Distributed Event Routing in Publish/Subscribe Communication Systems: a Survey

Roberto Baldoni[1], Leonardo Querzoni[1], and Antonino Virgillito[1]

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

Abstract. Distributed event routing has emerged as a key technology for achieving scalable information dissemination. In particular it has been used as preferential communication backbone within publish/subscribe communication system. Its aim is to reduce the network and computational overhead per event diffusion to a set (possibly large) of interested recipients. This paper introduces an unifying framework, namely a publish/subscribe architecture, that points out the functional decomposition between event-based routing layer, the overlay infrastructure layer and the network protocols layer. Hence the paper, firstly, surveys current algorithms for event based routing and possible overlay infrastructures in wired and mobile systems and, secondly, it discusses how and when single solutions at each level can be combined in the publish/subscribe architecture. Finally the paper positions existing publish/subscribe systems within the proposed architecture.

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) [1].

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 paper 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 paper 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 paper 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 paper we survey some of the most representative pub/sub systems, positioning them in the architectural model.

Being focused on scalable event routing and its relation with the underlying overlay network, this paper complements other surveys related to publish/subscribe systems such as [2], [3], [4] and [5]. The main aim of [2] has been indeed to position the pub/sub paradigm with respect to other communication paradigms, whereas [3] concentrated on software engineering aspects of a pub/sub system. Liu and Plale in [4] propose a survey of pub/sub systems considering overlay topology, matching algorithms and aspects such as reliability and security.

The paper is structured as follows. Section 2 presents a general overview of the pub/sub paradigm. Section 3 describes the various types of subscription models. Section 4 presents the architectural pub/sub model, discussing all the four layers. In Section 5 we focus on the event routing layer. Section 6 is a survey of some representative pub/sub systems. Section 7 concludes the paper.

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].

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

[1] The interested reader can refer to [2] for a more deep discussion on the publish/subscribe paradigm and subscription models.

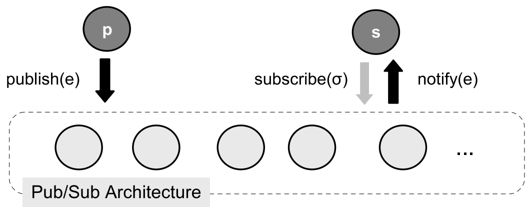


Fig. 1. High-level interaction model of a publish/subscribe system with its clients (p and s indicate a generic publisher and a generic subscriber respectively).

pub/sub system and viceversa (Figure 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* ($e \sqsubset f$).

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 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 [6], iBus [7], SCRIBE [8], Bayeux [9] and the CORBA Notification Service [10].

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 [11, 6]). 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 [10] 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 [12–14]. 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 [15, 16]). A complete specification of content-based subscription models can be found in [17]. Examples of systems that fall under the content-based category are Gryphon [18], SIENA [19], JEDI [20], LeSubscribe [21], Ready [22], Hermes [23], Elvin [24].

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 [25, 26]. 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 [27, 28] 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 [29]. 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 [30] 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 [31–33] 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 [34–36, 33]. The implementation of location-aware subscriptions requires the pub/sub system the ability to monitor the mobility of clients.

4 Architectural Model

In this section we describe the reference architectural model we use in our presentation. The architectural model is depicted in Figure 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).

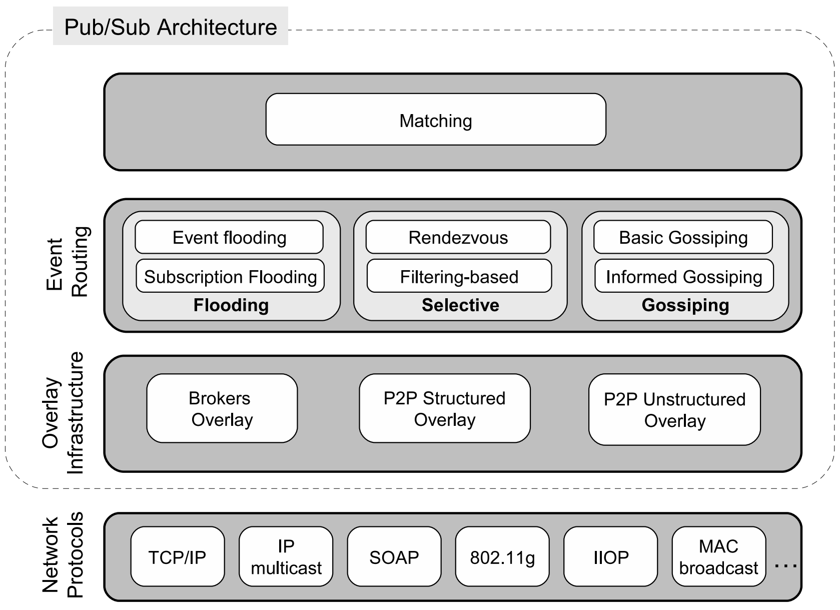


Fig. 2. Publish/Subscribe Architectural Model

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 [37–40]. The main drawback of IP multicast is in its lack of a widespread deployment [41, 42]. 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) [43, 44, 33, 34]. 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 [35, 45, 46]. 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. Moreover, mobile clients typically have to include the support for location-aware subscriptions.

Let us finally remark that in the rest of the paper we will focus on pub/sub systems realized over a MANET. One-hop mobile networks do not pose indeed important research issues being considered as an engineering of wired solutions. Pub/sub issues over MANET have to cope with constantly changing topologies and partitions that makes the problem of event routing non-trivial.

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 [47, 48].

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 [14, 49][2]

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 [6], Gryphon [18], SIENA [19], JEDI [20] and REDS [50], as well as in several event routing algorithms proposed in the literature [51, 52]. Apart from the routing protocols, that we analyze in Section 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).

[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.

Many contributions [53, 52] 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. [12] 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 [54], Chord [55], Tapestry [56] (unicast diffusion) or CAN [57], I3 [58] and Astrolabe [59] (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 [9] and Scribe [60], for what concerns topic-based systems, and Meghdoot, Hermes [61] and Rebeca [62], for what concerns content-based systems. Finally, we cite SelectCast [63], 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 [64]. 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 on the overlay graph to diffuse and to retrieve information associated with the nodes. This is due to the absence of a structure that facilitates event routing, which is difficult to maintain because of node dynamicity (see Section 5.2 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. Not many pub/sub systems have been proposed on the top of unstructured peer-to-peer overlay networks. Among them we cite [65], detailed in Section 5.3.

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 [43, 44, 66]. We detail this aspect in Section 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 [67]. These approaches have similarities to data management in the Bayou system [68] 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.

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 [69].

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 5 is entirely devoted to describing and comparing routing algorithms, as well as identifying which type of infrastructure is more suitable for each solution.

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 [70], 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 [71, 72, 14, 73]. Testing network algorithms ([74, 75, 15]) are based on a pre-processing of the set of subscriptions that builds a data structure (a tree in [74] and [75] or a binary decision diagram in [15]) 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 [76].

		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. Classification of Event Routing Algorithms

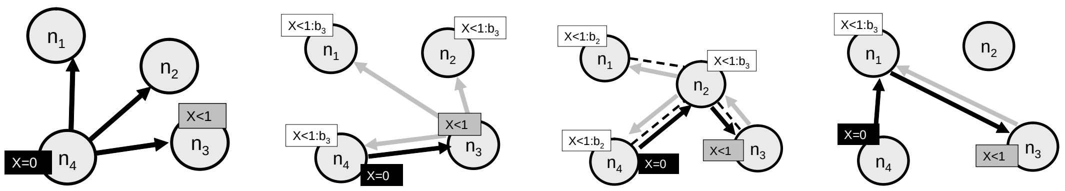
5 Event Routing

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, 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 5.2.



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

Fig. 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.

[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 [5].

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 3(a)). This algorithm 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 3(b)). However, both simulations studies ([77, 12]) and practical experiences ([13]) 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 [24], which was removed in a successive version ([13]), since it proved to be very costly. A recent work presenting a subscription flooding approach is MEDYM [78], an algorithm part of the DADI framework.

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 [79, 80] since it avoids the overhead due to the storage and update of event routing information.

Filtering-based Routing In *Filtering-based routing* [12] 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 [80, 5].

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 (see Section 6 for details).

The pseudo-code of filtering-based routing at a broker is presented in Figure 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 3(c), where the dashed lines represent connections at overlay level.

```
upon receive publish(event $e$) from node $x$
 $matchlist \leftarrow$ match($e$,$subscriptions$);
 send notify($e$) to $matchlist$;
 $fwdlist \leftarrow$ 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. 4. Pseudo code of Filtering-based routing

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 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 [61] and in [62, 81]. 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.

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* [82]. The pseudo-code of rendezvous-based routing is presented in Figure 5 (*subscriptions* list is defined as in Filtering-based routing), while an example is depicted in Figure 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 $i$
 $rvlist \leftarrow EN(e)$;
 if $i \in rvlist$ then
     $matchlist \leftarrow$ match($e$,$subscriptions$);
     send notify($e$) to $matchlist$;
 else
 send($e$) to $rvlist$;
upon receive subscribe(subscription $s$) from node $x$ at node $i$
 $rvlist \leftarrow SN(s)$;
 if $i \in rvlist$ then
     add $s$ to $subscriptions$;
 else
     send($s$) to $rvlist$;
```

Fig. 5. Rendezvous-based routing

Rendezvous-based routing has been introduced in [52], and recently many systems appeared following such a scheme (Scribe [8], Bayeux [9], Hermes [23], Meghdoot [83] and [82]). 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 mapping 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.

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 [69]. 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 [84]. 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 [85].

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 [86] and lead to high robustness, reliability and self-stabilization [87]. 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 [89, 88, 11, 65].

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 [89] (*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.

Finally let us remark that Costa and Picco in [65] 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 of 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.

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,

[5] Gossiping has been also used to improve delivery guarantee of a filtering-based event routing protocol in [88].

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

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 [90, 91, 67] or integrate with the classical MANET routing protocols such as MAODV [92, 93].

Huang and Garcia-Molina [43], Anceaume et al. [44] and Cugola et al. [66] 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 5.2 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 [90, 91, 67]. For example, [91] and [67] 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 [67] 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.

5.5 Event Routing for Sensor Networks

Two contributions were proposed ([48, 47]) adapting to the context of wireless sensor networks event routing solutions introduced for wired networks. Costa et al. in [48] propose a sensor-based implementation of their semi-probabilistic algorithm of [65], that exploits only broadcast for communication and introduces specific solutions for reducing the impact of packet collisions. Hall et al. in [47] adapt the content-based networking protocol of [79] to sensor networks, in the form of a routing protocol which extends the acyclic overlay used in [79] 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.

6 Surveying Publish/Subscribe Systems

In this section we specifically deal with real implementations of pub/sub systems. We take into account in detail the most popular pub/sub systems, in particular by specifying their characterizing features with respect to the general solutions presented above. Because of the large number of publish/subscribe systems proposed in the literature, this survey does not intend to fully cover all the different works. Rather, we decided to include in the presentation only those systems having a specific characterizations with respect to our general framework, such as original variations and combinations of event routing solutions and overlay infrastructures.

System	*Subscription Model*	*Network Protocol*	*Overlay Infrastructure*	*Event Routing*
TIB/RV	Topic	TCP/MAC bcast	Brokers	Filtering
Scribe	Topic	TCP	P2P Structured	Rendezvous
Siena, Gryphon, Rebeca	Content	TCP/UDP	Brokers	Filtering
Hermes	Content	TCP	Brokers	RVs/Filtering
Meghdoot	Content	TCP	P2P Structured	Rendezvous
DADI (Kyra)	Content	TCP	Brokers	RVs/Filtering
DADI (MEDYM)	Content	TCP/IP mcast	Brokers	Sub. Flooding
GREEN (WAN)	Various	TCP	P2P Structured	Rendezvous
GREEN (Mobile)	Various	802.11g	Unstructured	Gossiping

Table 2. Survey of Pub/Sub Systems

6.1 TIB/RV

TIB/RV [6] has been one of the first commercial systems to implement the publish/subscribe paradigm. TIB/RV is a topic-based system that relies on the abstraction of an event channel ideally connecting all subscribers interested in a same topic.

In TIB/RV, brokers are structured in a two-level hierarchical architecture. Brokers at the lowest level of the hierarchy are called *rendezvous daemon*. Each network host on which a publisher or a subscriber resides has to run such a daemon. Each daemon holds the subscriptions for the host on which it runs. Events are diffused through network-level broadcast.

Event diffusion spanning over a WAN is realized through a different type of brokers, namely *rendezvous router daemon*, constituting the higher level of broker hierarchy. Each local network is represented at wide-area level by a single router daemon, that receives all the events directed from the local to other networks and multicasts in the local network notifications received from other networks. Router daemons form an application-level network, in which daemons are connected in couples through TCP connections. Event routing is realized by building multicast trees among router daemons. Each daemon maintains a tree for each subject. A router daemon is added to a tree if there exists at least a subscriber for that subject in the local network represented by that daemon. In order to build trees, daemons have to know both the entire network topology and the current subscription configuration.

6.2 Scribe

An alternative approach for topic-based systems is the one proposed in *Scribe* [8], a research system designed at Microsoft Research. Scribe is built upon a structured overlay network infrastructure called *Pastry* [54], that allows to perform an efficient large-scale routing of messages in a application-level network of brokers[7]. Each broker in Pastry is assigned an unique identifier in the network and messages can be routed to a specific broker by simply specifying its identifier. Scribe is actually an application written using Pastry and represents a pub/sub interface for it. Subscription routing, event routing and notification routing are implemented in Scribe by leveraging Pastry's routing capabilities.

Scribe is based on a rendezvous-based routing algorithm over a structured overlay architecture. In the following we explain how the mapping policy is implemented. The basic idea is that each topic is assigned a random identifier and the Pastry node with the identifier closest to the topic becomes the target broker for that topic. A multicast tree is built for each topic, rooted at the corresponding target broker. When a new node subscribes for a subject, its subscription is routed by Pastry to the corresponding target broker, that updates the tree structure in order to include the new subscriber. When an event is published for a subject, event routing is performed through Pastry, by directly routing the event to the target broker for that subject. The target broker is simply addressed by the subject's identifier. When an event arrives at the target broker, matching reduces to identifying the correct multicast tree and notification routing is performed by diffusing the notification through such tree.

6.3 SIENA

A fundamental contribution to the research in content-based pub/sub is the SIENA system [19]. SIENA focuses on providing efficient and scalable notification routing over a wide-area network. SIENA is based on a brokers network architecture (deployed over a TCP or UDP transport layer) and it is the system where the filtering-based routing approach was introduced.

Thus event routing strictly follows our description in Section 5.2. Besides that, SIENA introduces techniques for constraining subscription propagation, by exploiting containment relationships among them: informally, a subscription σ_1 contains another subscription σ_2 if all events matching σ_2 also match σ_1. When a subscription update occurs, the new subscription is not propagated by a broker if this broker has already propagated a containing subscription before. Another technique to aid the event routing process introduced in SIENA is the advertisements. An advertisement is issued by a publisher to declare the set of events it is going to produce. Advertisements are also considered in building routing paths, to further reduce the set of involved brokers.

The SIENA algorithms have become a reference solution for the problem of routing content-based events and subscriptions in an application-level network (*content-based routing* problem). In [77], a general theoretical framework for content-based routing is proposed as well as some variants over the original SIENA algorithm and a performance evaluation.

6.4 REBECA

REBECA (acronym for Rebeca Event-Based Electronic Commerce Architecture - www.gkec.informatik.tu-darmstadt.de/rebeca) is an object-oriented notification service framework implemented in Java, providing a content-based addressing model.

[7] Another topic-based system with a similar approach is Bayeux [9], built over the overlay network infrastructure named Tapestry [56]

Rebecas topology is given by an acyclic graph of connected event brokers, communicating with each other in order to route notifications to clients. The system provides a generic routing engine that has been extended with different filtering-based algorithms besides event flooding, namely: a) simple routing (similar to subscription flooding, which allows events to be filtered at the producers), b) identity routing (which avoids forwarding a subscription identical to another one already forwarded), c) covering routing (similarly avoids forwarding a subscription that is covered by another one already sent), and d) merging routing (which lets a broker merge subscriptions of existing routing entries and forward only this merger). Rebeca also supports the usage of advertisements as an additional mechanism to further optimize notification routing, and can be used in conjunction with all the previous routing algorithms. Rebeca has been extensively used in research projects to explore toward different directions such as configurable systems, mobility support, concept-based addressing, security aspects, scoping, modularity, etc. Rebeca is available to the community as an open-source project.

6.5 Gryphon

Gryphon [94] is a content-based pub/sub system, developed at the IBM Watson research center. Gryphon is based on a brokers network architecture. The topology of the brokers' overlay is a tree whose vertex are actually clusters of nodes and edges are bundles of links. Replication of nodes and links is used for load-balancing and high-availability purposes. Publishers connect at the root brokers, while subscribers connect at leaves.

The event routing algorithm in Gryphon [53] belongs to the Filtering-based class: events are propagated downstream from publishers to subscribers. Each brokers propagates an event only if an interested subscriber is present downstream. The feature that differentiates the Gryphon algorithm from a plain Filtering-based routing is the guarantee of ordered and reliable delivery of events, that can be lost because of nodes and link failures. As in classical Filtering-based algorithm the decision about filtering an event at a broker or forwarding it downstream requires subscriptions to be propagated throughout the brokers network. The subscription propagation algorithm of Gryphon [95] ensures that subscription information is always consistent at each broker, despite of broker replication, failures, message losses and propagation over redundant links. Moreover, subscriptions are aggregated in order to save brokers memory and network bandwidth.

Besides being an actual system, Gryphon represents the reference framework for the research in content-based pub/sub carried out at IBM Watson. Not all the results in these papers are implemented in the actual system. However, they represent important steps in the evolution of the research in pub/sub systems. Relevant research results include a matching algorithm with sub-linear complexity [74], an efficient event routing protocol performing partial matching at each broker [96], efficient mapping of content-based subscription to network-level multicast [37–39], that we already commented in Section 4.1.

6.6 Hermes

Hermes [23] was presented as a pub/sub middleware rather than a simple system, because it encompasses also other functions such as type checking and security. In the context of our presentation, Hermes gathers in an interesting way several ideas from the systems described above.

Hermes is based on a rendezvous-based routing, with a type-based subscription model. It is implemented as a network of brokers but also exploits an overlay network infrastructure used as a facility for replacing failed nodes [61].

The mapping policy is realized considering the type of a subscription: a subscription is assigned to a broker whose identifier matches the hash of the type name. The mapping policy organizes subscriptions in coarse-grained clusters (types) and it may happen that the system contains more brokers than types, leaving some brokers not assigned to any type. At the same time, differently from SIENA, only a single copy of each subscription is retained in the system, at the corresponding target broker.

Event routing is implemented with SIENA-like algorithms. Paths are created for routing events belonging to a specific type to the corresponding target broker. Notification routing follows the same idea, with a diffusion tree for each type, rooted at the target broker and having each subscriber as a leaf. Content-based filtering of subscription is performed directly at recipients. Thus in Hermes there is only one spanning tree for each type, whereas in SIENA subscription routing builds in practice a spanning tree from each possible publisher to all subscribers.

6.7 Content-based Systems over Structured Overlays (Meghdoot)

Meghdoot [83] is one of the first examples of a content-based pub/sub system entirely based on a structured overlay infrastructure, with rendezvous-based event routing. With respect to Hermes it entirely exploits the structured overlay for communication between nodes, rather than creating its own broker network. Another difference with Hermes is that the subscription language in Meghdoot is a pure content-based one, where in Hermes the type allows an easy mapping, so that the realization of the mapping functions is less obvious.

Meghdoot exploits the CAN [57] structured overlay. CAN is based on the n-dimensional geometrical space. Physical nodes are assigned to zones in the space, having hyper-rectangular shape. CAN ensures that each point in the space always corresponds to a physical node. CAN allows to route messages by specifying a point in the space. The message is delivered to the node responsible for the zone in which the point falls.

The subscription language in Meghdoot allows subscriptions structured in n attributes, of numerical and string types, where constraints are in the general form of a range. A subscription is mapped to a $2n$-dimensional CAN point, whose coordinates are the bounds of each range constraints. An event maps to a CAN region spanning all the possible subscriptions that can map the event (event region), hence satisfying the mapping intersection rule. Event routing algorithm entirely exploits CAN routing: the event is first sent from the publisher to the rendezvous at the edge of the event region. From here, the event is forwarded to all the nodes inside the event region, with each node forwarding to its neighbors.

Other proposals following the same architectural approach have been presented in [97] and [82]. In particular, [82] differs from Meghdoot in the sense that it proposes a generic architecture for content-based pub/sub over structured overlays, which is independent from the specific infrastructure used. A set of 3 general stateless mappings is proposed for subscriptions and events. Chord is used as a reference infrastructure for experiments.

6.8 The DADI Project

The DADI (Discovery, Analysis and Dissemination of Information) project is a research project carried out at Princeton University focusing on large-scale content-based pub/sub. Though not being an actual system, the project outputs (namely Kyra and MEDYM) represent interesting contributions for what concern event routing.

Kyra [51] is based on a broker network architecture and can be considered as a mix of rendezvous-based and filter-based. The broker topology is built according to network proximity and brokers that are neighbors in the network are organized in clusters. Each cluster corresponds to a zone in the subscription space and one broker in the cluster is the rendezvous for the subscriptions and events falling in that zone. Filter-based routing is used to dispatch events inside a cluster, following an overlay tree built as the minimum spanning tree on the cluster and rooted at the rendezvous. The benefits of this scheme are the enhanced load-balancing and the reduced routing information with respect to a "flat" filter-based scheme. The drawbacks are those of rendezvous routing over broker networks: partitioning requires the previous knowledge of event and subscription distributions, limited support for string attributes, lack of support for dynamic reconfiguration.

A different approach is the one introduced in MEDYM [78]: each broker knows any other broker in the system (that is, the overlay is completely connected) as well as the subscription it stores (in the form of a unique subscription representing the sum of all the subscriptions on the server). Following our classification, MEDYM is based on a subscription flooding approach, where events can be filtered out directly at producers. Events matching at least one subscription are sent to subscribers through multicast. Multicast can be either performed through IP Multicast if available or with application-level multicast. In this latter case, the multicast tree is computed dynamically when the event is sent, based on the current state of the network.

6.9 Reconfigurable systems (GREEN and REDS)

Recently, two pub/sub systems, namely GREEN [33] and REDS [50] have been proposed, featuring reconfiguration capabilities. Both systems are based on a component architecture, where each component is responsible for a specific functionality (matching, event routing or overlay management). The system itself is actually the specification of a component framework enabling dynamic plugging of components inside the architecture. This allows such systems to adapt to various deployment contexts (from fixed to mobile networks, with both structured and unstructured overlay infrastructures) simply by implementing and plugging new specific components.

GREEN is organized into a two-layers structure, where a layer is a component framework in itself: the upper layer concerns the subscription language used. Supported languages are topic-based, content-based and context (proximity). The lower layer contains the components realizing the pub/sub overlay. These correspond to overlay infrastructure and event routing layer in our architectural model. The lower layer components include support for practically all the spanning from LAN with IP Multicast to WAN with broker networks or structured overlays, to wireless and mobile networks. GREEN supports dynamic reconfiguration at run-time, thanks to the fact that components are binary and can be substituted without recompiling the whole system.

7 Conclusions

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 paper 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, Leonardo Querzoni, 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 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 the paper.

The authors are partially supported by a grant of the IS-MANET/MIUR project.

References

1. 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. Volume 2584. Springer (2003)
2. Eugster, P., Felber, P., Guerraoui, R., Kermarrec, A.: The Many Faces of Publish/Subscribe. ACM Computing Surveys **35** (2003) 114–131
3. 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)
4. Liu, Y., Plale, B.: Survey of publish/subscribe event systems. In: Indiana University Computer Science Technical Report TR-574. (2003)
5. Bittner, S., Hinze, A.: A classification of filtering algorithms in content-based publish/subscribe systems. In: Proceedings of COOPIS 2005. (2005)
6. 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)
7. 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. (October 1999) 49–65
8. 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** (October 2002)
9. 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)
10. Object Management Group: CORBA event service specification, version 1.1. OMG Document formal/2000-03-01 (2001)

11. Baehni, S., Eugster, P.T., Guerraoui, R.: Data-aware multicast. In: Proceedings of the 2004 International Conference on Dependable Systems and Networks (DSN 2004). (2004) 233–242
12. Carzaniga, A., Rosenblum, D., Wolf, A.: Design and Evaluation of a Wide-Area Notification Service. ACM Transactions on Computer Systems **3** (Aug 2001) 332–383
13. Segall, B., Arnold, D., Boot, J., Henderson, M., Phelps, T.: Content Based Routing with Elvin4. In: Proceedings of AUUG2K, Canberra, Australia. (June 2000)
14. 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). (2001) 115–126
15. 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. (2001) 443–452
16. 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)
17. Muhl, G.: Generic Constraints for Content-Based Publish/Subscribe. In: Proceedings of the 6th International Conference on Cooperative Information Systems (CoopIS). (2001)
18. Gryphon Web Site: (http://www.research.ibm.com/gryphon/)
19. SIENA Web Site: (http://www.cs.colorado.edu/users/carzanig/siena/)
20. 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)
21. 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)
22. 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)
23. 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)
24. 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)
25. 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. (2000) 219–227
26. 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)
27. 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)
28. Eugster, P.T.: Type-based publish/subscribe. PhD thesis, EPFL (2001)
29. Eugster, P.T., Guerraoui, R.: Distributed programming with typed events. IEEE Software **21** (2004) 56–64
30. Cilia, M.: An Active Functionality Service for Open Distributed Heterogeneous Environments. PhD thesis, Department of Computer Science, Darmstadt University of Technology (2002)
31. Chand, R., Felber, P.: Xnet: A reliable content-based publish/subscribe system. In: 23rd International Symposium on Reliable Distributed Systems (SRDS 2004). (2004) 264–273
32. Chand, R., Felber, P.: Semantic peer-to-peer overlays for publish/subscribe networks. In: Parallel Processing, 11th International Euro-Par Conference (Euro-par 2005). (2005) 1194–1204
33. Sivaharan, T., Blair, G., Coulson, G.: GREEN: A Configurable and Re-configurable Publish-Subscribe Middleware for Pervasive Computing. In: Proceedings of DOA 2005. (2005)
34. 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)
35. 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). (2003) 103–122

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. 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). (2000) 185–207
38. 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)
39. 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). (2003) 678–686
40. 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. (2003) 67–74
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. Shi, Y.S.: Design of Overlay Networks for Internet Multicast. D.Sc. thesis, Washington University (2002)
43. 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)
44. 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). (2002) 74–81
45. 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** (2003) 1059–1071
46. 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). (2005) 103–116
47. Hall, C.P., Carzaniga, A., Rose, J., Wolf, A.L.: A content-based networking protocol for sensor networks. Technical Report CU-CS-979-04, Department of Computer Science, University of Colorado (2004)
48. 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)
49. Yan, T.W., Garcia-Molina, H.: The sift information dissemination system. ACM Trans. Database Syst. **24** (1999) 529–565
50. Cugola, G., Picco, G.P.: Reds: A reconfigurable dispatching system. In: Technical Report, Politecnico di Milano. (2005)
51. 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)
52. 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)
53. 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)
54. 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). (2001) 329–350
55. 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)
56. 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)
57. Ratnasamy, S., Handley, M., Karp, R., Shenker, S.: Application-level multicast using content-addressable networks. Lecture Notes in Computer Science **2233** (2001) 14–34

58. 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)
59. 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** (2003)
60. 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)
61. 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)
62. 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)
63. 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)
64. Pandurangan, G., Raghavan, P., Upfal, E.: Building low-diameter p2p networks. In: IEEE Symposium on Foundations of Computer Science. (2001) 492–499
65. Picco, G.P., Costa, G.: Semi-probabilistic publish/subscribe. In: Proceedings of 25th IEEE International Conference on Distributed Computing Systems (ICDCS 2005). (2005)
66. 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. (2003) 234–243
67. 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)
68. 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). (1997) 288–301
69. Baldoni, R., Beraldi, R., Piergiovanni, S.T., Virgillito, A.: On the modelling of publish/subscribe communication systems. Concurrency and Computation: Practice and Experience **17** (2005) 1471–1495
70. 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)
71. Yan, T.W., Garcia-Molina, H.: Index structures for selective dissemination of information under the boolean model. ACM Trans. Database Syst. **19** (1994) 332–364
72. Pereira, J., Fabret, F., Llirbat, F., Shasha, D.: Efficient matching for web-based publish/subscribe systems. Volume 1901 of LNCS., (Springer-Verlag)
73. Carzaniga, A., Wolf, A.L.: Forwarding in a content-based network. In: Proceedings of ACM SIGCOMM 2003. (2003) 163–174
74. 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). (1999) 53–61
75. Gough, K., Smith, G.: Efficient recognition of events in distributed systems. In: Proceedings of the ACSC-18. (1995)
76. 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)
77. Muhl, G.: Large-Scale Content-Based Publish/Subscribe Systems. Phd thesis, Department of Computer Science, Darmstadt University of Technology (2002)
78. 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)
79. Carzaniga, A., Rutherford, M.J., Wolf, A.L.: A routing scheme for content-based networking. In: Proceedings of IEEE INFOCOM 2004. (2004)
80. 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). (2002) 167–176

81. 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)
82. 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)
83. 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)
84. 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). (2002) 233–242
85. Eugster, P., Guerraoui, R., Kermarrec, A.M., Massoulie, L.: From Epidemics to Distributed Computing. IEEE Computer **37** (2004) 60–67
86. 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. (1987) 1–12
87. Birman, K.P., Hayden, M., Ozkasap, O., Xiao, Z., Budiu, M., Minsky, Y.: Bimodal multicast. ACM Transactions on Computer Systems (TOCS) **17** (1999) 41–88
88. 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)
89. Eugster, P.T., Guerraoui, R.: Probabilistic multicast. In: Proceedings of the 3rd IEEE International Conference on Dependable Systems and Networks (DSN 2002). (2002) 313–322
90. Huang, Y., Garcia-Molina, H.: Publish/subscribe tree construction in wireless ad-hoc networks. In: 4th International Conference on Mobile Data Management (MDM 2003). (2003) 122–140
91. Baehni, S., Chhabra, C., Guerraoui, R.: Mobility friendly publish/subscribe. In: EPFL Technical Report 200488. (2004)
92. 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). (2004) 788–793
93. Mottola, L., Cugola, G., Picco, G.P.: Tree overlays for publish-subscribe in mobile ad hoc networks. In: Technical Report, Politecnico di Milano. (2005)
94. Team, T.G.: Achieving scalability and throughput in a publish/subscribe system. In: IBM Research Report RC23103. (2004)
95. Zhao, Y., Sturman, D., Bhola, S.: Subscription propagation in highly-available publish/subscribe middleware. In: Proceedings of the ACM/IFIP/USENIX 6th International Middleware Conference (Middleware 2004). (2004)
96. 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)
97. Triantafillou, P., Aekaterinidis, I.: Content-based Publish/Subscribe over Structured P2P Networks. In: Proceedings of the 4th International Workshop on Distributed Event-Based Systems. (2004)