Chapter 1
Distributed Event Routing in Publish/Subscribe Communication Systems

Roberto Baldoni, Dipartimento di Informatica e Sistemistica, Sapienza Università di Roma
Leonardo Querzoni, Dipartimento di Informatica e Sistemistica, Sapienza Università di Roma
Sasu Tarkoma, Helsinki University of Technology and Nokia NRC
Antonino Virgillito, Dipartimento di Informatica e Sistemistica, Sapienza Università di Roma

1.1 Introduction

Since the early nineties, anonymous and asynchronous dissemination of information has been a basic building block for typical distributed application such as stock exchanges, news tickers and air-traffic control. With the advent of ubiquitous computing and of the ambient intelligence, information dissemination solutions have to face challenges such as the exchange of huge amounts of information, large and dynamic number of participants possibly deployed over a large network (e.g. peer-to-peer systems), mobility and scarcity of resources (e.g. mobile ad-hoc and sensor networks) [9].

Publish/Subscribe (pub/sub) systems are a key technology for information dissemination. Each participant in a pub/sub communication system can take on the role of a publisher or a subscriber of information. Publishers produce information in form of *events*, which is consumed by subscribers issuing *subscriptions* representing their interest only in specific events. The main semantical characterization of pub/sub is in the way events flow from senders to receivers: receivers are not directly targeted from publisher, but rather they are indirectly addressed according to the content of events. Thanks to this anonymity, publishers and subscribers exchange information without directly knowing each other, this enabling the possibility for the system to seamlessly expand to massive, Internet-scale size.

Interaction between publishers and subscribers is actually mediated by the pub/sub system, that in general is constituted by a set of distributed nodes that coordinate among themselves in order to dispatch published events to all (and possibly only) interested subscribers. A distributed pub/sub system for scalable information dissemination can be decomposed in three functional layers: namely the overlay infrastructure, the event routing and the algorithm for matching events against subscriptions. The overlay infrastructure represents the organization of the various entities that compose the system, (e.g., overlay network of dedicated servers, peer-to-peer structured overlay, etc.) while event routing is the mechanism for dispatching information from publishers to subscribers. Event routing has to effectively exploit

the overlay infrastructure and enhance it with routing information in order to achieve scalable event dispatching. Several research contributions have appeared in the last years proposing pub/sub solutions in which these functionalities are not sharply separated and their dependencies have not been clearly pointed out. This makes all the different proposals difficult to fully understand and compare among each other.

This chapter is aimed at giving the reader a structured overview of current solutions for event routing in distributed pub/sub systems. It firstly introduces a general pub/sub architectural model for scalable information dissemination, that decomposes a generic pub/sub system into the three layers identified above. Then the chapter focuses on the solutions proposed in the literature for event routing and discuss their relations with the overlay network level solutions and possible network deployments. As a result any specific pub/sub system can be easily characterized as a stack of solutions available at each layer. Specifically the chapter categorizes event routing algorithms into six classes, each of which corresponds to a basic general dispatching method. These classes are discussed according to their underlying assumptions in terms of aspects such as the induced message overhead, routing information required at each node, dependency from the subscription language, adaptivity to dynamic changes of the underlying network. Moreover we specify how algorithms in each class relate to the overlay network layer, in particular pointing out which overlay infrastructure is more suitable for a specific event routing solution and why. In the final part of the chapter we discuss specific issues related to the deployment of pub/sub systems in mobile environments.

Being focused on scalable event routing and its relation with the underlying overlay network, this chapter complements other surveys related to publish/subscribe systems such as [43], [52], [63] and [14]. The main aim of [43] has been indeed to position the pub/sub paradigm with respect to other communication paradigms, whereas [52] concentrated on software engineering aspects of a pub/sub system. Liu and Plale in [63] propose a survey of pub/sub systems considering overlay topology, matching algorithms and aspects such as reliability and security.

This chapter is structured as follows. Section 1.2 presents a general overview of the pub/sub paradigm. Section 1.3 describes the various types of subscription models. Section 1.4 presents the architectural pub/sub model, discussing all the four layers. In Section 1.5 we focus on the event routing layer. Section 1.6 discusses issues related to security in pub/sub systems. Section 1.7 consider mobility support in pub/sub systems, and finally Section 1.8 concludes this chapter.

1.2 Elements of a Publish/Subscribe System

A generic pub/sub communication system (often referred to in the literature as *Event Service* or *Notification Service*) is composed of a set of nodes distributed over a communication network. Clients to the systems are divided according to their role into publishers, which act as producers of information, and subscribers, which act as consumers of information. Clients are not required to communicate directly among

themselves but they are rather *decoupled*: the interaction takes place through the nodes of the pub/sub system. This decoupling is a desirable characteristic for a communication system because applications can be made more independent from the communication issues, avoiding to deal with aspects such as synchronization or direct addressing of subscribers from publishers[1].

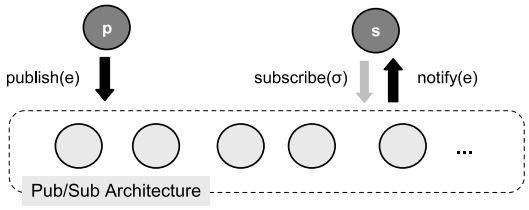


Fig. 1.1 High-level interaction model of a publish/subscribe system with its clients (*p* and *s* indicate a generic publisher and a generic subscriber respectively).

Operationally, the interaction between client nodes and the pub/sub system takes place through a set of basic operations that can be executed by clients on the pub/sub system and viceversa (Figure 1.1). A publisher submits a piece of information *e* (i.e., an event) to the pub/sub system by executing the `publish(`e`)` operation. Commonly, an event is structured as a set of attribute-value pairs. Each attribute has a *name*, a simple character string, and a *type*. The type is generally one of the common primitive data types defined in programming languages or query languages (e.g. integer, real, string, etc.). On the subscribers' side, interest in specific events is expressed through *subscriptions*. A subscription σ, is a filter over a portion of the event content (or the whole of it), expressed through a set of constraints that depend on the subscription language. A subscriber installs and removes a subscription σ from the pub/sub system by executing the `subscribe(`σ`)` and `unsubscribe(`σ`)` operations respectively.

We say a notification *e matches* a subscription σ if it satisfies all the declared constraints on the corresponding attributes. The task of verifying whenever a notification *e* matches a filter f is called *matching* ($e \sqsubset f$).

1.3 Subscription Models

Various ways for specifying the events of interest have led to identifying distinct variants of the pub/sub paradigm. The subscription models that appeared in the literature are characterized by their expressive power: highly expressive models offer to subscribers the possibility to precisely match their interest, i.e. receiving only

[1] The interested reader can refer to [43] for a more deep discussion on the publish/subscribe paradigm and subscription models.

the events they are interested in. In this section we briefly review the most popular pub/sub subscription models.

Topic-based Model

Notifications are grouped in topics i.e., a subscriber declares its interest for a particular topic and will receive all events related to that topic. Each topic corresponds to a logical channel ideally connecting each possible publisher to all interested subscribers. For the sake of completeness, the difference between channel and topics is that topics are carried within an event as a special attribute. Thanks to this coarse grain correspondence, either network multicast facitilies or diffusion trees, one for each topic, can be used to disseminate events to interested subscribers.

Topic-based model has been the solution adopted in all early pub/sub incarnations. Examples of systems that fall under this category are TIB/RV [71], iBus [2], SCRIBE [26], Bayeux [109] and the CORBA Notification Service [70].

The main drawback of the topic-based model is the very limited expressiveness it offers to subscribers. A subscriber interested in a subset of events related to a specific topic receives also all the other events that belong to the same topic. To address problems related to low expressiveness of topics, several solutions are exploited in pub/sub implementations. For example, the topic-based model is often extended to provide hierarchical organization of the topic space, instead of a simple flat structure (such as in [5, 71]). A topic B can be then defined as a sub-topic of an existing topic A. Events matching B will be received by all clients subscribed to both A and B. Implementations also often include convenience operators, such as wildcard characters, for subscribing to more than one topic with a single subscription. For the sake of completeness, we point out that the word *subject* can be used to refer to hierarchical topics instead of being simply a synonymous for topic. Analogously, *channel-based* is sometimes [70] used to refer to a flat topic model where the topic name is not explicitly included in the event.

Content-based Model

Subscribers express their interest by specifying conditions over the content of notifications they want to receive. In other words, a filter in a subscription is a query formed by a set of constraints over the values of attributes of the notification composed through disjunction or conjunction operators. Possible constraints depend on the attribute type and on the subscription language. Most subscription languages comprise equality and comparison operators as well as regular expressions [23, 90, 49]. The complexity of the subscription language obviously influences the complexity of matching operation. Then it is not common to have subscription languages allowing queries more complex than those in conjunctive form (examples are [18, 15]). A complete specification of content-based subscription models can be found in [66]. Examples of systems that fall under the content-based category are

Gryphon [55], SIENA [92], JEDI [38], LeSubscribe [81], Ready [54], Hermes [79], Elvin [89].

In content-based publish/subscribe, events are not classified according to some predefined criterion (i.e., topic name), but rather according to properties of the events themselves. As a consequence, the correspondence between publishers and subscribers is on an event basis. Then, the higher expressive power of content-based pub/sub comes at the price of the higher resource consumption needed to calculate for each event the set of interested subscribers [22, 42]. It is straightforward to see that a topic-based scheme can be emulated through a content-based one, simply considering filters comprising a single equality constraint.

Type-based

The type-based [44, 46] pub/sub variant events are actually objects belonging to a specific type, which can thus encapsulate attributes as well as methods. With respect to simple, unstructured models, Types represent a more robust data model for application developer, enforcing type-safety at the pub/sub system, rather than inside the application [48]. In a type-based subscription the declaration of a desired type is the main discriminating attribute. That is, with respect to the aforementioned models, type-based pub/sub sits itself somehow in the middle, by giving a coarse-grained structure on events (like in topic-based) on which fine-grained constraints can be expressed over attributes (like in content-based) or over methods (as a consequence of the object-oriented approach).

Concept-based

The underlying implicit assumptions within all the above-mentioned subscription models is that participants have to be aware of the structure of produced events, both under a syntactic (i.e., the number, name and type of attributes) and a semantic (i.e., the meaning of each attribute) point of view. Concept-based addressing [31] allows 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

Some research works [27, 28, 93] describe pub/sub systems supporting a semistructured data model, typically based on XML documents. XML is not merely a matter of representation but differs in the fact that introduces the possibility of hierarchies in the language, thus differentiating from a flat content-based model in terms of an added flexibility. Moreover, it provides natural advantages such as interoperability,

independence from implementation and extensibility. As a main drawback, matching algorithms for XML-based language requires heavier processing.

Location-awareness

Pub/Sub systems used in mobile environments typically require the support for location-aware subscriptions. For example, a mobile subscriber can query the system for receiving notifications when it is in the proximity of a specific location or service. Works describing various forms of location-aware subscriptions are [64, 50, 36, 93]. The implementation of location-aware subscriptions requires the pub/sub system the ability to monitor the mobility of clients.

1.4 Architectural Model

In this section we describe the reference architectural model we use in our presentation. The architectural model is depicted in Figure 1.2, including four logical layer, namely *Network Infrastructure*, *Overlay Infrastructure*, *Event Routing* and *Matching*. We present in the following the functionality associated to each layer as well as the different possible solutions for its realization (also illustrated in the figure).

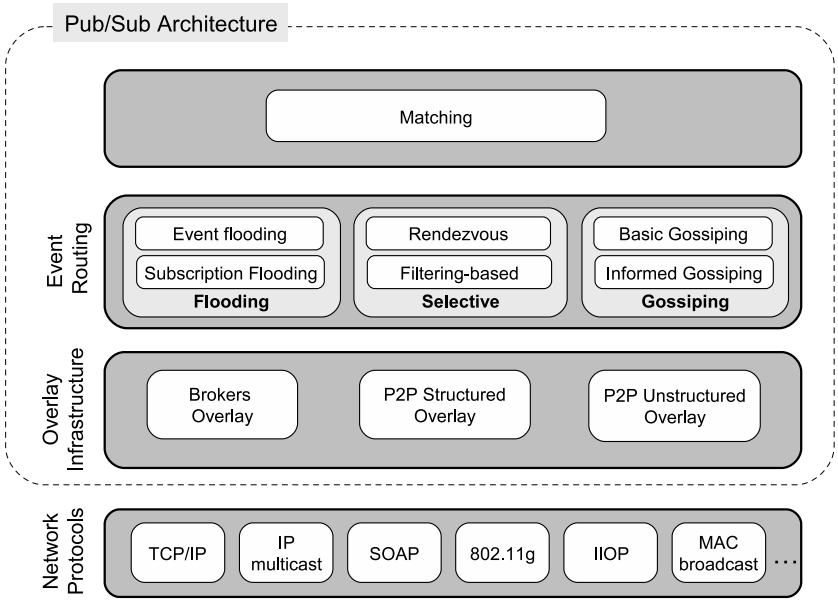


Fig. 1.2 Publish/Subscribe Architectural Model

In Table 1.1 we report how different pub/sub systems present in the current literature implement the proposed architectural model. For each system we tried to abstract its main characteristic in order to let them fit our model. Note, however, that this Table is not meant to be complete as it only includes a limited subset of all existing pub/sub solutions, and for those presented here, it avoids to detail many important technical aspects that characterize and differentiate each single system.

System	*Subscription Model*	*Network Protocol*	*Overlay Infrastructure*	*Event Routing*
TIB/RV[71]	Topic	TCP & MAC bcast	Brokers	Filtering
Scribe[26], CAN[82]	Topic	TCP	P2P Structured	Rendezvous
Siena[23], Gryphon[11], Rebeca[99]	Content	TCP/UDP	Brokers	Filtering
Hermes[79]	Content	TCP	Brokers	RVs/Filtering
Meghdoot[57]	Content	TCP	P2P Structured	Rendezvous
DADI (Kyra[19])	Content	TCP	Brokers	RVs/Filtering
DADI (MEDYM[20])	Content	TCP & IP mcast	Brokers	Sub. Flooding
GREEN (WAN)[93]	Various	TCP	P2P Structured	Rendezvous
GREEN (Mobile)[93]	Various	802.11g	Unstructured	Gossiping
Sub-2-Sub[101]	Content	UDP	P2P Structured & Unstructured	Informed Gossiping
Spidercast[30]	Topic	UDP	P2P Unstructured	Gossiping
TERA[8]	Topic	UDP	P2P Unstructured	Informed Gossiping

Table 1.1 Pub/Sub Systems

1.4.1 Network Protocols

Network protocols anchor a pub/sub system to the underlying network by allowing transmission of data among pub/sub system components. Due to the fact that a pub/sub system could span over heterogeneous networks (e.g., LANs, WANs, mobile networks, etc.), it could employ more than a single network protocol either to cope with different software/hardware condition that could be found in a given part of the network or to maximize performance. For example a pub/sub system deployed over a WAN could use MAC broadcast inside a LAN to reach in one shot all recipients of an events while sending evens between two LANs using TCP connections.

Transport level

Pub/sub systems are usually built exploiting the functionality of common transport-level protocols. That is, nodes in the overlay infrastructure communicate directly through TCP or UDP sockets or using specific TCP-based middleware protocols (like IIOP or SOAP). This choice allows the greater flexibility and ease of deployment, though in such situations, the deployment over a wide-area network can be limited by the presence of network firewalls or private networks, requiring the intervention of an administrator for configuration.

Network-level Multicast

Directly exploiting local-area or wide-area multicast and broadcast network primitives is an efficient way to realize many-to-many diffusion experiencing low latencies and high throughput, thanks to the small delays introduced by implementing the protocols exclusively involving routers and switches. For example, IP Multicasting can be directly used in wide-area topic-based systems, as each topic corresponds exactly to one multicast group. Using IP multicast for content-based systems is not as straightforward because subscribers cannot be directly mapped to multicast groups. This has inspired some research work targeting at organizing subscribers in clusters, where subscribers in the same cluster contain most of the subscriptions in common [72, 84, 85, 56]. The main drawback of IP multicast is in its lack of a widespread deployment [41, 91]. Hence, network-level multicasting cannot in general be considered as a feasible solution for applications deployed over a WAN (for example TIB/RV or the CORBA Notification Service uses network multicast only for diffusing notifications inside a local area network).

Mobile Networks

Several works on pub/sub systems address the mobile network scenario, in two different fashions. One group of works consider each node in the infrastructure having the possibility to move, e.g. constituting a mobile ad-hoc network (MANET) [59, 3, 93, 64]. In other works, part of the nodes form a fixed infrastructure and only clients can roam, being always at one-hop away from the fixed infrastructure [50, 21, 69]. In both cases the network interfaces for mobile nodes can be either a transport protocol built on the top of a data link layer specific for mobile nodes, such as 802.11b, or the protocol 802.11b itself. Obviously mobility induces specific resource constraints over the overall architecture design such as battery drain, limited bandwidth etc. Also phenomenon such as temporary disconnections and node unavailability should be considered common and have to be dealt with.

Section 1.7 specifically focuses on issues related to mobility in pub/sub systems.

Sensor Networks

A sensor network is composed by small devices capable of taking various measurements from the environment, and transmitting them toward applications hosted into specific base stations. Sensors communicate (among each other and with base stations) through broadcast-based facilities over wireless protocols such as 802.15.4. It is evident that the pub/sub paradigm fits naturally this context: sensors publish measurements, that are received by subscribers placed in base stations. With respect to a general pub/sub system, a further assumption can be made in this context: the number of subscribing nodes is very low, as sensors are exclusively publishers and they are predominant in number with respect to base stations.

From the architectural point of view, sensor networks are similar to MANETs, regarding aspects such as the topology determined by the devices transmission range and the limited power supply. Obviously, the fact that a sensor network is a fixed network reduces the complexity related to dynamic topology changes, though dynamicity has still to be taken into account, because devices are in general failure-prone and they frequently rely on stand-by periods for power saving. Works presenting pub/sub solutions suited to sensor networks are [58, 32].

1.4.2 Overlay Infrastructure

A pub/sub system generally builds upon an application-level overlay network. In the following we discuss the possible pub/sub overlays, characterized by the organization of the nodes, the role of each node (pure server or also acting as client) and the overall functionality on which the event-routing algorithm rely on. The discussion includes the conditions under which each infrastructure is more feasible and the constraints it imposes.

Broker Overlay

The support for distributed applications spanning a wide-area, Internet-size network requires the pub/sub system to be implemented as a set of independent, communicating servers. In this context, each single server is called a *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 particular case of systems composed by a single broker (centralized architecture) is often considered in the literature [49, 106][2]

[2] Though centralized architectures are of practical interest being particularly suitable for small-scale deployments, they are evidently out of the focus of our work and will not be considered in the following.

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, so that the neighborhood in the network is determined purely by a knowledge relation. The topology is assumed to be managed by an administrator, based on technical or administrative constraints. For this reason, a broker overlay is inherently static: topology changes are considered to be rare, mainly to face events such as addition of new brokers or repairing after a failure.

The broker network is the most common choice in actual pub/sub implementations, being used by system such as TIB/RV [71], Gryphon [55], SIENA [92], JEDI [38] and REDS [39], as well as in several event routing algorithms proposed in the literature [19, 103]. Apart from the routing protocols, that we analyze in Section 1.5, the main aspect to be clarified in this type of infrastructure is the topology formed by the brokers themselves. There are basically two solutions, hierarchical or flat. In a hierarchical topology, brokers are organized in tree structures, where subscribers' access points lie at the bottom and publishers' access points are roots (or vice versa). Many contributions [12, 103] rely on this topology, thanks to the simplifications it can allow since notifications are diffused only in one direction. In a flat topology, a broker can be connected with any other broker, with no restrictions. [23] shows the more effective load-balance obtained with a flat topology with respect to a hierarchical one, due to the fact that brokers belonging to upper levels of the hierarchy experience a higher load than ones at lower levels.

Peer-to-peer Structured Overlay

A peer-to-peer structured overlay infrastructure 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. Differently from a broker overlay infrastructure, a structured overlay allows to better handle dynamic aspects of the systems such as faults and node joins. Then, it is more suited in unmanaged environments (for example, large-scale decentralized networks) characterized by high dynamicity, where human administration interventions cannot be considered a feasible solution.

As a consequence of the popularity of structured overlays, many such systems have been developed: we cite among the others Pastry [88], Chord [96], Tapestry [108] (unicast diffusion) or CAN [82], I3 [95] and Astrolabe [83] (multicast diffusion). Structuring a pub/sub system over an overlay network infrastructure means leveraging the self-organization capabilities of the infrastructure, by building a pub/sub interface over it. The event routing algorithm is realized only exploiting the communication primitives provided by the underlying overlay. Examples of systems

using this solution are Bayeux [109] and Scribe [87], for what concerns topic-based systems, and Meghdoot, Hermes [80] and Rebeca [99], for what concerns content-based systems. Finally, we cite SelectCast [16], a multicast system built on top of Astrolabe providing a SQL-like syntax for expressing subscriptions.

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 [73]. 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 (that is, it does not rely on a human administrator).

Unstructured overlays use flooding, gossiping or random walks [77, 101, 30, 8] on the overlay graph to diffuse and to retrieve information associated with the nodes. This is due to the difficulties involved in maintaining deterministic data structures for event routing in a setting characterized by the dynamic behaviour of participants (see Section 1.5.2.4 for details). On the other hand, unstructured overlays are widely used for file sharing applications for their simplicity in handling joins and leaves of nodes (with respect to their structured counterparts) and for the fact that, in such applications, there is no need for precise searches. So unstructured overlay are probabilistic in nature as there is non-zero possibility that some item present in the network is not found during a search.

Overlays for Mobile Networks

In a mobile setting, topology changes in the overlay are due to the mobility pattern as well as churn and node failures. Mobility determines the topology of the network and makes impossible to make optimization as in the peer-to-peer unstructured overlays, such as keeping small diameter networks. Moreover, specific algorithms are required for creating and maintaining the conditions under which the event routing algorithm can work, such as connectivity or consistency of the event routing data structures [59, 3, 78]. We detail this aspect in Section 1.5.4. Running algorithms for keeping tree-topology over a set of mobile nodes can be expensive in terms of resources by blocking the computation till a tree is formed. This, as well as the scarce computational resources normally available to mobile nodes, makes the broker overlay and the structured overlay infrastructures not suited to this environment. Hence, typical solutions to event routing are based on an unstructured overlay and rely directly on the MAC layer (e.g. 802.11b) by exploiting its beaconing system and its broadcast characteristics [6]. These approaches have similarities to data management in the Bayou system [76] and can be easily ported over a unstructured peer-to-peer overlay network.

Overlays for Sensor Networks

Differently to mobile networks, in sensor networks it is less difficult to maintain some form of structured overlay, though the limited reliability and computational capability of devices, pose some limits to its realization. Broker overlays are obviously unfeasible, while structured overlays can be realized, including accurate design solutions that allow to cope with frequent failures. In overall, unstructured overlays suits more seamlessly to the communication model used by the sensor devices.

1.4.3 Event Routing

The core mechanism behind a distributed pub/sub system is event routing. Informally, event routing is the process of delivering an event to all the subscribers that issued a matching subscription before the publication. This involves a visit of the nodes in the Notification Service in order to find, for any published event, all the clients whose registered subscription is present in the system at publication time.

The impossibility of defining a global temporal ordering between a subscription and a publication that occurred at two different nodes makes this definition of routing rather ambiguous. A discussion on this point as well as formal specifications of the event routing problem can be found in [7].

The main issue with an event routing algorithm is scalability. That is, an increase of the number of brokers, subscriptions and publications should not cause a serious (e.g., exponential) degradation of performance. This requires on one hand controlling the publication process, in order to possibly involve in propagation of events only those brokers hosting matching subscriptions. On the other, reducing the amount of routing information to be maintained at brokers, in order to support and flexibly allow subscription changes. These two aspects are evidently conflicting and reaching a balance between them is the main aim of a pub/sub system's designer.

We have identified and classified the approaches presented in literature for event routing. Routing approaches are oblivious to the particular architectural solution in the sense that a same routing algorithm can be used in different infrastructures, though each approach can be more suitable for a specific architecture. Section 1.5 is entirely devoted to describing and comparing routing algorithms, as well as identifying which type of infrastructure is more suitable for each solution.

1.4.4 Matching

Matching is the process of checking an event against a subscription. Matching is performed by the pub/sub system in order to determine whether dispatching the

event to a subscriber or not. As we show in the following section, also event routing algorithms often require a matching phase to support the routing choices. As the context of interest is that of large-scale systems, we expect on one side the overall number of subscriptions in the system to be very high, and on the other a high rate of events to be processed. Then, in general the matching operation has to be performed often and on massive data sizes. While obviously this poses no problems in a topic-based system, where matching reduces to a simple table lookup, it is a fundamental issue for the overall performance of a content-based system. The trivial solution of testing sequentially each subscription against the event to be matched often results in poor performance. Techniques for efficiently performing the matching operation are then one important research issue related in the pub/sub field. They can be grouped in two main categories [86], namely predicate indexing algorithms and testing network algorithms. Predicate indexing algorithms are structured in two phases: the first phase is used to decompose subscriptions into elementary constraints and determine which constraints are satisfied by the notification; in the second phase the results of the first phase are used to determine the filters in which all constraints match the event. Matching algorithms falling into the predicate indexing family are [105, 74, 49, 25]. Testing network algorithms ([1, 53, 18]) are based on a pre-processing of the set of subscriptions that builds a data structure (a tree in [1] and [53] or a binary decision diagram in [18]) composed by nodes representing the constraints in each filter. The structure is traversed in a second phase of the algorithm, by matching the event against each constraint. An event matches a filter when the data structure is completely traversed by it. This quick overview is not intended to cover all the works proposed in the literature and was introduced here mainly for the sake of completeness, since the focus of our work is on distributed event routing. A formal complexity analysis and comparison of matching algorithms can be found in [61].

		Routing	*Filtering*	*Nodes storing Subs*	*Nodes handling Events*
Flooding	Event flooding	Det.	Subscribers	None	All
	Subs Flooding	Det.	Publishers	All	None
Selective	Filter-Based	Det.	Intermediaries	Subset	Subset
	Rendezvous-based	Det.	Intermediaries	Subset	Subset
Gossiping	Basic gossiping	Prob.	Subscribers	None	All
	Informed gossiping	Prob.	Intermediaries	Subset	Subset

Table 1.2 Classification of Event Routing Algorithms

1.5 Event Routing

In this section, we investigate the general solutions for event routing to achieve scalable information dissemination. Three categories are identified, flooding algorithms

(event flooding and subscription flooding), selective algorithms (rendezvous-based and filter-based) and event gossiping algorithms (basic gossiping and informed gossiping). Roughly, flooding algorithms are based on a complete deterministic dissemination of event or subscriptions to the entire system. Selective algorithms aims at reducing this dissemination thanks to a deterministic routing structure built upon subscriptions, that aids in the routing process. Event gossiping are probabilistic algorithms with no routing structure, suitable for highly dynamic contexts such as mobile ad-hoc networks. The general characteristics of the algorithms are summarized in Table 1.2, reporting for each algorithm the type of routing decisions (probabilistic or deterministic), the nodes that perform the filtering (producers, consumers or intermediate nodes on the path from publishers to subscribers) and the nodes to which events and subscriptions are sent (none, all or a subset) [3].

In the remainder of this section we give a detailed description of all routing solutions, stating the relationship between each algorithm and the various overlay infrastructures and identifying their trade-offs in terms of the following dimensions [4]:

- Message overhead: the overhead induced on the network by sending both publication and subscription messages. It is normally measured in terms of overlay hops, that is the number of nodes that are traversed by an event along propagation. Ideally, an event routing algorithm should reach all subscribers in a single hop. All the further messages besides these are considered as overhead.
- Memory overhead: the amount of information stored at each process. Related to subscription replication, which is the number of copies of each subscription that are present in the system.
- Subscription language limitations: the routing mechanism may induce limitations on the supported subscriptions, for example regarding the type of constraints.

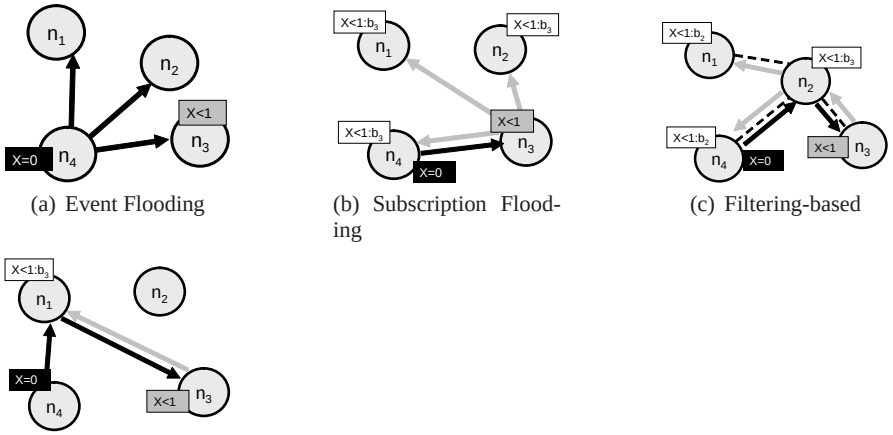
Besides these aspects, one has finally to consider that event routing algorithm are subject to two types of dynamic changes: i) the behavior of users dynamically changing their subscriptions and ii) the changes in the composition of the system due to the process of arrival, departures and failure of nodes (that is, churn). While all event routing algorithms (except event flooding) are equally subject to the first type of dynamicity, some are more sensitive than others to the latter type, according to the overlay infrastructure they are deployed over. We also highlight this issue in the presentation of the algorithms and discuss it in detail in Section 1.5.2.4.

1.5.1 Event/Subscription Flooding

The trivial solution for event routing consists in propagating each event from the publisher to all the nodes in the system (*event flooding*, Figure 1.3(a)). This algo-

[3] In the following we refer as node to a generic node of the pub/sub system, let this be a broker in a broker overlay or a peer in a structured/unstructured overlay. Clients in broker-based architectures are not considered and their behavior is completely handled by nodes in the pub/sub system.

[4] A simulation study of event routing algorithms has recently appeared in [14].



(a) Event Flooding (b) Subscription Flooding (c) Filtering-based

(d) Rendezvous-based

Fig. 1.3 Example of Event Routing Algorithms. Black boxes and arrows represent published and sent events, gray boxes and arrows represent stored and sent subscriptions and white boxes represent stored routing information.

rithm can be simply implemented in all the architectures: a network-based solution consists in broadcasting each event in the whole network, while with any form of overlay it suffices for a node to forwards each event to all the known processes. The obvious drawback is that this routing mechanism does not scale in terms of message overhead. However, event flooding presents minimal memory overhead (no routing information needs to be stored at a node) and there are no language limitations.

On the other side of the spectrum of routing solutions with respect to the routing information stored at each node lies the *subscription flooding* approach: each subscription is sent to all the nodes together with the identifier of the subscriber. That is, each node has the complete knowledge of the entire system, thus recipients can be reached in a single hop (the ideal value) and non-interesting events can be immediately filtered out at producers (Figure 1.3(b)). However, both simulations studies ([67, 23]) and practical experiences ([90]) report that subscription flooding can rarely be considered a feasible solution if subscriptions change at a high rate, as each node has to send all the changes to all other nodes (in other words, the overlay is completely connected). For example, the complete flooding of subscriptions was a characterizing feature (referred to as “quenching”) of an older version of Elvin [89], which was removed in a successive version ([90]), since it proved to be very costly. A recent work presenting a subscription flooding approach is MEDYM [20], an algorithm part of the DADI framework.

1.5.2 Selective Event Routing

The principle behind *Selective Event Routing* algorithms is to reduce the message overhead of event flooding by letting only a subset of the nodes in the system store each subscription and a subset of the nodes be visited by each event (both subsets possibly spanning the whole system). Selective routing algorithms allow to save network resources particularly when an event has to be transmitted only to a restricted portion of subscribers. When most events are of interest for a large number of subscribers, flooding can be considered an option [24, 68] since it avoids the overhead due to the storage and update of event routing information.

1.5.2.1 Filtering-based Routing

In *Filtering-based routing* [23] events are forwarded only to nodes that lie on an overlay path leading to interested subscribers. Message overhead is reduced by identifying as soon as possible events that are not interesting for any subscriber and arrest their forwarding. This approach has been largely studied and used in the literature [68, 14].

The construction of diffusion paths requires routing information to be stored and maintained on the nodes. Routing information at a node is associated to each of its neighbors in the overlay and consists in the set of subscriptions that are reachable through that broker. This allows to build reverse paths to subscribers followed by events. In practice, copies of all the subscriptions have to be diffused toward all possible publishers, and in the general case when all nodes may act as publishers for any subscription, this means again flooding all subscriptions. However, differently from the subscription flooding approach, a node communicates directly only with its neighbors, thus reducing the local message overhead due to subscription update. Subscription diffusion can also be limited in this approach by exploiting subscription containment, as done in SIENA and REBECA.

The pseudo-code of filtering-based routing at a broker is presented in Figure 1.4. A broker can handle *publish* or *subscribe* messages, respectively sent by a client or by another broker. Each broker maintains three structures: a *neighbors* list, a *routing* table and a *subscription* list. The *routing* table associates a neighbor with an entry representing a set of subscriptions. The *subscription* list associates a node to its subscription. The *match* function matches an event against either the subscription list or a routing table entry and returns a list with all the matching nodes. An example of Filtering-based routing is depicted in Figure 1.3(c), where the dashed lines represent connections at overlay level.

The natural architecture for this kind of solution is the brokers' network, usually structured in an acyclic topology (tree or graph). Actually, the presence of cycles requires duplicate detection while diffusing both event and subscriptions and thus is usually avoided in implemented systems. The addressing scheme of a structured overlay does not represent a useful feature in this type of solution, except for the fact that it can keep the consistent association between a node and its position in the

```
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 s to subscriptions;
  else add (x,s) to routing
  send s to neighbors − x;
```

Fig. 1.4 Pseudo code of Filtering-based routing

overlay, allowing easy overlay repairing upon failure. However, the consistency of information in the routing tables has still to be provided by specific event routing-level algorithms. This type of solution is considered in Hermes [80] and in [99, 33]. The use of Filtering-based routing over unstructured overlays suffers mainly from the dynamicity of the network, that requires frequent updates of the routing information. Moreover, it is not possible to assume an acyclic topology.

The performance of filtering-based routing is obviously influenced by the topology of the overlay network. In particular, the diameter of the topology is related to the length of the overlay paths traveled by events, thus affecting notifications latency. Obviously, increasing the number of neighbors of a node lowers the diameter of the network, but also the amount of routing information kept by nodes (memory overhead) increases. This is the reason why the efficiency of the matching algorithm also impacts on delivery latency.

Finally, filtering-based routing does not impose any limitation on the subscription language. Indeed, the only point in the algorithm dependent on the language is the $match()$ function, that can be implemented easily for any data type.

1.5.2.2 Rendezvous-based Routing

Rendezvous-based event routing is based on two functions, namely SN and EN, used to associate respectively subscriptions and events to nodes in the pub/sub system. In particular, given a subscription σ, $SN(\sigma)$ returns a set of nodes, named *rendezvous nodes of* σ, which are responsible for storing σ and forwarding events matching σ to all the subscribers of σ. $EN(e)$ complements SN by returning the *rendezvous nodes of e*, which are the nodes responsible for matching e against subscriptions registered in the system. Upon issuing a subscription σ, a subscriber sends σ to the nodes in $SN(\sigma)$, which store σ and the subscribers' identifier.

Then, rendezvous-based event routing is a two phase process: a publisher sends their events to nodes in $EN(e)$, which match e against the subscriptions they host. For each subscription matched by e, e is forwarded to the corresponding subscriber.

In order for the matching scheme to work and forward e to the consumers, it is necessary that the rendezvous nodes of e collectively store all the subscriptions matched by e, i.e., if $e \in \sigma$ for any subscription σ, then $EN(e) \cap SN(\sigma) \neq \emptyset$. We refer to this property as the *mapping intersection rule* [10]. The pseudo-code of rendezvous-based routing is presented in Figure 1.5 (*subscriptions* list is defined as in Filtering-based routing), while an example is depicted in Figure 1.3(d), where we assume SN/EN functions that assign subscription $x < 1$ and event $x = 0$ to node n_1.

```
upon receive publish(event e) from node x at node i
 rvlist ← EN(e);
 if i ∈ rvlist then
      matchlist ← match(e,subscriptions);
      send notify(e) to matchlist;
 else
 send(e) to rvlist;
upon receive subscribe(subscription s) from node x at node i
 rvlist ← SN(s);
 if i ∈ rvlist then
      add s to subscriptions;
 else
      send(s) to rvlist;
```

Fig. 1.5 Rendezvous-based routing

Rendezvous-based routing has been introduced in [103], and recently many systems appeared following such a scheme (Scribe [26], Bayeux [109], Hermes [79], Meghdoot [57] and [10]). This approach is motivated by the fact that a controlled subscription distribution allows to better load balance subscription storage and management: all subscriptions matching the same events will be hosted by the same node, avoiding a redundant matching to be performed in several different nodes. Also delivery of events is simplified, consisting in the creation of single-rooted diffusion trees starting from target brokers and spanning all subscribers.

However, it is clear that defining the couple of $EN(e)$ and $SN(\sigma)$ functions so that they satisfy the mapping intersection rule is a non-trivial task. This implies defining a clustering of the subscription space, such that each cluster is assigned to a node that becomes the rendezvous for the subscriptions and events that fall into that cluster.

A rendezvous-based algorithm over a broker-based architecture does not handle well dynamicity: when a new node n joins the system, the whole partitioning criteria has to be rearranged among nodes. Moreover, subscriptions that map to n's partition have to be moved to n from the node that was previously in charge. Similarly, when a node leaves or crashes, the subscriptions that it stores should be relocated to another node. Unstructured networks are even less suitable in this sense because the system is highly dynamic and its size not known. On the contrary, the powerful abstraction realized by structured overlay networks greatly helps in the definition of the map-

ping functions, thanks to the fact that the fixed-size address space can be used as a target of the functions rather than the set of nodes. This allows the mapping to be independent from the actual system composition and not be influenced by changes in it.

Maybe the biggest drawback of rendezvous-based solutions is the restrictions it may impose to the subscription language. In general, mapping a multi-dimensional, multi-typed content-based subscription to the uni-dimensional or bi-dimensional numerical-only address space of structured overlays is not straightforward. While numerical range constraints can be intuitively handled, constraints over string attributes, like substrings, prefixes or suffixes, that are an important part of a content-based language, can be hardly reduced to numerical ranges, then they may be excluded from the subscription language.

As for performance, memory overhead depends on the mapping function used. In general, the mapping function should map a subscription to the lower number of nodes possible in order to satisfy the mapping intersection rule. It is natural though that "larger" subscriptions (i.e. matching more events) will be mapped to more nodes with respect to "smallest" ones. This allows also to share the load due to matching. Moreover, routing information should be preserved at a node to reach the rendezvous nodes.

1.5.2.3 Filter Merging

Filter merging, or aggregation, has been proposed as an optimization strategy for distributed pub/sub systems. Filter merging techniques combine filters to reduce the number of propagated filters and thus the size of distributed state. Merging and covering are needed to reduce processing power and memory requirements both on client devices and on event routers. These techniques are typically general and may be applied to subscriptions, advertisements, and other information represented using filters.

A filter-merging-based routing mechanism was presented in the Rebeca distributed event system [67]. The mechanism merges conjunctive filters using perfect merging rules that are predicate-specific. Routing with merging was evaluated mainly using the routing table size and forwarding overhead as the key metrics in a distributed environment. Merging was used only for simple predicates in the context of a stock application [67]. The integration of the merging mechanism with a routing data structure was not elaborated and we are not aware of any results on this topic. Recently, a framework for dynamic filter merging has been proposed that integrates with content-based routing tables, such as a poset or a forest organized based on the covering relation [97].

The optimal merging of filters and queries with constraints has been shown to be NP-complete [35]. Subscription partitioning and routing in content-based systems have been investigated in [104] using Bloom filters and R-trees for efficiently summarizing subscriptions.

Bloom filters are an efficient mechanism for probabilistic representation of sets, and support membership queries, but lack the precision of more complex methods of representing subscriptions. To take an example, Bloom filters and additional predicate indices were used in a mechanism to summarize subscriptions [100]. An Arithmetic Attribute Constraint Summary (AACS) and a String Attribute Constraint Summary (SACS) structures were used to summarize constraints, because Bloom filters cannot capture the meaning of other operators than equality. The subscription summarization is similar to filter merging, but it is not transparent, because routers need to be aware of the summarization mechanism. Filter merging, on the other hand, does not necessarily require changes to other routers. In addition, the set of attributes needs to be known a priori by all brokers and new operators require new summarization indices. The benefit of the summarization mechanism is improved efficiency, since a custom-matching algorithm is used that is based on Bloom filters and the additional indices.

1.5.2.4 On the guarantee of event delivery

Let us point out that selective-based solutions are deterministic approaches to event routing, in the sense that they build event routing data structure to deterministically route event to its intended destinations. Nodes cooperate for letting these data structures do their best to timely track subscription changes. It is important to remark that deterministic event routing does not imply any deterministic guarantee on event delivery. There is indeed a non-zero delay between a change and the time in which the event routing data structure captures this change. During this delay deterministic approaches to event routing might become inefficient, in the sense that they can lead, on one hand, to event loss due to the fact that an event is routed to part of the overlay where there are no longer interested recipients (e.g., due to recent unsubscription) and, on the other hand, to not routing an event to an interested destination that just did the subscription [7]. Therefore, deterministic approaches to event routing are clearly best-effort in terms of delivery of events due to topology rearrangements.

Moreover, the effect of churn makes much more pronounced the discrepancy between the event routing data structures at a given time and the ones that would allow ideal deterministic event routing, amplifying, thus, the inefficiency of the event routing with respect to the delivery of events. Deterministic event routing approaches work therefore better over overlay infrastructures where the churn is mastered by some external entity. For example, in managed environments such as a broker overlay, the churn is very low and strictly under control of humans. In a peer-to-peer structured overlay, the churn effect is handled by the overlay infrastructure layer and then masked to the event routing level.

As the size and the dynamic of the system grow, the effect of churn can be disruptive in terms of delivery of events in deterministic event routing even in structured peer-to-peer networks [62]. This is why gossip-based (or epidemic) protocols have emerged as an important probabilistic event routing approach to cope with these large scale and dynamic settings [45].

1.5.3 Gossip-based

In basic gossip-based protocols, each node contacts one or a few nodes in each round (usually chosen at random), and exchanges information with these nodes. The dynamics of information spread resembles the spread of an epidemic [40] and lead to high robustness, reliability and self-stabilization [13]. Being randomized, rather than deterministic, these protocols are simple and do not require to maintain any event routing data structure at each node trying to timely track churn and subscriptions changes. The drawback is a moderate redundancy in message overhead compared to deterministic solution. Gossiping is therefore a probabilistic and fully distributed approach to event routing and the basic algorithm achieves high stability under high network dynamics, and scales gracefully to a huge number of nodes [5]. Specific gossip algorithms for pub/sub systems have been proposed in [47, 34, 5, 77, 101, 30, 8].

In gossip protocols, the random choice of the nodes to contact can be sometimes driven by local information, acquired by a node during its execution, describing the state either of the network or of the subscription distribution or both [6]. In this case, we are in the presence of an *informed gossip protocol*. The algorithm presented by Eugster and Guerraoui in [47] (*pmcast*) is an example of informed gossip specifically targeted to pub/sub system. It follows a principle similar to that of filter-based routing: avoiding to gossip a message to not-interested subscribers. *pmcast* organizes processes in a hierarchy of groups. Groups are built and organized in hierarchies according to the physical proximity of nodes. Each process maintains in its view the identities and the subscriptions of its neighbors in a group. Special members in a group, namely delegates, maintain an aggregation of the subscriptions within a group and have access to the delegates view of nodes at adjacent levels of the tree. Events are gossiped throughout the tree. The membership information allows to exclude from gossiping the nodes that are not interested in an event.

Costa and Picco in [77] proposed a hybrid approach that mixes deterministic and probabilistic event routing. Subscriptions are propagated only in the immediate vicinity of a subscriber. Deterministic event routing leverages this subscription information, whenever available, by deterministically routing an event along the link a matching subscription was received from. If no subscription information exists at a given node, events are forwarded along a randomly chosen subset of the available links over the peer-to-peer overlay.

Voulgaris et al. in [101] exploit a more complex architecture where content-based event diffusion is realized traversing multiple layers, each characterized by a different overlay technology. The lower layer exploits a randomized overlay to gossip continuously information about subscriptions among participants; a middle layer is used to maintain semantic links among participants sharing similar interest; finally, an upper layer connects in a ring-like structure all participants that should be no-

[5] Gossiping has been also used to improve delivery guarantee of a filtering-based event routing protocol in [34].

[6] To help this process of acquiring information at each node some limited horizon advertising mechanism can be employed.

tified about a specific event. Event diffusion is realized through gossiping till the point where semantic links can be used to directly adress events.

Chockler et al. in [30] use a different approach. The system they propose is based on a single overlay network where each participant manages both a set of randomized links and a set of semantic links through which it is connected to other participants sharing similar interests. Event diffusion is again realized leveraging semantic links as long as they are available, or resorting to gossiping when they are not.

Finally, Baldoni et al. in [8] propose a two layer infrastructure for topic-based event diffusion based on a uniform peer sampling service. This service is leveraged to uniformly distribute information about participant interests. Event diffusion is realized in two steps: first a random walk traverses a low-level general overlay connecting all participants looking for someone subscribed to the target topic; once such a subscriber is found event diffusion continues at an upper layer leveraging an overlay connecting all participants subscribed to the same topic. This second step can be realized through various techniques like message flooding or gossiping.

1.5.4 Event Routing for MANETs

Wireless MANETs can support both deterministic and probabilistic event routing protocols that we presented till now. However as remarked in previous sections, while in wired networks all event routing algorithms are built on the top of a transport protocol using MAC broadcast only for local performance improvements, in a wireless network this sharp layering is no more a dogma due to battery drain and to the fact that unicast is expensive while multicast can be cheap. This is why event routing algorithms can also either rely on the MAC layer [60, 4, 6] or integrate with the classical MANET routing protocols such as MAODV [107, 65].

Huang and Garcia-Molina [59], Anceaume et al. [3] and Cugola et al. [78] present three algorithms for building and maintaining a tree event routing structure in a mobile ad-hoc network on the top of a transport protocol. In particular, Cugola et al. describe an algorithm for restoring the event routing tables after a disconnection in a generic acyclic graph topology within a mobile ad-hoc network. The paper advocates a separation of concerns between the connectivity layer and the event dispatching layer. The algorithm for repairing the event routing data structure works on the assumption that the tree is kept connected by some loop-free algorithm at the routing level.

The above-described event routing protocols, as well as the ones integrating with a MANET routing protocol, are subject to the problem described in Section 1.5.2.4 since they aim at building deterministic event routing structure over a MANET. This problem is amplified in this context by the frequent overlay topology changes due to mobility, besides the ones due to churn and subscription changes. This is why recent approaches to event routing in MANET rely directly on the MAC layer exploiting the broadcast nature of the medium at the same time [60, 4, 6]. For example, [4] and [6] are structureless in the sense that they do not maintain any deterministic

data structure on the topology at a peer. Therefore, event routing in such cases can only be based on either gossip or on flooding. In particular Baldoni et al. in [6] employ a form of informed flooding event routing based on the euclidean distance between two nodes to direct the event to the destination. This distance is estimated by counting the number of beacon messages missed from a given source. Each peer *p* periodically broadcast a message summarizing its local subscriptions. This allows mobile peers in the proximity of *p* to know *p*'s subscription and to construct its own subscription table. When an event *e* arrives to a peer *p* it checks if there is a matching in its own subscription table and in the affirmative it broadcasts *e* with a delay proportional to the number of beacon messages missed by *p* from the peer matching *e*. If a peer in the proximity of *p* received *e* as well and heard the relay of *e* from *p*, it drops the planned *e*'s forwarding. This creates a wave effect that brings most of the time the event to the intended destination very quickly.

1.5.5 Event Routing for Sensor Networks

Two contributions were proposed ([32, 58]) adapting to the context of wireless sensor networks event routing solutions introduced for wired networks. Costa et al. in [32] propose a sensor-based implementation of their semi-probabilistic algorithm of [77], that exploits only broadcast for communication and introduces specific solutions for reducing the impact of packet collisions. Hall et al. in [58] adapt the content-based networking protocol of [24] to sensor networks, in the form of a routing protocol which extends the acyclic overlay used in [24] with backup routes for handling permanent and transient failures. A further optimization is made, by assuming that the set of possible receivers (i.e., the base stations) is small and known by all nodes: this allows to evaluate the actual receivers directly on the publisher side.

1.6 Security

Security is an important requirement for pub/sub systems. An overview of pub/sub security topics was given in [102]. They propose several techniques for ensuring the availability of the information dissemination network. Prevention of denial-of-service attacks is essential and *customized publication control* is proposed to mitigate large-scale attacks. In this technique, subscribers can specify which publishers are allowed to send them information. A challenge-response mechanism is proposed, in which the subscriber issues a challenge function, and the publisher has to respond to the challenge. The use of the mechanism in a distributed environment was not elaborated.

The EventGuard system comprises of a set of *security guards* to secure pub/sub operations, and a resilient pub/sub network [94]. The basic security building blocks

are tokens, keys, and signatures. Tokens are used within the pub/sub network to route messages, which is not directly applicable for content-based routing. Bogus messages are prevented by authenticating subscribers and publishers using *ElGamal* signatures.

Pub/sub broker networks are vulnerable to message dropping attacks. For example, overlays such as Hermes and Maia [75] may suffer from bogus nodes. The prevention of message dropping attacks has a high cost and only a few systems address them. The EventGuard uses an *r-resilient* network of brokers [94].

Secure event types and type-checking was proposed in [75]. Secure event type definitions contain issuer's public key, version information, attributes, delegation certificates, and a digital signature. Scope-based security was discussed in [51], in which trust networks are created in the broker network using PKI techniques.

1.7 Mobility Support

Most research on event systems has focused on event dissemination in the fixed network, where clients are stationary and have reliable, low-latency, and high bandwidth communication links. Recently, mobility support and wireless communication have become active research topics in many research projects working with event systems, such as Siena [21] and Rebeca [50]. A mobility-aware event system needs to be able to cope with a number of sporadic and unpredictable end systems, to provide fast access to information irrespective of access location, medium and time. Problems such as delayed events, events generated for offline systems and the delay posed by the transmission of events create synchronization and event delivery problems, need to be solved.

User mobility occurs when a user becomes disconnected or changes the terminal device. *Terminal mobility* occurs when a terminal moves to a new location and connects to a new access point. Mobility transparency is a key requirement for the event system and it should be able to hide the complexity of subscription management caused by mobility. Mobile components typically require that the pub/sub topology is updated and thus it is necessary to prove for a mobility protocol that the safety properties are not violated, which is referred to as *mobility-safety* in [98]. Typically, a *stateful* mobility protocol is used that buffers messages for a disconnected client.

In the simplest mobility support setting, a special broker is used to buffer messages for offline pub/sub clients. This was used in the Elvin event system that supports disconnected operation using a centralized proxy, but does not support mobility between proxies [90]. In a wide-area system, mobility support between proxies is needed. A handover or handoff protocol allows clients to move between proxies. Such a handover protocol can also be used for load balancing purposes.

The JEDI event system was one of the first pub/sub systems to support mobile components in a hierarchical topology of event brokers [37]. JEDI maintains causal ordering of events and is based on a tree-topology, which has a potential performance bottleneck at the root of the tree with subscription semantics. Handover is

easier to implement for the hierarchical topology than for a tree or graph topology, because the root of the hierarchy can be used as an anchor point for mobility related signalling.

Mobility support in a tree or graph based distributed event infrastructure, such as Siena and Rebeca, is challenging because of the high cost of the flooding, and issues with mobile publishers. Content-based flooding is needed when filter covering or merging optimizations are being used to compact routing tables. These optimizations lose information regarding the source of a subscription or advertisement, and thus lead to more complicated handover protocols. Mobile publishers, on the other hand, require that events published during mobility or after mobility are delivered to subscribers who were active at the time of the handover and are still active after the handover.

A generalized subscription state transfer protocol consists of four phases:

1. Subscriptions are moved from broker *A* to broker *B*.
2. *B* subscribes to the events.
3. *A* sends buffered notifications to *B*.
4. *A* unsubscribes if necessary.

The problem with this protocol is that B may not know when the subscriptions have taken effect — especially if the routing topology is large and arbitrary. This is solved by synchronizing A and B using events, which potentially involves flooding the content-based network.

To solve this synchronization problem, the Siena event system was extended with generic mobility support, which uses existing pub/sub primitives: publish and subscribe [21]. The mobility-safety of the protocol was formally verified. The benefits of a generic protocol are that it may work on top of various pub/sub systems and requires no changes to the system API. On the other hand, the performance of the mobility support decreases, because mobility-specific optimizations are difficult to realize when the underlying topology is hidden by the API.

In addition to Siena, several other event systems have been extended with a mobility support protocol. In the rest of this section, we will briefly cover mobility friendly systems and recent insights into this problem domain.

Rebeca supports both logical and physical mobility. The basic system is an acyclic routed event network using advertisement semantics. The mobility protocol uses an intermediate node, between the source and target of mobility, called Junction for synchronizing the servers. If the brokers keep track of every subscription the Junction is the first node with a subscription that matches the relocated subscription propagated from the target broker. If covering relations or merging is used this information is lost, and the Junction needs to use content-based flooding to locate the source broker [50].

JECho is a mobility-aware event system that uses opportunistic event channels in order to support mobile clients [29]. The central problem is to support a dynamic event delivery topology, which adapts to mobile clients and different mobility patterns. The requirements are addressed primarily using two mechanisms: proactively locating more suitable brokers and using a mobility protocol between brokers, and

using a load-balancing system based on a central load-balancing component that monitors brokers in a domain.

Recent findings on the cost of mobility in hierarchical routed event infrastructures that use unicast include that network capacity must be doubled to manage with the extra load of 10% of mobile clients [17]. Recent findings also present optimizations for client mobility: *prefetching*, *logging*, *home-broker*, and *subscriptions-on-device*. Prefetching takes future mobility patterns into account by transferring the state while the user is mobile. With logging, the brokers maintain a log of recent events and only those events not found in the log need to be transferred from the old location. The home-broker approach involves a designated home broker that buffers events on behalf of the client. This approach has extra messaging costs when retrieving buffered events. Subscriptions-on-device stores the subscription status on the client so it is not necessary to contact the old broker. In this study the cost of reconfiguration was dominated by the cost of forwarding stored events (through the event routing network).

The cost of publisher mobility has also been recently addressed [69]. They start with a basic model for publisher mobility that simply tears down the old advertisement and establishes it at the new location after mobility. Thus a specific handover protocol is not needed. They confirm the high cost of publisher mobility and present three optimization techniques, namely *prefetching*, *proxy*, and *delayed*. The first exploits information about future mobility patterns. The second uses special proxy nodes that advertise on behalf of the publisher and maintain the multicast trees. The third delays the unadvertisement at the source to exploit the overlap of advertisements, but does not synchronize the source and target brokers. The publisher mobility support mechanisms used in the study are not necessarily mobility-safe.

A formal discrete model for both publisher and subscriber mobility was presented in [98, 97]. In this work, two new properties are defined for the pub/sub topology, namely mobility-safety and completeness. A handover protocol is mobility-safe if it prevents false negatives. A topology or a part of a topology is complete if subscriptions and advertisements are fully established (propagated) throughout it.

Mobility-safety of a generic stateful handover was shown for acyclic pub/sub networks. The completeness of the topology is used to characterize pub/sub handover protocols and optimize them. One of the results of this work is that rendezvous-points are good for pub/sub mobility, because they can be used to limit signalling and flooding of updates.

1.8 Summary and Outlook

Publish/subscribe is now largely acknowledged as one of the most interesting paradigm for distributed interaction. However, the positive characteristics of a pub/sub system (such as scalability) are not directly inherited from the paradigm but has to be enforced by specific architectural and algorithmic solutions. Following this observation, the architectural model that we proposed in this chapter intends to critically

classify and analyze the large amount of research works proposed for distributed event routing since then, pointing out the critical aspects of the different solutions, specifically in terms of interaction and dependencies among the choices made at each architectural layer. We believe that our classification can be of valuable importance to define new solutions for event routing algorithms and their mapping to the overlay infrastructure.

Acknowledgements

The authors wish to thank Emmanuelle Anceaume, Roberto Beraldi, Mariangela Contenti, Gianpaolo Cugola, Maria Gradinariu, Carlo Marchetti, Sara Tucci Piergiovanni and Roman Vitenberg with whom we worked around the theme of publish/subscribe in the last few years. The authors would like also to thank the attendees at the MINEMA Summer School on Middleware for Network Eccentric and Mobile Applications that was held in Klagenfurt in 2005. Discussions during those days were instrumental to form structure and content of this chapter.

References

1. Aguilera, M.K., Strom, R.E., Sturman, D.C., Astley, M., Chandra, T.D.: Matching Events in a Content-Based Subscription System. In: Proceedings of The ACM Symposium on Principles of Distributed Computing (PODC 1999), pp. 53–61 (1999)
2. Altherr, M., Erzberg, M., Maffeis, S.: iBus - a software bus middleware for the java platform. In: Proceedings of the International Workshop on Reliable Middleware Systems, pp. 49–65 (October 1999)
3. Anceaume, E., Datta, A.K., Gradinariu, M., Simon, G.: Publish/subscribe scheme for mobile networks. In: Proceedings of the 2002 Workshop on Principles of Mobile Computing (POMC 2002), pp. 74–81 (2002)
4. Baehni, S., Chhabra, C., Guerraoui, R.: Mobility friendly publish/subscribe. In: EPFL Technical Report 200488 (2004)
5. Baehni, S., Eugster, P.T., Guerraoui, R.: Data-aware multicast. In: Proceedings of the 2004 International Conference on Dependable Systems and Networks (DSN 2004), pp. 233–242 (2004)
6. Baldoni, R., Beraldi, R., Cugola, G., Migliavacca, M., Querzoni, L.: Structure-less Content-Based Routing in Mobile Ad Hoc Networks. In: In proceedings of the International Conference on Pervasive Services (ICPS '05), Santorini, Greece, July 2005 (2005)
7. Baldoni, R., Beraldi, R., Piergiovanni, S.T., Virgillito, A.: On the modelling of publish/subscribe communication systems. Concurrency and Computation: Practice and Experience **17**(12), 1471–1495 (2005)
8. Baldoni, R., Beraldi, R., Quema, V., Querzoni, L., Tucci-Piergiovanni, S.: Tera: topic-based event routing for peer-to-peer architectures. In: DEBS '07: Proceedings of the 2007 inaugural international conference on Distributed event-based systems, pp. 2–13. ACM, New York, NY, USA (2007). DOI http://doi.acm.org/10.1145/1266894.1266898
9. Baldoni, R., Contenti, M., Virgillito, A.: The Evolution of Publish/Subscribe Systems. In: André Schiper and Alexander A. Shvartsman and Hakim Weatherspoon and Ben Y. Zhao

(ed.) Future Trends in Distributed Computing, Research and Position Papers, vol. 2584. Springer (2003)
10. Baldoni, R., Marchetti, C., Virgillito, A., Vitenberg, R.: Content-based publish-subscribe over structured overlay networks. In: International Conference on Distributed Computing Systems (ICDCS 2005) (2005)
11. Banavar, G., Chandra, T., Mukherjee, B., Nagarajarao, J., Strom, R., Sturman, D.: An Efficient Multicast Protocol for Content-based Publish-Subscribe Systems. In: Proceedings of International Conference on Distributed Computing Systems (ICDCS '99) (1999)
12. Bhola, S., Strom, R., Bagchi, S., Zhao, Y., Auerbach, J.: Exactly-once Delivery in a Content-based Publish-Subscribe System. In: Proceedings of The International Conference on Dependable Systems and Networks (2002)
13. Birman, K.P., Hayden, M., Ozkasap, O., Xiao, Z., Budiu, M., Minsky, Y.: Bimodal multicast. ACM Transactions on Computer Systems (TOCS) **17**(2), 41–88 (1999)
14. Bittner, S., Hinze, A.: A classification of filtering algorithms in content-based publish/subscribe systems. In: Proceedings of COOPIS 2005 (2005)
15. Bittner, S., Hinze, A.: On the benefits of non-canonical filtering in publish/subscribe systems. In: Proceedings of the International Workshop on Distributed Event-Based Systems (ICDCS/DEBS'05) (2005)
16. Bozdog, A., van Renesse, R., Dumitriu, D.: Selectcast – a scalable and self-repairing multicast overlay routing facility. In: Proceedings of the First ACM Workshop on Survivable and Self-Regenerative Systems (2003)
17. Burcea, I., Jacobsen, H.A., de Lara, E., Muthusamy, V., Petrovic, M.: Disconnected operation in publish/subscribe middleware. In: Mobile Data Management (2004)
18. Campailla, A., Chaki, S., Clarke, E.M., Jha, S., Veith, H.: Efficient filtering in publish-subscribe systems using binary decision diagrams. In: Proceedings of The International Conference on Software Engineering, pp. 443–452 (2001)
19. Cao, F., Singh, J.P.: Efficient Event Routing in Content-based Publish-Subscribe Service Networks. In: Proceedings of the 23rd Conference on Computer Communications (IEEE INFOCOM 2004) (2004)
20. Cao, F., Singh, J.P.: Medym: Match-early with dynamic multicast for content-based publish-subscribe networks. In: Proceedings of the ACM/IFIP/USENIX 6th International Middleware Conference (Middleware 2005) (2005)
21. Caporuscio, M., Carzaniga, A., Wolf, A.L.: Design and evaluation of a support service for mobile, wireless publish/subscribe applications. IEEE Transactions on Software Engineering **29**(12), 1059–1071 (2003)
22. Carzaniga, A., Rosenblum, D., Wolf, A.: Achieving Scalability and Expressiveness in an Internet-Scale Event Notification Service. In: Proceedings of the ACM Symposium on Principles of Distributed Computing, pp. 219–227 (2000)
23. Carzaniga, A., Rosenblum, D., Wolf, A.: Design and Evaluation of a Wide-Area Notification Service. ACM Transactions on Computer Systems **3**(19), 332–383 (Aug 2001)
24. Carzaniga, A., Rutherford, M.J., Wolf, A.L.: A routing scheme for content-based networking. In: Proceedings of IEEE INFOCOM 2004 (2004)
25. Carzaniga, A., Wolf, A.L.: Forwarding in a content-based network. In: Proceedings of ACM SIGCOMM 2003, pp. 163–174 (2003)
26. Castro, M., Druschel, P., Kermarrec, A., Rowston, A.: Scribe: A large-scale and decentralized application-level multicast infrastructure. IEEE Journal on Selected Areas in Communications **20**(8) (October 2002)
27. Chand, R., Felber, P.: Xnet: A reliable content-based publish/subscribe system. In: 23rd International Symposium on Reliable Distributed Systems (SRDS 2004), pp. 264–273 (2004)
28. Chand, R., Felber, P.: Semantic peer-to-peer overlays for publish/subscribe networks. In: Parallel Processing, 11th International Euro-Par Conference (Euro-par 2005), pp. 1194–1204 (2005)
29. Chen, Y., Schwan, K., Zhou, D.: Opportunistic channels: Mobility-aware event delivery. In: Middleware, pp. 182–201 (2003)

30. Chockler, G., Melamed, R., Tock, Y., Vitenberg, R.: Spidercast: a scalable interest-aware overlay for topic-based pub/sub communication. In: DEBS '07: Proceedings of the 2007 inaugural international conference on Distributed event-based systems, pp. 14–25. ACM, New York, NY, USA (2007). DOI http://doi.acm.org/10.1145/1266894.1266899
31. Cilia, M.: An active functionality service for open distributed heterogeneous environments. Ph.D. thesis, Department of Computer Science, Darmstadt University of Technology (2002)
32. Costa, G., Picco, G.P., Rossetto, S.: Publish-subscribe on sensor networks: A semi-probabilistic approach. In: Proceedings of 2nd IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS 2005) (2005)
33. Costa, P., Frey, D.: Publish-subscribe tree-maintenance over a dht. In: Proceedings of the International Workshop on Distributed Event-Based Systems (ICDCS/DEBS'05) (2005)
34. Costa, P., Migliavacca, M., Picco, G., Cugola, G.: Introducing Reliability in Content-Based Publish-Subscribe through Epidemic Algorithms. In: Proceedings of the 2nd International Workshop on Distributed Event-Based Systems (DEBS'03) (2003)
35. Crespo, A., Buyukkokten, O., Garcia-Molina, H.: Query merging: Improving query subscription processing in a multicast environment. IEEE Trans. Knowl. Data Eng. **15**(1), 174–191 (2003)
36. Cugola, G., de Cote, J.E.M.: On introducing location awareness in publish-subscribe middleware. In: Proceedings of the International Workshop on Distributed Event-Based Systems (ICDCS/DEBS'05) (2005)
37. Cugola, G., Nitto, E.D., Fuggetta, A.: The JEDI Event-Based Infrastructure and Its Application to the Development of the OPSS WFMS. IEEE Transactions on Software Engineering **27**(9), 827–850 (2001)
38. Cugola, G., Nitto, E.D., Fuggetta, A.: Exploiting an event-based infrastructure to develop complex distributed systems. In: Proceedings of the 10th International Conference on Software Engineering (ICSE '98) (April 1998)
39. Cugola, G., Picco, G.P.: Reds: A reconfigurable dispatching system. In: Technical Report, Politecnico di Milano (2005)
40. Demers, A., Greene, D., Hauser, C., Irish, W., Larson, J.: Epidemic algorithms for replicated database maintenance. In: Proceedings of the sixth annual ACM Symposium on Principles of Distributed Computing, pp. 1–12 (1987)
41. Diot, C., Levine, B., Lyles, B., Kassem, H., Balensiefen, D.: Deployment issues for the ip multicast service. IEEE Network Magazine, special issue on Multicasting (2000)
42. Eugster, P., Felber, P., Guerraoui, R., Handurukande, S.: Event Systems: How to Have Your Cake and Eat It Too. In: Proceedings of the International Workshop on Distributed Event-Based Systems (DEBS'02) (2002)
43. Eugster, P., Felber, P., Guerraoui, R., Kermarrec, A.: The Many Faces of Publish/Subscribe. ACM Computing Surveys **35**(2), 114–131 (2003)
44. Eugster, P., Guerraoui, R., Damm, C.: On Objects and Events. In: Proceedings of the Conference on Object-Oriented Programming Systems, Languages and Applications (OOPSLA) (2001)
45. Eugster, P., Guerraoui, R., Kermarrec, A.M., Massoulie, L.: From Epidemics to Distributed Computing. IEEE Computer **37**(5), 60–67 (2004)
46. Eugster, P.T.: Type-based publish/subscribe. Ph.D. thesis, EPFL (2001)
47. Eugster, P.T., Guerraoui, R.: Probabilistic multicast. In: Proceedings of the 3rd IEEE International Conference on Dependable Systems and Networks (DSN 2002), pp. 313–322 (2002)
48. Eugster, P.T., Guerraoui, R.: Distributed programming with typed events. IEEE Software **21**(2), 56–64 (2004)
49. Fabret, F., Jacobsen, A., Llirbat, F., Pereira, J., Ross, K., Shasha, D.: Filtering algorithms and implementation for very fast publish/subscribe. In: Proceedings of the 20th Intl. Conference on Management of Data (SIGMOD 2001), pp. 115–126 (2001)
50. Fiege, L., Gärtner, F.C., Kasten, O., Zeidler, A.: Supporting mobility in content-based publish/subscribe middleware. In: ACM/IFIP/USENIX International Middleware Conference (Middleware 2003), pp. 103–122 (2003)

51. Fiege, L., Zeidler, A., Buchmann, A.P., Kilian-Kehr, R., Mühl, G.: Security aspects in publish/subscribe systems. In: Third Intl. Workshop on Distributed Event-based Systems (DEBS'04). Edinburgh, Scotland, UK (2004). URL http://www.kbs.cs.tu-berlin.de/publications/fulltext/debs04.pdf
52. Filho, R.S.S., Redmiles, D.F.: A survey of versatility for publish/subscribe infrastructures. In: ISR Technical Report UCI-ISR-05-8, Institute for Software Research, University of California, Irvine (2005)
53. Gough, K., Smith, G.: Efficient recognition of events in distributed systems. In: Proceedings of the ACSC-18 (1995)
54. Gruber, R.E., Krishnamurthy, B., Panagosf, E.: The architecture of the ready event notification service. In: Proceedings of The International Conference on Distributed Computing Systems, Workshop on Middleware (Austin, Texas, 1999)
55. Gryphon Web Site: http://www.research.ibm.com/gryphon/
56. Guimaraes, M., Rodrigues, L.: A genetic algorithm for multicast mapping in publish-subscribe systems. In: Proceedings of the 2nd IEEE International Symposium on Network Computing and Applications, pp. 67–74 (2003)
57. Gupta, A., Sahin, O., Agrawal, D., Abbadi, A.E.: Meghdoot: Content-based publish:subscribe over p2p networks. In: Proceedings of the ACM/IFIP/USENIX 5th International Middleware Conference (Middleware'04) (2004)
58. Hall, C.P., Carzaniga, A., Rose, J., Wolf, A.L.: A content-based networking protocol for sensor networks. Tech. Rep. CU-CS-979-04, Department of Computer Science, University of Colorado (2004)
59. Huang, Y., Garcia-Molina, H.: Publish/Subscribe in a Mobile Environment. In: Proceedings of the 2nd ACM International Workshop on Data Engineering for Wireless and Mobile Access (MobiDE) (2001)
60. Huang, Y., Garcia-Molina, H.: Publish/subscribe tree construction in wireless ad-hoc networks. In: 4th International Conference on Mobile Data Management (MDM 2003), pp. 122–140 (2003)
61. Kale, S., Hazen, E., Cao, F., Singh, J.P.: Analysis and Algorithms for Content-based Event Matching. In: Proceedings of the Fourth International Workshop on Distributed Event-Based Systems (DEBS '05) (2005)
62. Liben-Nowell, D., Balakrishnan, H., Karger, D.R.: Analysis of the evolution of peer-to-peer systems. In: Proceedings of the Twenty-First Annual ACM Symposium on Principles of Distributed Computing (PODC 2002), pp. 233–242 (2002)
63. Liu, Y., Plale, B.: Survey of publish/subscribe event systems. In: Indiana University Computer Science Technical Report TR-574 (2003)
64. Meier, R., Cahill, V.: Steam: Event-based middleware for wireless ad hoc networks. In: Proceedings of the International Workshop on Distributed Event-Based Systems (ICDCS/DEBS'02) (2002)
65. Mottola, L., Cugola, G., Picco, G.P.: Tree overlays for publish-subscribe in mobile ad hoc networks. In: Technical Report, Politecnico di Milano (2005)
66. Muhl, G.: Generic Constraints for Content-Based Publish/Subscribe. In: Proceedings of the 6th International Conference on Cooperative Information Systems (CoopIS) (2001)
67. Muhl, G.: Large-Scale Content-Based Publish/Subscribe Systems. Phd thesis, Department of Computer Science, Darmstadt University of Technology (2002)
68. Mühl, G., Fiege, L., Gärtner, F.C., Buchmann, A.P.: Evaluating advanced routing algorithms for content-based publish/subscribe systems. In: The Tenth IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS 2002), pp. 167–176 (2002)
69. Muthusamy, V., Petrovic, M., Jacobsen, H.A.: Effects of routing computations in content-based routing networks with mobile data sources. In: Proceedings of the 11th annual international conference on Mobile computing and networking (MobiCom '05), pp. 103–116 (2005)
70. Object Management Group: CORBA event service specification, version 1.1. OMG Document formal/2000-03-01 (2001)

71. Oki, B., Pfluegel, M., Siegel, A., Skeen, D.: The information bus - an architecture for extensive distributed systems. In: Proceedings of the 1993 ACM Symposium on Operating Systems Principles (December 1993)
72. Opyrchal, L., Astley, M., Auerbach, J.S., Banavar, G., Strom, R.E., Sturman, D.C.: Exploiting IP multicast in content-based publish-subscribe systems. In: Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2000), pp. 185–207 (2000)
73. Pandurangan, G., Raghavan, P., Upfal, E.: Building low-diameter p2p networks. In: IEEE Symposium on Foundations of Computer Science, pp. 492–499 (2001)
74. Pereira, J., Fabret, F., Llirbat, F., Shasha, D.: Efficient matching for web-based publish/subscribe systems. Springer-Verlag (2001)
75. Pesonen, L., Bacon, J.: Secure Event Types in Content-based, Multi-Domain Publish/Subscribe Systems. In: Proceedings of SEM 2005. ACM (2005)
76. Petersen, K., Spreitzer, M.J., Terry, D.B., Theimer, M.M., Demers, A.J.: Flexible update propagation for weakly consistent replication. In: Proceedings of the 16th ACM Symposium on Operating Systems Principles (SOSP-16), pp. 288–301 (1997)
77. Picco, G.P., Costa, G.: Semi-probabilistic publish/subscribe. In: Proceedings of 25th IEEE International Conference on Distributed Computing Systems (ICDCS 2005) (2005)
78. Picco, G.P., Cugola, G., Murphy, A.L.: Efficient content-based event dispatching in the presence of topological reconfiguration. In: 23rd International Conference on Distributed Computing Systems (ICDCS 2003), 19-22 May 2003, Providence, RI, USA, pp. 234–243 (2003)
79. Pietzuch, P., Bacon, J.: Hermes: a distributed event-based middleware architecture. In: Proceedings of the International Workshop on Distributed Event-Based Systems (DEBS'02) (2003)
80. Pietzuch, P., Bacon, J.: Peer-to-peer overlay broker networks in an event-based middleware. In: Proceedings of the 2nd International Workshop on Distributed Event-Based Systems (DEBS'03) (2003)
81. Preotiuc-Pietro, R., Pereira, J., Llirbat, F., Fabret, F., Ross, K., Shasha, D.: Publish/subscribe on the web at extreme speed. In: Proc. of ACM SIGMOD Conf. on Management of Data. Cairo, Egypt (2000)
82. Ratnasamy, S., Handley, M., Karp, R., Shenker, S.: Application-level multicast using content-addressable networks. Lecture Notes in Computer Science **2233**, 14–34 (2001)
83. van Renesse, R., Birman, K.P., Vogels, W.: Astrolabe: A robust and scalable technology for distributed systems monitoring, management, and data mining. ACM Transactions on Computing Systems **21**(3) (2003)
84. Riabov, A., Liu, Z., Wolf, J., Yu, P., Zhang, L.: Clustering algorithms for content-based publication-subscription systems. In: Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS'02), Vienna, Austria (2002)
85. Riabov, A., Liu, Z., Wolf, J., Yu, P., Zhang, L.: New algorithms for content-based publication-subscription systems. In: Proceedings of the 23rd International Conference on Distributed Computing Systems (ICDCS'03), pp. 678–686 (2003)
86. Rjaibi, W., Dittrich, K., Jaepel, D.: Event Matching in Symmetric Subscription Systems. In: Proceedings of the 12th Annual IBM Centers for Advanced Studies Conference (CASCON '02) (2002)
87. Rowston, A., Kermarrec, A., Castro, M., Druschel, P.: Scribe: The design of a large-scale notification infrastructure. In: 3rd International Workshop on Networked Group Communication (NGC2001) (2001)
88. Rowstron, A., Druschel, P.: Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In: IFIP/ACM International Conference on Distributed Systems Platforms (Middleware 2001), pp. 329–350 (2001)
89. Segall, B., Arnold, D.: Elvin Has Left the Building: A Publish /Subscribe Notification Service with Quenching. In: Proc. of the 1997 Australian UNIX and Open Systems Users Group Conference (1997)
90. Segall, B., Arnold, D., Boot, J., Henderson, M., Phelps, T.: Content Based Routing with Elvin4. In: Proceedings of AUUG2K, Canberra, Australia (June 2000)

91. Shi, Y.S.: Design of overlay networks for internet multicast. D.Sc. thesis, Washington University (2002)
92. SIENA Web Site: http://www.cs.colorado.edu/users/carzanig/siena/
93. Sivaharan, T., Blair, G., Coulson, G.: GREEN: A Configurable and Re-configurable Publish-Subscribe Middleware for Pervasive Computing. In: Proceedings of DOA 2005 (2005)
94. Srivatsa, M., Liu, L.: Securing publish-subscribe overlay services with eventguard. In: CCS '05: Proceedings of the 12th ACM conference on Computer and communications security, pp. 289–298. ACM Press, New York, NY, USA (2005). DOI http://doi.acm.org/10.1145/1102120.1102158
95. Stoica, I., Adkins, D., Ratnasamy, S., Shenker, S., Surana, S., Zhuang, S.: Internet indirection infrastructure. In: First International Workshop on Peer-to-Peer Systems (2002)
96. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. In: Proceedings of ACM SIGCOMM (2001)
97. Tarkoma, S.: Efficient content-based routing, mobility-aware topologies, and temporal subspace matching. Ph.D. thesis, University of Helsinki, Department of Computer Science, Helsinki, Finland (2006). URL http://ethesis.helsinki.fi/julkaisut/mat/tieto/vk/tarkoma/
98. Tarkoma, S., Kangasharju, J.: On the cost and safety of handoffs in content-based routing systems. Computer Networks **51**(6) (2007). DOI 10.1016/j.comnet.2006.07.016. URL http://dx.doi.org/10.1016/j.comnet.2006.07.016
99. Terpstra, W.W., Behnel, S., Fiege, L., Zeidler, A., Buchmann, A.P.: A peer-to-peer approach to content-based publish/subscribe. In: Proceedings of the 2nd International Workshop on Distributed Event-Based Systems (DEBS'03) (2003)
100. Triantafillou, P., Economides, A.: Subscription summarization: A new paradigm for efficient publish/subscribe systems. In: ICDCS, pp. 562–571 (2004)
101. Voulgaris, S., Rivière, E., Kermarrec, A.M., van Steen, M.: Sub-2-sub: Self-organizing content-based publish subscribe for dynamic large scale collaborative networks. URL citeseer.ist.psu.edu/voulgaris06subsub.html
102. Wang, C., Carzaniga, A., Evans, D., Wolf, A.L.: Security issues and requirements for internet-scale publish-subscribe systems. In: the proceedings of the Hawaii International Conference on System Sciences (2002)
103. Wang, Y., Qiu, L., Achlioptas, D., Das, G., Larson, P., Wang., H.J.: Subscription Partitioning and Routing in Content-based Publish/Subscribe Networks. In: 16th International Symposium on DIStributed Computing (DISC'02) (October 2002)
104. Wang, Y.M., Qiu, L., Verbowski, C., Achlioptas, D., Das, G., Larson, P.: Summary-based routing for content-based event distribution networks. SIGCOMM Comput. Commun. Rev. **34**(5), 59–74 (2004). DOI http://doi.acm.org/10.1145/1039111.1039113
105. Yan, T.W., Garcia-Molina, H.: Index structures for selective dissemination of information under the boolean model. ACM Trans. Database Syst. **19**(2), 332–364 (1994)
106. Yan, T.W., Garcia-Molina, H.: The sift information dissemination system. ACM Trans. Database Syst. **24**(4), 529–565 (1999)
107. Yoneki, E., Bacon, J.: Content based routing with on-demand multicast. In: Proceedings of the 24th IEEE International Conference on Distributed Computing Systems, Workshop on Wireless Ad Hoc Networking (ICDCS - WWAN 2004), pp. 788–793 (2004)
108. Zhao, B.Y., Huang, L., Stribling, J., Rhea, S.C., Joseph, A.D., Kubiatowicz, J.: Tapestry: A Resilient Global-scale Overlay for Service Deployment. IEEE Journal on Selected Areas in Communications (2003)
109. Zhuang, S.Q., Zhao, B.Y., Joseph, A.D., Katz, R., Kubiatowicz, J.: Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination. In: 11th Int. Workshop on Network and Operating Systems Support for Digital Audio and Video (2001)

Glossary

Broker An application layer server. Brokers typically form an overlay network.

Content-based pub/sub A publish/subscribe model that supports the routing of messages or packets based on their content.

Event flooding A simple technique for propagating information by flooding it in the distributed environment.

Event matching The process of associating an event message with a number of subscriptions. For an event to match a subscription, it must satisfy the constraints defined in that subscription.

Event Service A logically centralized service that provides the publish/subscribe functionality.

Filter A constraint on a message. Typically filters are defined using stateless Boolean functions.

Filter-based routing A routing technique that uses filters to selectively propagate information towards subscribers. Typically, messages are not delivered to those parts of a network that have not expressed prior interest in receiving the messages.

Filter Merging An optimization technique that aggregates routing information, namely filters. The aggregation is typically based on logical rules or probabilistic techniques.

Gossiping event routing A probabilistic event routing technique that does not typically require a particular network structure, and aims to guarantee the delivery of messages in a highly dynamic environment.

Overlay network A network implemented on top of an underlying network, typically TCP/IP.

Publish/Subscribe A communication paradagim that consists of two types of entities: publishers and subscribers of information. Pub/sub introduces indirection by decoupling publishers and subscribers from each other.

Publish/Subscribe Mobility Support A family of mobility support protocols that aim to support the network mobility of either subscribers, publishers, or both. In this context, network mobility means that a subscribing or publishing endpoint relocates from one physical network attachment point to another. In order to support seamless information delivery, the publish/subscribe event routing topology needs to be adjusted due to this mobility.

Rendezvous-based Routing A routing technique that uses special network nodes, called Rendezvous Points, to coordinate event routing table updates and message propagation. Rendezvous-based routing is typically implemented using an overlay network.

Selective event routing A process that aims to reduce the size of the distributed routing tables and the protocol overhead by selectively propagating routing information and messages.

Topic-based Routing A publish/subscribe model that supports the routing of messages or packets based on a particular topic. This topic has to be agreed beforehand by the communicating entities.

Type-based Routing A publish/subscribe model that supports the routing of messages or packets based on a type definition.