Institut EURECOM
Department of Corporate Communications
2229, route des Crètes
B.P. 193
06904 Sophia-Antipolis
FRANCE

Research Report N^o 123 — RR-04-123

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

R. Chand, P.A. Felber

November 1, 2004

[a]Institut Eurécom's research is partially supported by its industrial members: Bouygues Télécom,Fondation d'entreprise Groupe Cegetel, Fondation Hasler, France Télécom, Hitachi, ST Microelectronics, Swisscom, Texas Instruments, Thales

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

Raphaël Chand
Institut EURECOM
raphael.chand@eurecom.fr

Pascal Felber
University of Neuchâtel
pascal.felber@unine.ch

***Abstract*— Content-based publish/subscribe systems offer a convenient abstraction for data producer and consumers, as most of the complexity related to addressing and routing is encapsulated within the network infrastructure. Most existing publish/subscribe systems suffer, however, from several drawbacks. They are usually based on a fixed infrastructure of reliable brokers, which cannot easily be modified or extended as the population of the producers and consumers evolves. Further, 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. Most importantly, 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.

I. Motivations

In content-based publish/subscribe systems, messages are routed on the basis of their content and the interests (subscriptions) of the message consumers. This form of communication is well adapted to loosely-coupled distributed systems with large consumer populations, with diverse interests, wide geographical dispersion, and heterogeneous resources (e.g., CPU, bandwidth). Several techniques have been proposed to implement content routing, with various trade-offs in terms of algorithmic complexity, runtime overhead, or bandwidth utilization.

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 a 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, 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 off at the design level with efficient peers management algorithms.

II. 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 networks (P2P) 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. Events are first guided to a rendezvous node before being disseminated along a multicast tree of interested subscribers.

HOMED [7] is a peer-to-peer overlay for distributed publish/subscribe systems. Peers are organized in a mesh-like structure based on their interests, by assigning to each peer an identifier that represents its subscription. Peers are then organized in a logically binary hypercube according to their identifiers. Routing is achieved by propagating the event along a multicast tree embedded in the hypercube.

Two proposals have been made to implement content based routing on top of the Chord [8] P2P network. In [9], event propagation and filter updates are similar to the broadcast mechanism proposed in [10], but are attenuated by the use of filters on the edges of the graph and by taking advantage of covering relationship. In [11], the event schema is a set of typed attributes. Each node stores pieces of information regarding some subscriptions, which are called *subscription ids*. The main idea is to store a subscription id at the nodes of the graph selected by appropriately hashing the values of the attributes of the subscription. Routing is achieved by fetching the subscription ids of the nodes selected by hashing the values of the attributes in the event.

Although these systems benefit from the traditional advantages of P2P networks, like their self-organization or limited diameter, they still suffer from a number of drawbacks, such as the lack of expressiveness of their subscription language (e.g., due to restrictions enforced by consistent hashing). Most importantly, these systems still require a significant amount of routing information to be kept at the brokers, and the routing process is in most cases complex and time-consuming.

III. The Routing Process

A. 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 assume 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.

The intuition of the algorithm is to spread messages within a community that share similar interests and to stop forwarding them once they reach the community's boundary. We emphasize on the fact that the routing protocol is extremely simple and requires almost no resources from the peers. It consists of a single filtering operation and message forwarding. In addition to that, 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 ids of its direct neighbors, *not* their interests.

Algorithm 1 Routing protocol

```
1: Receive message m for the first time from neighbor n
2: if m matches interests then
3:     Forward m to all neighbors (except n)
4: end if
```

B. Accuracy

Clearly, the aforementioned process is not perfectly accurate and may lead to a peer receive 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.

C. 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 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].

As a consequence, we are using a special case of proximity metric based on the notion of *interest containment* specified in Definition 1. Note that the containment relation is transitive and defines a partial order.

Definition 1 (Containment): 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

The relation of *interest equivalence*[1] in defined in a similar manner:

Definition 2 (Equivalence): Interest I_1 is *equivalent* to interest I_2, or $I_1 \sim I_2$, if and only if $I_1 \supseteq I_2 \wedge I_2 \supseteq I_1$. 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.

The containment-based proximity metric can be used with any subscription language, provided that it defines a containment relationship. We wish to emphasize, however, that our routing protocol can be used with any other proximity metric.

IV. Containment-based Hierarchy

We now describe a hierarchical organization of the peers based on containment that yields no false negatives and only a limited amount of false positives.

A. 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.

We now consider the special case of peers that have registered equivalent interests in the system. From definition 2, it follows that if peer P_1 and P_2 are neighbors, P_1 would never deliver false positives to P_2, and vice versa. It is then clear that equivalent peers in the system should always be neighbors in the topology. As a consequence, in our tree topology, we organize equivalent peers together in specialized, balanced subtrees 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 at its root.

Given that the containment relation is transitive, a peer contains all its descendants in the subtree rooted at itself. Since there may not be a peer in the system that contains all the others, we introduce an artificial node that interconnects all top-level peers and that we refer to as the *root node* or the *producer node*. This node is purely virtual and can be implemented by simply connecting top-level peers with each other through "sibling" links.

A simple containment hierarchy tree is illustrated in Figure 1.

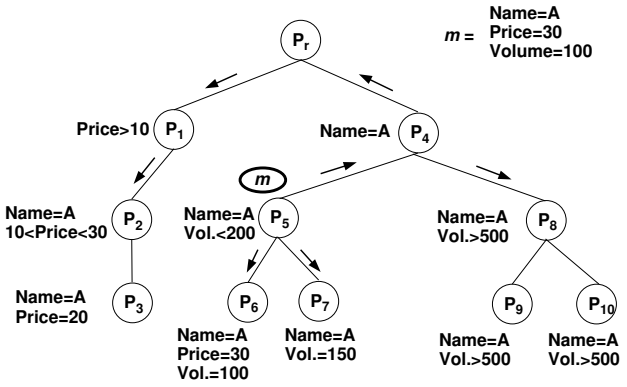

Fig. 1. A simple publish/subscribe system for stock quotes with participants organized in a *containment hierarchy tree*. The subscription registered by a peer is represented next to it. The equivalent peers P_8, P_9 and P_{10} are organized in the *equivalence tree* rooted at P_8. Note that both P_2 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.

[1]We intentionally do not use the term "equality" because some subscriptions languages allow interests to be formally different and yet match the same set of messages.

B. Impact on the routing process

The fact that peers are organized in a containment hierarchy tree has for consequence that the paths followed by a message form a content distribution tree, i.e., a spanning tree rooted at the root node of the containment tree. From algorithm 1, it follows that its leaves are either false positives (peers that are not interested in the message) or peers that are leaf nodes in the tree topology. The inner nodes of the content distribution tree are all true positives (peers interested in the message and that received it). 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.

As a consequence, it is easy to show that there are *no* false negatives in our system. Indeed, if peer P is interested in document D, then so are all its ancestors in the tree. It directly follows from algorithm 1 that P receives D.

Further, the containment hierarchy tree topology enables us to minimize the occurrence of false positives. 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. If message m traverses all of P's ancestors, it means that these peers were interested in the message and there is a good chance that P is also interested in it.

Finally, since only the leaves of the spanning tree followed by a message may be false positives, routing is *efficient* in terms of bandwidth usage. 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.

C. 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 (more details can be found in [12]). We would like to point out that these algorithms are executed only when a peer joins or leaves the system, which we assume to happen at a much lower frequency than the publication of messages. As previously discussed, the algorithm executed for routing such messages is trivial and extremely efficient.

a) Join algorithm: Let P be a new peer that wishes to join the system and register subscription S. In order to insert P in the tree topology, the system is first *probed* to find adequate containment relationships between S and the other registered subscriptions. For that purpose, P sends a *join* message to the root node of the system, which is then propagated recursively downward the tree and processed at each encountered peer that has a subscription containing S. It is important to note that a *join* message usually traverses only a fraction of the tree, very much like regular messages. Peer P then uses the results of this probing phase to actually join the tree. It first connects 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 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 to obtain the most accurate containment hierarchy tree and minimize the occurrence of false positives, at the cost of a higher complexity. The second variant of the algorithm *never* performs any reorganizations. It has the lowest complexity but at the same time produces less accurate containment hierarchies with poor load-balancing properties. Finally, a third variant of the algorithm *periodically* perform reorganizations: during the probing phase, a *join* message has a given probability of being propagated further for reorganization purposes. It reaches a compromise between join complexity and routing accuracy. An example of the join algorithm with reorganizations is illustrated in Figure 2.

b) Leave algorithm: 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*: we look for a leaf in the subtree rooted at P and promote it to P's position. If P is not part of an equivalence tree, there is no trivial replacement parent for P's children. In fact, since peers are stateless in the system, the best potential replacement for P known by the peers is P's own parent. Therefore, the leave algorithm simply consists in reconnecting P's children to their grandparent. Although extremely simple, this algorithm 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 order to avoid this problem, P's children can look for a better replacement parent by executing the join

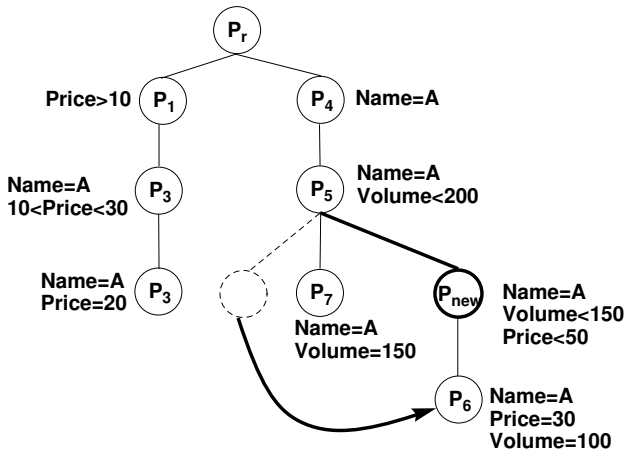

Fig. 2. Peer P_{new} has been inserted in the network with P_5 as its closest peer. Peer P_6 is reorganized as P_{new}'s child because the latter is a better parent than P_5.

algorithm, at the price of a higher complexity. Note that, if we wish to maintain an optimal tree, additional peers among P's descendants might need to be reorganized if P's departure has decreased their depth (see [12] for more details).

The failure of a peer is handled in the same manner as a departure. The only additional complexity is to ensure that each peer knows its grand-parent. This can be achieved with trivial modifications to the join algorithm and negligible additional control traffic. If that is not the case, a peer that cannot directly re-connect to its grand-parent has to execute the join algorithm.

V. 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 are mostly interested in studying the routing process of our system. Indeed, we have seen that the cost for its extreme simplicity is that it induces a certain number of false positives, but that an efficient topology enables to minimize their occurrence. The purpose of this evaluation is to 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 [13] to specify complex, tree-structured subscriptions. We have generated realistic subscription workloads using a custom XPath generator that takes a Document Type Descriptor (DTD) as input and creates a set of valid XPath expressions based on a set of parameters that control: the maximum height h of the tree patterns; the probabilities p_* and $p_{//}$ of having a wildcard (*) and ancestor/descendant (//) operators at a node of a tree pattern; the probability p_λ of having more than one child at a given node; and the skew θ of the Zipf distribution used for selecting element tag names. For our experiments, we generated sets of tree patterns of various sizes, with $h = 10$, $p_* = 0.1$, $p_{//} = 0.1$, $p_\lambda = 0.1$, and $\theta = 1$. We employed the widely-used NITF (News Industry Text Format) DTD [14] as the input DTD of our XPath generator. The events published are XML NITF documents. We used an XML generator to generate XML documents with an average size of 22 tag pairs.

To quantify experimentally the number of false positives generated by the routing process in our system, we proceeded as follows. We first simulated networks of different sizes, with each version of the *join* algorithm presented in section IV, by sequentially adding peers with randomly-generated subscription. We then routed $1,000$ random documents by injecting them at the root node.[2] 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.

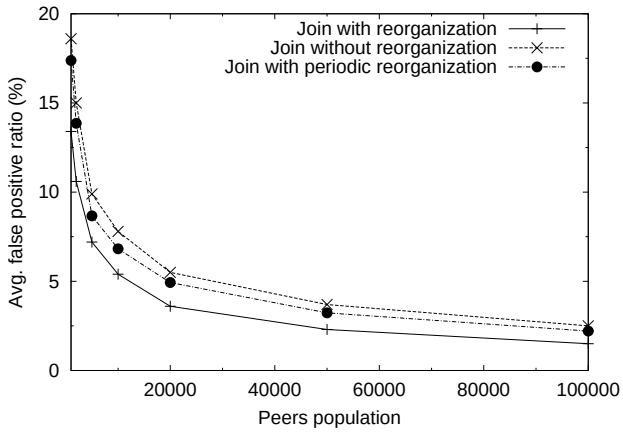

Fig. 3. False positive ratio for networks of different sizes.

We first observe that the average false positives ratio remains small, typically less than 10% in most cases. This shows that our system delivers documents to all interested peers with only a very small fraction of false positives, that is, with good routing accuracy. In comparison, we have computed that a broadcast would yield false positives ratios over 75%, independent of the consumer population. In addition, we observe that the average false positives ratio decreases exponentially with

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

the size of the consumer population, which means that the routing accuracy improves as additional peers join the system. These excellent results are 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.

Unsurprisingly, join algorithms that reorganize the peers more frequently produce network topologies that have lower false positive ratio. As explained in section IV, 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.

To summarize, the limited set of experimental results discussed here demonstrate that our routing protocol, which is almost as simple as a broadcast, enables us to distribute documents to *all* interested consumers with only a small fraction of false positives by carefully organizing the peers according to their interests.

VI. Conclusion

We have designed a publish/subscribe system to specifically address 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. 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] A. Carzaniga, D. Rosenblum, and A. Wolf, "Design and Evaluation of a Wide-Area Event Notification Service," *ACM Transactions on Computer Systems*, vol. 19, no. 3, pp. 332–383, 2001.

[2] C.-Y. Chan, W. Fan, P. Felber, M. Garofalakis, and R. Rastogi, "Tree Pattern Aggregation for Scalable XML Data Dissemination," in *Proceedings of VLDB*, Aug. 2002.

[3] G. Banavar, T. Chandra, B. Mukherjee, J. Nagarajarao, R. Strom, and D. Sturman, "An efficient multicast protocol for content-based publish-subscribe systems," in *Proceedings of ICDCS*, May 1999.

[4] G. Cugola, E. D. Nitto, and A. Fugetta, "The JEDI event-based infrastructure and its application to the development of the opss wfms," *IEEE Transactions on Software Engineering*, vol. 27, no. 9, pp. 827–850, Sept. 2001.

[5] R. Chand and P. Felber, "A scalable protocol for content-based routing in overlay networks," in *Proceedings of NCA*, Cambridge, MA, Apr. 2003.

[6] G. Perng, C. Wang, and M. Reiter, "Providing content based services in a peer to peer environment," in *Proceedings of DEBS*, Edinburgh, UK, May 2004.

[7] Y. Choi, K. Park, and D. Park, "Homed: A peer-to-peer overlay architecture for large-scale content-based publish/subscribe systems," in *Proceedings of DEBS*, Edinburgh, UK, May 2004.

[8] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proceedings of ACM SIGCOMM*, 2001, pp. 149–160.

[9] W. Terpstra, S. Behnel, L. Fiege, A. Zeidler, and A. Buchman, "A peer-to-peer approach to content-based publish/subscribe," in *Proceedings of DEBS*, San Diego, USA, June 2003.

[10] S. El-Ansary, L. Alima, P. Brand, and S. Haridi, "Efficient broadcast in structured p2p networks," in *Proceedings of IPTPS03*, Berkeley, USA, Feb. 2003.

[11] P. Triantafillou and I. Aekaterinidis, "Content-based publish-subscribe over structured p2p networks," in *Proceedings of DEBS*, Edinburgh, UK, May 2004.

[12] R. Chand and P. Felber, "Semantic Peer-to-Peer Overlays for Publish/Subscribe Networks," Institut EURECOM, Tech. Rep., 2004.

[13] W3C, "XML Path Language (XPath) 1.0," http://www.w3.org/TR/xpath, Nov. 1999.

[14] I. P. T. Council, "News Industry Text Format," http://www.nitf.org/.