to subscriptions;
    else
        send subscribe(s) to rvlist - i;
```

Gossip

- Gossip-based approaches are a popular mechanism for achieving multicast (including reliable multicast)
- They operate by nodes in the network periodically and probabilistically exchanging events (or data) with neighboring nodes.
- It is possible to take into account local information and, in particular, content to achieve what is referred to as informed gossip

Example publish-subscribe system

System (and further reading)	*Subscription model*	*Distribution model*	*Event routing*
CORBA Event Service (Chapter 8)	Channel-based	Centralized	-
TIB Rendezvouz [Oki *et al.* 1993]	Topic-based	Distributed	Ffiltering
Scribe [Castro *et al.* 2002b]	Topic-based	Peer-to-peer (DHT)	Rendezvous
TERA [Baldoni *et al.* 2007]	Topic-based	Peer-to-peer	Informed gossip
Siena [Carzaniga *et al.* 2001]	Content-based	Distributed	Filtering
Gryphon [www.research.ibm.com]	Content-based	Distributed	Filtering
Hermes [Pietzuch and Bacon 2002]	Topic- and content-based	Distributed	Rendezvous and filtering
MEDYM [Cao and Singh 2005]	Content-based	Distributed	Flooding
Meghdoot [Gupta *et al.* 2004]	Content-based	Peer-to-peer	Rendezvous
Structure-less CBR [Baldoni *et al.* 2005]	Content-based	Peer-to-peer	Informed gossip

Thank You