Semantic Peer-to-Peer Overlays for Publish/Subscribe Networks

Raphaël Chand[1] and Pascal Felber[2]

[1] Institut EURECOM, France
chand@eurecom.fr
[2] University of Neuchâtel, Switzerland
pascal.felber@unine.ch

Abstract. Existing publish/subscribe systems suffer from several drawbacks, such as the reliance on a fixed infrastructure of reliable brokers, or the lack of expressiveness of their subscription language. Most importantly, the challenging task of routing messages based on their content remains a complex and time-consuming operation, and often provides results that are just barely better than a simple broadcast.

In this paper, we present a novel approach to publish/subscribe that was designed to specifically address these issues. The producers and consumers are organized in a peer-to-peer network that self-adapts upon peer arrival, departure, or failure. Our publish/subscribe system features an extremely simple and efficient routing process and excellent scalability to large consumer populations, both in terms of routing and peer management overhead.

1 Introduction

Motivations. The publish/subscribe paradigm is well adapted to loosely-coupled, large scale distributed systems. However, in most traditional publish/subscribe systems, the routing process is a complex and time-consuming operation. It often requires the maintenance of large routing tables on each router and the execution of complex filtering algorithms to match each incoming document against every known subscription. The use of summarization techniques (e.g., subscription aggregation [1, 2]) alleviates those issues, but at the cost of significant control message overhead or a loss of routing accuracy.

In addition, content networks usually rely on a fixed infrastructure of reliable brokers, or assume that a spanning tree of reliable brokers is known beforehand. This approach clearly limits the scalability of the system in the presence of large and dynamic consumer populations. Finally, in most existing systems, the network topology has no relationships with the subscriptions registered by the consumers. As a consequence, the process of routing an event often involves a large number of routers, some of which have no interests in the event but only act as forwarders. The routing process is then only barely more efficient than a broadcast (which benefits from a much lower processing overhead).

To address these limitations, we have designed a publish/subscribe system that follows a radically different approach to content-based networking. First,

J.C. Cunha and P.D. Medeiros (Eds.): Euro-Par 2005, LNCS 3648, pp. 1194–1204, 2005.

the routing process in our system is extremely simple and has very low resource requirements. Second, by organizing peers based on their interests, content distribution is highly efficient as compared to broadcast. Finally, instead of relying on a fixed infrastructure of reliable brokers, our system is organized as a peer-to-peer network: join and leave operations, as well as peer failures, are taken care of at the design level with efficient peers management algorithms. We present in this paper two instantiations of our system that use the same routing protocol but differ by the way peers are organized. Experimental evaluation illustrates the various trade-offs that they offer in terms of efficiency and accuracy.

We would like to emphasize that we propose a new peer-to-peer approach for publish/subscribe, which relies on a system model that differs significantly from other peer-to-peer applications like file sharing. In particular, we assume that peers are well behaved and remain online for reasonably long periods of time, in the sense that the rate of message publication is higher than the frequency of peers' arrivals or departures. Our system provides mechanisms for organizing communities of peers that wish to exchange information using the publish/subscribe paradigm, without reliance on central servers or fixed infrastructures.

Related Work. Most publish/subscribe systems use an overlay network of event brokers to implement some form of distributed content based routing, most notably IBM Gryphon [3], Siena [1], Jedi [4] and XNet [5]. As previously mentioned, these systems suffer from various limitations in terms of extensibility, scalability, and cost. To address some of these issues, a few content-based systems based on peer-to-peer (P2P) networks have been recently proposed. In [6], the authors combine the notion of rendezvous nodes and content-based multicast to implement content based routing in a peer-to-peer environment. HOMED [7] is a peer-to-peer overlay for distributed publish/subscribe systems. Peers are organized in a logical binary hypercube according to their subscriptions. Routing is achieved by propagating the event along a multicast tree embedded in the hypercube. In [8], the authors also implement publish/subscribe in a peer-to-peer environment. The system is "data-aware" in the sense that it exploits information about registered subscriptions to build hierarchical structures. However, they differ from our approach in that the system is topic-based and the routing algorithm is based on multicast. Finally, some proposals have been made to implement content based routing on top of the Chord [9] P2P network. Examples of such systems are [10] and [11]. Unlike in our approach, they consider structured P2P networks and do not take advantage of semantic communities.

2 The Routing Process

Protocol. Our system is composed of a collection of peers. Each peer has registered certain interests that specify the types of messages that it is willing to receive. Each peer is connected with a set of other peers—its neighbors—with which it exchanges messages. We initially make the natural assumption that peers publish messages that match their own interests (we can easily relax this assumption, as will be discussed later). The routing protocol in our system is entirely based on the principle that every peer forwards a message to its neighbors if and only if the message matches its own interests. The routing process starts when a peer P publishes a message m. Since P is interested in m, it forwards it to all its neighbors. Routing then proceeds trivially as shown in Algorithm 1.

Algorithm 1 Routing protocol.

```
Receive message $m$ for the first time from neighbor $n$
if $m$ matches interests then
    Forward $m$ to all neighbors (except $n$)
end if
```

The intuition of the algorithm is to spread messages within a community that shares similar interests and to stop forwarding them once they reach the community's boundary. We emphasize the fact that the routing protocol is extremely simple and requires almost no resources from the peers. It consists of a single comparison and message forwarding operation. In addition, it requires no routing state to be maintained in the peers in the system. Each peer is only aware of its own interests and the identity of its direct neighbors, *not* their interests. The key to the protocol is the proper organization of the peers into semantic communities.

Accuracy. Clearly, the aforementioned process is not perfectly accurate and may lead to a peer receiving a message that it is not interested in—which we call a *false positive*— as well as missing a message that matches its subscriptions—a *false negative*. In other words, our system may deliver out-of-interest messages and may fail to deliver messages of interest. This is obviously due to the fact that a peer is not aware of the interests of its neighbors and forwards messages only based on its own interests. The challenge is thus to organize the peers so as to maximize routing accuracy. It should be noted that false positives are usually benign, because peers can easily filter out irrelevant messages, whereas false negatives can adversely impact application consistency.

Interest-Driven Peers Organization. Consider two neighbor peers P_1 and P_2. If P_1 and P_2 have registered close interests, it means that they are interested in similar types of messages. That is, if P_1 is interested in a message, it is likely that P_2 is also interested in it, and vice versa. It follows that neighbor peers should have close interests in order to minimize occurrences of false positives and false negatives in our system. In other words, we must organize peers based on the interests they registered: proximity in terms of neighborhood should reflect the proximity of the peers' interests.

To evaluate the proximity between two registered interests I_1 and I_2, a proximity metric must be used, that is, a function $f(I_1, I_2)$ that indicates how similar I_1 and I_2 are. Unfortunately, defining a good proximity metric is a challenging problem. It very much depends on the target application, on the language used to specify interests, and most of all on the messages being distributed in the system. The problem of interest proximity has been further discussed in [2].

3 Organizing Peers According to Containment

We now describe a hierarchical organization of the peers that yields no false negatives and only a limited amount of false positives. It uses a proximity metric based on the notion of *interest containment*, as specified in Definition 1. Note that the containment relation is transitive and defines a partial order. We define in a similar manner the relation of *interest equivalence.*

Definition 1. *Interest* I_1 contains *interest* I_2*, or* $I_1 \supseteq I_2 \Leftrightarrow (\forall$ *message* m*,* m *matches* $I_2 \Rightarrow m$ *matches* $I_1)$*.*

Interest I_1 *is* equivalent *to interest* I_2*, or* $I_1 \sim I_2 \Leftrightarrow (I_1 \supseteq I_2 \wedge I_2 \supseteq I_1)$*. That is:* $\forall$ *message* m*,* m *matches* $I_2 \Leftrightarrow m$ *matches* I_1*.*

The containment-based proximity metric, which we refer to as f_c, allows us to compare interests that share containment relationships and is defined as follows. Consider the set of all registered interests $\mathcal{I} = \{I_1, I_2, \cdots, I_n\}$ that contain I. Let $\{I_i, I_j, \cdots, I_m\} \subseteq \mathcal{I}$ be the longest sequence of non-equivalent interests such that $I_i \supseteq I_j \supseteq \cdots \supseteq I_m$. Then,

$$f_c(I, I') = \begin{cases} -\infty, & \text{if } I \not\supseteq I'; \\ \infty, & \text{if } I \sim I'; \\ |\{I_i, I_j, \cdots, I_m\}|, & \text{otherwise.} \end{cases}$$

Intuitively, the objective of this metric is to favor interests that are themselves contained in many other interests, i.e., that are very specific and selective. Note that this metric is not symmetric. The containment-based proximity metric can be used with any subscription language, provided that it defines a containment relationship. Of course, it applies best to subscription languages that are likely to produce subscriptions with many containment relationships. We wish to emphasize, however, that our routing protocol can be used with any other proximity metric, as we shall see in Section 4.

Network Description. Peers are organized in a *containment hierarchy tree*, based on the proximity metric f_c defined earlier. To simplify, we assume that each peer has expressed its interests by registering exactly one subscription (if that is not the case, the peer will appear multiple times in the hierarchy). The *containment hierarchy tree* is defined as follows. A peer P that registered subscription S is connected in the tree to a parent peer P_a that registered subscription S_a if S_a is the subscription in the system closest to S according to the proximity metric f_c. Given the definition of the metric f_c, this means that S_a is the deepest subscription in the tree among those that contain S. When we have more than one peer to choose from, we select as parent the peer that has the lowest number of children in order to keep the tree as balanced as possible. Because of Definition 1, peers that have registered equivalent interests in the system are organized in specialized, balanced subtrees with limited degree that we call *equivalence trees*. From the perspective of other peers in the system, an equivalence tree is considered as a single entity represented by its root node, which is positioned in the *containment hierarchy tree* using the rules described above. Non-equivalent children of the peers in an equivalence tree are always connected to its root. To interconnect top level peers that do not share containment relationships with each others, we introduce an artificial node that we refer to as the *root node*. This node is purely virtual and is implemented by simply connecting top-level peers with each other through "sibling" links. A simple containment hierarchy tree is illustrated in Figure 1. The equivalent peers P_8, P_9 and P_{10} are organized in the *equivalence tree* rooted at P_8. Note that both P_2 and P_4 contain P_3, but P_2 has a greater depth and is hence a better parent. Similarly, P_6 is connected to P_5 rather than P_1.

Impact on the Routing Process. From Algorithm 1 and the fact that peers are organized in a containment hierarchy tree, it follows that the paths followed by a message form a content distribution tree, in which inner nodes are true positives and leaves are either false positives or leaves in the tree topology. Consequently, routing is *efficient* in terms of bandwidth usage. Besides, there are no false negatives in our system. We wish to point out that false positives can only be avoided by having each peer know about its neighbors' interests, which conflicts with our design guidelines. Finally, the construction of the containment hierarchy tree topology enables us to minimize the occurrence of false positives with uniform subscription and message workloads. Indeed, the fact that a peer P has for parent the peer of highest possible depth that contains it means that a message m has a greater chance of being discarded on the way from the root node to P. A simple example is illustrated in figure 1, where peer P_5 publishes message D. The path followed by D is highlighted by the arrows.

Maintaining the Containment Hierarchy Tree. We have implemented several peers management algorithms to maintain the containment hierarchy tree when peers dynamically join and leave the system. We now briefly discuss their basic principles and most relevant features.

The join algorithm aims at inserting a new peer P with subscription S in the tree topology. Consequently, the system is first probed to find adequate containment or equivalence relationships between S and the other registered subscriptions. This can be done by recursively propagating *join* messages in the hierarchy tree. It is important to note that a *join* message usually traverses only a fraction of the tree, very much like regular messages. As a result of the probing phase, P joins the tree by connecting to a parent that is either an equivalent peer, if any, or a peer of highest depth whose subscription contains S. Next, P proceeds to the *reorganization* phase, which might lead to moving some existing peers so as to become P's children. Indeed, when P has connected to a parent in the tree, some other peers may now be closer to P than their actual parent in the

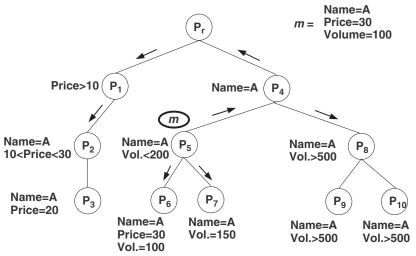

Fig. 1. A simple publish/subscribe system for stock quotes with participants organized in a *containment hierarchy tree*. The subscription registered by a peer is represented next to it.

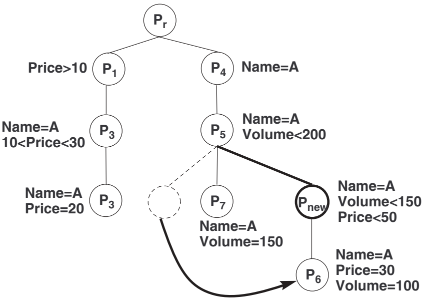

Fig. 2. Peer P_{new} has been inserted in the network with P_5 as its closest peer. Peer P_6 is reorganized as P_{new}'s child because the latter is a better parent than P_5.

tree. The *reorganization* phase introduces significant overhead in the system, in particular because it requires additional propagations of join messages. As a consequence, we have implemented three different flavors of the join algorithm. The first variant of the algorithm *always* performs all possible reorganizations. The second variant of the algorithm *never* performs any reorganizations. Finally, the third variant of practical relevance *periodically* performs reorganizations (only a given percentage of peers are reorganized). The different variants reach different compromises between joining complexity and accuracy of the hierarchy tree.

When peer P with registered subscription S wishes to leave the system—or when it fails—each of its children has to be reconnected to another parent in the tree. If P is part of an equivalence tree, then we simply perform a *leaf promotion* (a leaf downstream P is promoted to P's position). If P is not part of an equivalence tree, then the leave algorithm consists in reconnecting P's children to their grand-parent. It follows that every peer needs to know its grand-parent (or several ancestors for increased fault-tolerance); this is achieved with trivial modifications to the join algorithm and negligible additional control traffic. Although extremely simple, this recovery technique may cause the accuracy of the containment hierarchy tree to degrade over time. This is due to the fact that P's parent may not be the closest peer in the system for P's children. In addition, P's parent may suffer from the increased number of connections that it has to manage. To address those issues or in case P's parent has also failed, P's children can look for another replacement parent by executing the join algorithm, typically starting from some ancestor, at the price of higher overhead. Note that, if we wish to maintain an optimal tree, additional peers among P's descendants might need to re-evaluate their position as well if P's departure has decreased their depth.

Scalability Issues. The instantiation of our system using the containment-based metric organizes peers in tree topologies. It follows that high-level peers receive a high number of messages. As these peers have very "broad" interests, it is not unnatural that they receive a high percentage of published messages: they are interested in those messages. They are also more exposed to control messages from the peer management algorithms, but this traffic can be reduced by confining the join and reorganization procedures within selected subtrees. The most serious scalability issue comes from the fact that high-level peers may have a large number of neighbors to forward messages to (recall that both the routing process and the peers management algorithms are straightforward and demand very little resources). To address this problem, we have performed slight modifications to our original protocol to reduce by a great deal the bandwidth utilization at the peers. Because of space requirements, we shall only briefly introduce these techniques. Informally, the principle of the improved scheme consists in connecting the children of a node with "sibling" links in a double-linked list. When a peer receives a message, it forwards it to its two neighboring siblings and one of its children, chosen uniformly at random. This approach dramatically reduces the bandwidth requirements of peers that have a large number of children, but also slows down the propagation of messages in the system. It is therefore desirable to use it only for overloaded peers.

4 Organizing Peers According to Similarity

As previously mentioned, the routing protocol used to disseminate messages does not make specific assumptions about the proximity metric used to organize the peers in semantic communities. We now present a generalization of the containment-based proximity designed to alleviate two of its limitations: (1) its poor applicability to subscription language and/or consumer workloads with little or no containment relationships, and (2) its tree topologies that may be fragile with dynamic consumer populations. This generalization is based on the general principle of interest similarity.

Similarity Metric. We first define the notion of interest similarity as follows.

Definition 2 (Interest similarity). *Consider two interests I_1 and I_2. Let $\mathcal{I}$ be the universe of all possible interests. We define the similarity between I_1 and I_2, noted $Sim(I_1, I_2)$, as a function from $\mathcal{I}^2$ in the interval $[0, 1]$ that returns the probability that a message m matching I_1 also matches I_2.*

We then define our proximity metric based on interest similarity, which we refer to as f_s:

Definition 3 (Proximity metric f_s). $f_s : \mathcal{I}^2 \mapsto [0, 1]$:

$$f_s(I_1, I_2) = \frac{Sim(I_1, I_2) + Sim(I_2, I_1)}{2}$$

Note that the proximity metric f_s is symmetric, that is, if I_1 is close to I_2 according to f_s, then I_2 is equally close to I_1. However, the similarity function is a priori *not* symmetric.

Network Description. We now briefly describe the hierarchical organization of peers based on the proximity metric f_s. A peer P with registered interest I chooses a set of n neighbors P_i, which are the n peers in the system with interests closest to I according to f_s (in case of equality, the peers with less connections are chosen). In turn, P can be chosen by some other peers as one of the n best peers according to f_s (such that I is amongst the n interests in the system closest to their subscription according to f_s).

This approach effectively organizes the peers in "interest communities," i.e., groups of peers that share similar interests. Because of the definition of the similarity function and the proximity metric f_s, this organization optimizes routing accuracy by minimizing the number of false positives and negatives exchanged by neighbor peers. To maintain good connectivity between the communities and prevent some of them from being closed (because their interests do not compare with the other communities' interests), P also chooses r neighbors at random in the system, in addition to the n peers selected with f_s. Routing proceeds as described in Section 2. Peer management algorithms are also very similar to those presented earlier, with a few differences discussed in [12]. However, the routing algorithm can be enhanced to have even better control on false positives and false negatives. For instance, we can add an indulgence factor γ that allows a peer to forward a message even if it is not interested in it. The process, which may be performed only γ times per message, is expected to reduce the false negatives ratio, notably by improving the transfer of messages between communities. Another improvement is to add a random neighbor forwarding probability ρ, which controls the probability for a peer P to actually forward a message to its r random neighbors. The base case, $\rho = 100\%$, produces fewer false negatives but more false positives; lower values of ρ have the opposite effect.

Obviously, if $n + r > 1$, the peers are organized in graphs instead of trees. We can also observe that, if we set $n = 1$, $r = 0$ and we define $Sim(I_1, I_2) = 1$ if $I_1 \supseteq I_2$ and $Sim(I_1, I_2) = 0$ otherwise, peers are organized using a containment-based metric similarly to the topology of Section 3.

The organization of peers in graphs rather than trees benefits from several advantages. It has better connectivity and is hence more resilient to failures and frequent arrivals or departures. Also, it has better flexibility and offers higher scalability since the traffic load is more evenly distributed amongst the peers. Finally, this model can be applied to any subscription languages and consumer workloads even if the subscriptions share little or no containment relationships.

5 Performance Evaluation

To test the effectiveness of our publish/subscribe system, we have conducted simulations using real-life document types and large numbers of peers. We propose an evaluation of our system when using both the containment and similarity metrics presented earlier. We are mostly interested in studying the routing process in the system. Indeed, we have seen that the cost for its extreme simplicity is that it induces a certain inaccuracy in terms of false positives and negatives but that an efficient topology enables to minimize their occurrence. In this evaluation, we quantify the accuracy of our system experimentally. In-depth evaluation of other aspects of our system is available in [12].

Peers in our system register their interests using the standard XPath language to specify complex, tree-structured subscriptions. Documents are XML documents. To evaluate our system when using the similarity metric, we have implemented a proximity metric for XML documents and XPath subscriptions. Because of space limitations, the metric is not detailed in this paper (a description can be found in [12]).

Containment Metric. We first focus on the system when peers are organized in a containment hierarchy according to the proximity metric f_c. We have seen that this topology enables to suppress all occurrences of false negatives. As a consequence, we aim at quantifying experimentally the number of false positives generated by the routing process in the system. For that purpose, we proceeded as follows. We first simulated networks of different sizes, with each version of the *join* algorithm presented in section 3, by sequentially adding peers with randomly-generated subscription (we used a reorganization rate of 10% for the *join* version with periodic reorganization). We then routed $1,000$ random documents by injecting them at the root node.[1] For each document, we computed the false positives ratio as the percentage of peers in the system that received a message that did not match its interests. The results, shown in Figure 3, were obtained by taking the average of $1,000$ executions.

We observe that the average false positives ratio remains small, typically less than 10% in most cases, and decreases exponentially with the size of the consumer population. This is due to the efficiency of the tree topology. By organizing peers based on their interests, documents are filtered out as soon as they reach the boundary of the community of interested consumers. The efficiency of the tree topology improves with the size of the consumer population because of the increasing number of containment relationships shared between the peers. Besides, we computed that on average, and independently of the consumer population, the percentage of uninterested peers in the system is 75%, which illustrates the benefits of our routing protocol over a broadcast. Unsurprisingly, join algorithms that reorganize the peers more frequently produce network topologies that have a lower false positives ratio. As explained in section 3, this is directly related to the number of reorganizations that are performed by each algorithm. However, the differences are very small and the benefits of the slight increase in accuracy may not justify the additional overhead of the reorganization process.

Similarity Metric. We now study the accuracy of the system when peers are organized in a graph according to the proximity metric f_s based on subscrip-

[1] Note that the number of false positives would not be affected when injecting the messages at another node than the root.

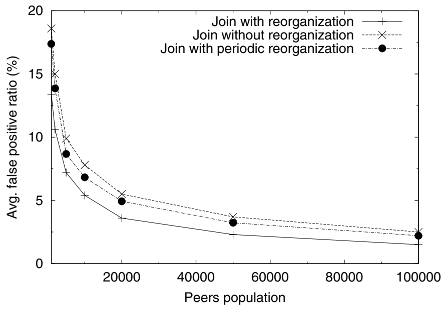

Fig. 3. False positives ratio for networks of different sizes.

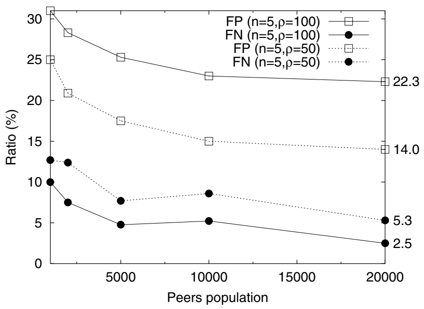

Fig. 4. False positives and false negatives ratios for networks of different sizes.

tion similarities. Since this topology does not prevent the occurrence of false negatives, we are interesting in quantifying the accuracy of our system both in terms of false positives and false negatives. For that purpose, we proceeded as in the case of the metric based on containment. We first generated networks of different sizes, using a value of $n = 5$ for the number of proximity neighbors and $r = 1$ for the number of random neighbors. We then injected random documents and quantified the routing accuracy. We measured the false positives ratio as the percentage of the peers in the system that received a message that did not match their interests, and the false negatives ratio as the percentage of peers interested in a message that did not receive it. We experimented with random neighbor forwarding probabilities ρ of 100% and 50%. Results are shown in Figure 4.

We first observe that the average false negatives ratio remains small, typically less than 5%, which shows that on average, for a given document, only a small fraction of the population of interested consumers does not receive it. The false positives ratio, while significantly higher, still remains at reasonable values, typically around 25%. We also remark that, as expected, a lower value of the parameter ρ favors the false positives ratio over the false negatives ratio. For a value of $\rho = 50\%$, the false positives ratio improves significantly (14% for $20,000$ peers), at the cost of a slight increase of the false negatives ratio (5% for $20,000$ peers). Finally, all performance metrics decrease with the size of the consumer population, which shows that the routing accuracy globally improves with the consumer population. This can be explained by the fact that, in larger populations, peers are able to find better neighbors according to the proximity metric f_s and hence reduce the occurrence of false positives and false negatives.

6 Conclusion

We have designed a novel publish/subscribe system, based on the peer-to-peer paradigm, that specifically address some of the limitations of existing systems. In particular, our network does not rely on a dedicated network of content routers, nor on complex filtering and forwarding algorithms: it features an extremely simple routing process that requires almost no resources and no routing state to be maintained at the peers. The price to pay for this simplicity is that routing may not be perfectly accurate, in the sense that some peers may receive some messages that do not match their interests (false positives), or fail to receive relevant messages (false negatives). By organizing the peers according to adequate proximity metrics, one can limit the scope of this problem. We have proposed a containment-based proximity metric that allows us to build a bandwidth-efficient network topology that produces no false negatives and very few false positives. We have also developed a proximity metric based on subscription similarities that yields a more solid graph structure with negligible false negatives ratios and very few false positives. As part of our ongoing research, we are studying refinements of our proximity metrics that take into account additional factors such as physical proximity or link bandwidth, in order to minimize latency and maximize throughput.

References

1. Carzaniga, A., Rosenblum, D., Wolf, A.: Design and Evaluation of a Wide-Area Event Notification Service. ACM Transactions on Computer Systems **19** (2001)
2. Chan, C.Y., Fan, W., Felber, P., Garofalakis, M., Rastogi, R.: Tree Pattern Aggregation for Scalable XML Data Dissemination. In: Proceedings of VLDB. (2002)
3. 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 ICDCS. (1999)
4. Cugola, G., Nitto, E.D., Fugetta, A.: The JEDI event-based infrastructure and its application to the development of the opss wfms. IEEE Transactions on Software Engineering **27** (2001) 827–850
5. Chand, R., Felber, P.: A scalable protocol for content-based routing in overlay networks. In: Proceedings of NCA, Cambridge, MA (2003)
6. Perng, G., Wang, C., Reiter, M.: Providing content based services in a peer to peer environment. In: Proceedings of DEBS, Edinburgh, UK (2004)
7. Choi, Y., Park, K., Park, D.: Homed: A peer-to-peer overlay architecture for large-scale content-based publish/subscribe systems. In: Proceedings of DEBS, Edinburgh, UK (2004)
8. Baehni, S., Th, P., Guerraoui, E.: Data-aware multicast. In: Proceedings of the 5th IEEE International Conference on Dependable Systems and Networks. (2004)
9. Stoica, I., Morris, R., Karger, D., Kaashoek, F., Balakrishnan, H.: Chord: A scalable peer-to-peer lookup service for internet applications. In: Proceedings of ACM SIGCOMM. (2001) 149–160
10. Terpstra, W., Behnel, S., Fiege, L., Zeidler, A., Buchman, A.: A peer-to-peer approach to content-based publish/subscribe. In: Proceedings of DEBS, San Diego, USA (2003)
11. Triantafillou, P., Aekaterinidis, I.: Content-based publish-subscribe over structured p2p networks. In: Proceedings of DEBS, Edinburgh, UK (2004)
12. Chand, R., Felber, P.: Semantic Peer-to-Peer Overlays for Publish/Subscribe Networks. Technical report, Institut EURECOM (2005)