A Scalable Interest-oriented Peer-to-Peer Pub/Sub Network

Kaoutar Elkhiyaoui[*] , Daishi Kato[†], Kazuo Kunieda[†], Keiji Yamada [†] and Pietro Michiardi[*]
[*] Eurecom, 2229, route des Cretes, Sophia Antipolis, France
[†] C&C Innovation Research Laboratories, NEC corporation
8916-47, Takayama-Cho, Ikoma, Nara 630-0101, Japan

***Abstract*—Publish/subscribe represents a new paradigm for distributed content delivery. It provides an alternative to address-based communication due to its ability to decouple communication between the source and the destination. However, it has remained a challenge to devise a scalable overlay supporting expressive content-filtering while satisfying the desirable requirements large distributed systems should fulfill. Our goal is to build an efficient P2P publish/subscribe network where only interested nodes are involved in event dissemination, and the amount of overhead generated by network discovery and membership management is small. In order to do so, we use a Bloom filter based mapping scheme to map IDs to nodes' interests, in addition to a new interest proximity metric to forward events and to build nodes' routing tables. As for network discovery we propose a new approach we call "shared interest approach". Our scheme ensures an upper bound of routing tables size that only depends on the size of the ID digest. To evaluate the algorithms proposed in this work we conducted simulations in both static and dynamic settings.**

I. INTRODUCTION

Publish/subscribe systems have received a lot of attention in the last years as they allow efficient, distributed and selective content delivery to a potentially large set of users. In such systems, users register subscriptions representing their interests in content while publishers inject events which are delivered to the matching subscribers.

There are two common types of publish/subscribe systems:

- *Topic-based*, which rely on a set of predefined topics to which subscribers register their interests: all messages related to a particular topic are broadcast to all registered users;
- *Content-based*, which allow subscribers to specify any filter over the entire content. A data event specifies values for a set of attributes associated with the event. Subscribers thus, register their interests in form of filters that are used by the system to deliver relevant events to the subscribers.

In this paper we focus on content-based publish/subscribe systems. Many applications require content-based publish/subscribe systems with fine grained expressiveness: for example, real-time stock quotes notification, Internet games and sensor network applications, to name a few. However, the implementation of such systems has remained a challenging issue.

Most content-based systems employ an overlay network of event brokers, which support rich subscription languages (e.g. SIENA [1], [2]). However, they commonly have two drawbacks. Firstly, a broker should maintain large routing tables. Indeed, every broker can be an intermediate relay on the paths of an event dissemination tree and should match each incoming event against every known subscription. Secondly, these systems require static overlay networks where the brokers are highly reliable and under administrative control, or assume the entire broker set to be known beforehand [3]. Scalability and reliability issues affecting content-based schemes have been addressed in the literature using system design inspired from the peer-to-peer (P2P) paradigm. Several implementations of content-based systems have been investigated in the literature, for instance Meghdoot[4], Mirinae [5], or HOMED [6]. Although these proposals address scalability and reliability issues, they whether have low overhead but involve not interested nodes in event dissemination [5], [6], [4], or they engage only interested nodes but generate a large amount of overhead (e.g. [7] where nodes use gossiping for membership management and routing table construction).

In this paper, we propose a new peer-to-peer content-based publish/subscribe scheme based on structured overlays. Our system aims at involving only interested nodes in event dissemination while ensuring a low overhead. In order to do so, we map nodes' interests to their identities (ID) using Bloom filters, and we use a novel proximity metric. This metric is used to cluster nodes according to their subscriptions' similarity. Therefore, events will be directly posted to the proper cluster where they are going to be disseminated efficiently. Indeed, our scheme ensures an upper bound of routing table size that only depends on the size of a node's ID. Furthermore, application overhead is reduced thanks to a new approach to network discovery.

The remainder of the paper is structured as follows: we present related works in Section II. Section III describes the design and the algorithms used by our system. We present a simulation-based evaluation of our system in Section IV and an analytical approach to evaluate application overhead in Section V. We conclude in Section VI.

II. RELATED WORK

The first implementations of content-based publish/subscribe systems used a network of event brokers to implement distributed content based routing: SIENA[2] and KYRA [8]. Although these approaches can support rich subscription languages, they have two main limitations. First of all, they require static networks which lead to un-optimized network topology. In other words, the network topology should cope with the changing nodes' interests in order to reduce network congestion and minimize routing depth. This means that for an optimized design, the network of the brokers should be dynamic. Secondly, in these approaches a broker keeps a large amount of routing information and generates a considerable amount of overhead in order to perform routing and to minimize notifications relaying. A broker needs to keep track of the changing state of its clients as they issue new or cancel subscriptions so that it reflects perfectly its clients interests. Although summarization using Bloom filter and aggregation using *covering* relation and *merger* are currently used to reduce notification overhead, a leaving node could generate a lot of overhead since she has to forward all the subscriptions she covers.

Other implementations rely on a peer-to-peer architecture in order to achieve self organization and robustness. In a peer-to-peer system all participants act as subscribers and publishers but, in addition, they also route notification among themselves. Some approaches implement content-based routing on top of DHTs. Terpstra et al. [9] used Chord [10] combined with filter-based routing algorithms (merging and covering) in order to attenuate the overhead generated by event broadcast. A variant based on CAN [11] was implemented in [4]. The nodes build a multidimensional DHT and maintain information about the coordinates of their zones and store coordinate information of their neighboring zones. The idea behind these schemes is to have a rendez-vous node for each event. Rendez-vous nodes act as an entry point to a distinct overlay network composed by the group of interested nodes. Other approaches aim at clustering nodes semantically using an interest proximity distance to route the events introduced into the overlay and to build routing tables. Some implementations intend to have a mesh-like structure for event dissemination [5], [6] and use the hamming distance combined with a hypercube overlay to route and disseminate events published by different nodes. In [7], nodes maintain semantic links to nodes with which they share some interests. Moreover, [7] uses gossip algorithms for membership management and to provide nodes with random links that represent a partial view of the overlay to ensure connectivity.

Our proposal is also based on the semantic approach. We aim at clustering nodes based on their interests: in our system events are forwarded using a new interest proximity metric while application overhead is reduced through a new mechanism for network discovery.

III. Overlay design and event dissemination

This section outlines our content-based publish/subscribe system. Our goal is to organize nodes semantically in a manner that only interested nodes in event e will forward it while minimizing the overhead due to membership management and network discovery.

In the remainder of this section we will make extensive use of the following definitions:

Definition 1. ***Filters:*** *a filter denotes the set of subscriptions issued by a given subscriber.*

Definition 2. ***Coverage:*** *a filter F_1 covers F_2, iff $N(F_1) \subset N(F_2)$ where $N(F)$ is the set of notifications that match the filter F.*

Definition 3. ***Mergers:*** *merger operation consists of deriving new filters from existing ones such that each new filter covers the set of filters it was generated from.*

We now describe the method to assign a "semantic" ID to a node, which is inspired by [5]. In our system, we partition the event space Ω into cells c_i of a regular grid. The process of partitioning the event space depends on the publish/subscribe application. For instance, in a stock quote application, the partitioning could correspond to a price/company's name partitioning.

Specifically, consider a set $S_n = \{c_i, c_i \cap S \neq \emptyset\}$. We use k independent hash functions h_1, ..., h_k, each with range 1 to d. The bits at position $h_1(c_i)$, ..., $h_k(c_i)$ in ID_n are set to 1 for each cell $c_i \in S_n$. As an event has a single cell c_e, its ID is set to 1 at positions $h_1(c_e)$, ..., $h_k(c_e)$. Figure 1 illustrates the process of generating a semantic ID for a node.

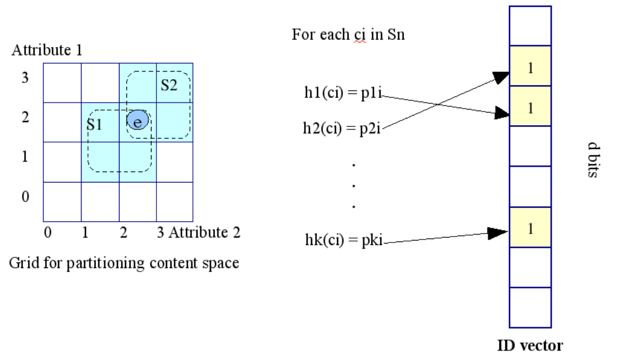


Fig. 1. Process to compute the semantic identifier of a node.

Assigning IDs using this approach renders the semantic clustering easier, as the similarity between two nodes can be estimated using the distance between two IDs. Moreover, this ID fulfills a very important property: if a node N is covered by another node M, her ID_M subsumes all 1s of ID_N.

Since node IDs obtained with this method might not be unique, we could concatenate an additional vector to distinguish nodes whose original ID collide. Instead, we organize nodes with the same ID in a cluster that will be transparent to other nodes that have a different ID. Hereafter, we refer to this ID by $ID_{semantic}$.

Moreover, we assign to each node a *random ID* uniformly drawn from a large identifier space: this ID comes in addition to her $ID_{semantic}$.

A. Building the routing tables

The main challenge in peer-to-peer publish/subscribe systems is how to build a routing table that would ensure no false negatives and at the same time involve only interested nodes in event dissemination. To do so, we use an interest proximity metric which is the product of the inverse of an affinity and the hamming distance normalized by the size of the $ID_{semantic}$ digest d. The distance between two nodes N and M in the network graph G is:

$$d(N, M) = \frac{d_{hamming}(N, M)}{d \times A(N, M)} \qquad (1)$$

$A(N, M)$ represents the *affinity* between nodes N and M and is computed according to the following expression:

$$A(N, M) = \frac{|S_n \cap S_m|}{min(|S_n|, |S_m|)} \qquad (2)$$

where S_n refers to $\{c_i, c_i \cap S \neq \emptyset\}$ and c_i is the i^{th} cell of the grid obtained by partitioning the event space S and the subscriptions issued by N. Furthermore, $|S_n|$ refers to cardinality of the set S_n.

The distance defined in Equation 1 allows a given node N to connect to the nodes that cover her interests with the smallest hamming distance. Formally, given two nodes N and M, N connects to M when:

$$M \in G \text{ , M covers N and } \rightarrow d_{hamming}(N, M) = min\{d_{hamming}(N, K), K \in G\}$$

We note that when two nodes N and M do not share any interests, the distance $d(N, M)$ will be infinite:

$$if S_n \cap S_m = \emptyset \Rightarrow d(N, M) = \infty$$

In our system, nodes are organized in a containment hierarchy based on covering relationship. Hence, every node N in the network has three types of bidirectional links: *covering links*, which correspond to links to nodes that cover N, *covered links* which refers to links N keeps to nodes she covers, and *neighbors links* that correspond to links N keeps to nodes with which she shares part of her interests. In the following, we present how a node in the overlay picks her neighboring nodes:

- ***Covering links:*** A node N connects first to the closest node in term of hamming distance which covers her interests, this latter is the parent of N.
- ***Covered links:*** N goes through the nodes she knows in increasing order of the random ID (looping when she reaches the maximal sequence ID) and selects a node only if she intersects N's interests at some region not yet covered by the already selected covered nodes. This process is then repeated in decreasing order.
- ***Neighbors links:*** N keeps links to nodes with which she shares a part of her interests. The process of picking these links is identical to the covered links.

Algorithms 1 and 2 illustrate how covering and covered links are created, where we introduce the following notation:

- nodes_to_add(N) denotes all nodes that N has discovered and used to build her routing table.
- $N \supset M$ denotes that N covers M interests. $N \subset M$ denotes that N's interests are covered by M. Similarly $N \not\supset M$ means that N does not cover M's interests.
- $\sqcup\{N_1, ..., N_k\}$ denotes the mergers of N_1 ... N_k

Algorithm 1 Building the overlay -Covering Nodes-

```
for all  $N \in G$ do
    for $i \in nodes\_to\_add(N)$ do
        if $i \supset N$ then
            if
              $d_{hamming}(i, N) < d_{hamming}(covering\_node(N), N)$
             then
                $covering\_node(N) \leftarrow i$
            end if
        end if
    end for
end for
```

Algorithm 2 Building the overlay -Covered Nodes-

```
for all  node N $\in G$ do
    Initialize covered_nodes(N)
    Sort nodes_to_add(N) in increasing order of random ID
    for node i $\in$ nodes_to_add(N) do
        if $n \supset i \wedge \sqcup(covered\_nodes(N)) \not\supset (N)$ then
            if $i \supset$ some of $N$'s interests not covered yet by
            $covered\_nodes(N)$ then
                covered_nodes(n).add(i)
            end if
        end if
    end for
    sort nodes_to_add(N) in decreasing order of random ID
    for  node i $\in$ nodes_to_add(N) do
        if $n \supset i \wedge \sqcup(covered\_nodes(N)) \not\supset (N)$ then
            if $i \supset$ some of $N$'s interests not covered yet by
            $covered\_nodes(N)$ then
                covered_nodes(n).add(i)
            end if
        end if
    end for
end for
```

Our overlay construction mechanism ensures a published event to be delivered to all interested nodes with high probability. Furthermore, the routing table size is upper-bounded, as derived in the following proposition:

Proposition 1. *The routing table size in the overlay has an upper bound of $2 \times d + c$, where c is a constant parameter referring to the number of covering links a node N can have and d is the size of the $ID_{semantic}$ digest.*

Proof: If node N has n bits set to 1 then she will have $d-n$ neighbors when Algorithm 1 loops over ID_{random} in increasing sequence order and another $d-n$ neighbors links when Algorithm 1 loops over ID_{random} in decreasing sequence order, at most.

Moreover, N will have n covered links when Algorithm 2 loops over ID_{random} in increasing sequence order and another n covered links when Algorithm 2 loops over ID_{random} in decreasing sequence order, at most.

It is also clear that N has c covering links.

Therefore, N will have $2d+c$ entries in her routing table, at most. ∎

B. Event dissemination

In this work, we cluster the overlay network semantically: hence, every node connects to neighbors with shared interests. When an event reaches a matching node N, N relays the message to her neighbors that match the event. Our system differentiate between two types of messages:

- ***Multicast***: When a node N receives a *Multicast* message, it sends a *Multicast* message to her covered nodes and neighbors nodes that match the event.
- ***Forward***: Upon the receipt of a *Forward* message, a node N sends a *Forward* message to her covering and covered nodes, as well as to her neighbors nodes that match the event.

Unlike the works in [5], [6] that rely on a technique to make node IDs unique (as the uniqueness of the semantic IDs cannot be guaranteed), we cluster nodes with the same semantic ID and to organize them into a logical ring[1]. Each ring will have the $ID_{semantic}$ identifier of the member nodes. A leader labelled *primary node* is assigned to each ring: the primary node acts as a relay point between the nodes on the ring and the outer nodes. Therefore, the cluster is transparent to the outer neighbors that will only point to the leader. When the primary node receives an event which she is interested in, she forwards the event on the ring. Leader election is based on joining time: the first node joining a cluster is automatically elected as a primary node. If a primary node fails or leaves, the node that joined the cluster after the failing or leaving leader is elected as the next clsuter leader.

C. Membership management

In current peer-to-peer publish/subscribe systems, network connectivity and network discovery is achieved by space splitting and gossip-based membership management. The use of the former approach leads to engaging nodes in disseminating events they are not interested in with high probability. The latter approach allows nodes to maintain random views for membership management but generates a large amount of overhead due to the periodic exchange of views between different nodes.

Our system relies on a new approach we called *shared interest approach*. In this approach, all nodes have a common subscription. This common subscription renders possible to find a route between any two nodes in the network which allows nodes joining the network to find their closest neighbors semantically. This common subscription can just be presented as a fixed bit that is set to 1 in $ID_{semantic}$ of all nodes.

Join: When a node N joins the network, she contacts a bootstrap node that is already a member of the system. In this work we gloss over the details of how system bootstrap is achieved: for example, a list of well-known bootstrap nodes could be published on a separate communication channel. The bootstrap node routes the join query using the distance we defined earlier. The join mechanism takes several steps until the routing table of N converges. The number of these steps depends on the number of bits that are set to 1 in the $ID_{semantic}$. If we assume that subscriptions are uniformly random, the number of these steps will be on average $\frac{d}{2}$ where d is the size of $ID_{semantic}$.

Once N receives the first join reply, she will build her routing table based on the one she receives from the replying node, and then she will send another join query but this time she will advertise a new ID which represents the interests that are not covered yet by her current neighbors links.

When a node M replies to the join query issued by N she proceeds as following:

- if N covers M, M checks if N is closer than some of her covering links.
- if M covers N, M checks if N covers part of her interests not covered yet by her covered links.
- if N shares just a part of M's interests, M checks if N covers part of her interests not covered yet by her neighbors links.

In each of these cases, M updates her routing table.

If there are nodes in the network which have the same $ID_{semantic}$ as the joining node N, N will join the cluster defined by her $ID_{semantic}$ and copy the routing table from one of the nodes that has the same $ID_{semantic}$.

Leave: When a node N leaves the network or fails, the nodes that point to N will update their routing table based on the routing table of N. These nodes will use merger and covering relationships to update their routing table.

We implement a heart beat mechanism in order to detect failed nodes. When a node N fails, the first node detecting the failure notifies all the nodes pointing to N which we call *incoming links*. These incoming links are piggybacked in the heart beat messages.

IV. Evaluation

We built a discrete event-based simulator, which does not model packet loss and assumes unlimited bandwidth along all the links.

Our work focuses on a stock quote application. The events in a stock quote application are generated by various stock exchanges where trading occurs and the subscribers are clients interested in the price of the stocks they trade. Without loss of generality, events in our simulation are mapped to a 2-dimensional event space. In fact, there are well-known

[1]Note that the logical ring is not a DHT.

methods to map a multidimensional space to one dimensional space using a space filling curve [5].

In our simulations, an event corresponds to the value of these attributes (stock-name, price) and a subscription corresponds to (stock-name, low price, high price).

Metrics: In our evaluation, we will focus on the following metrics:

- *Delivery depth:* this metric accounts for the number of hops required by a node to receive events she is interested in. We evaluate this metric as it mirrors the delay that an event takes to be disseminated;
- *Routing table size:* this metric measures the size of routing tables stored at each node. Small routing table sizes are preferred as they indicate the ability of a publish/subscribe network to scale with system size;
- *False positives ratio:* A false positive is defined as an event received by a node not interested in it. As we partition the event space and we use Bloom filters, this metric allows us to evaluate the amount of overhead our approach generates.

Parameter space: With the goal of collecting statistics on the average delivery depth and on the routing table sizes, we ran simulations while varying the network size. We assume a uniform, random distribution of users' subscriptions. In all the experiments we conducted the $ID_{semantic}$ size is 1024 bits.

Furthermore, we evaluate the impact of the distribution of users' subscriptions on false positives and routing table sizes by varying its skewness. We first simulate uniformly random subscriptions. Then we focus on subscriptions distributed according to a Pareto law x^{-a}: we vary the distribution skewness by varying the coefficient $a \in [1, 4]$. For these experiments we ran simulations in an overlay of 4000 nodes in which every node publishes at most 10 events

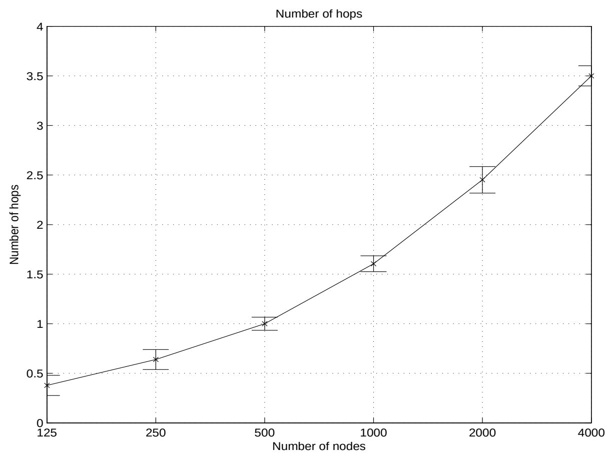
In order to gain statistical confidence in our results, we conducted 10 simulation runs for each experiment.

A. Static settings

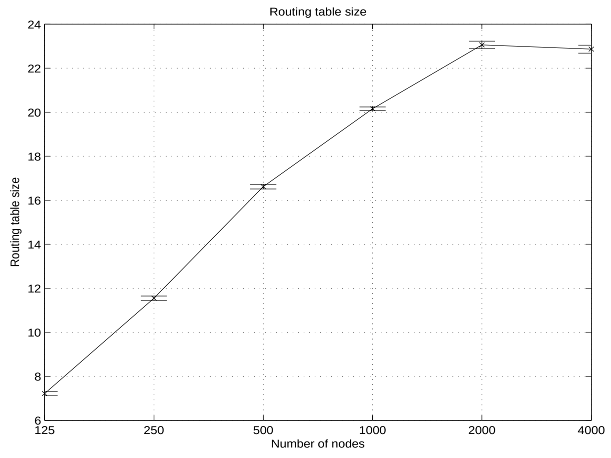
We ran simulations while varying the network size. We assume a uniformly random distribution of subscriptions.

Figure 2.(a) shows the average delivery depth we measured as the system scale increases: we observe that the average delivery depth growth can be roughly approximated to a logarithmic growth. Figure 2.(b) illustrates that the average routing table size stabilizes with system scale. Indeed, when the size of the network is small, nodes are not able to build complete routing tables whose mergers cover their interests. Hence, when the size of the network grows the size of the routing tables do so. When the size of the network becomes large, the nodes are able to build routing tables that cover their interests and thus, the size of the routing table will not vary drastically.

Figure 3 shows that the percentage of false positives decreases with subscriptions popularity. Figure 4 shows that the process of building routing tables size depends on the distribution of interests when the size of the network is



(a) Average delivery depth in static settings.



(b) Routing table size in static settings.

Fig. 2. Evaluation of our system in a static setting with uniformly distributed subscriptions and with a size of $ID_{semantic}$ of 1024 bits

large and that decreases with subscriptions popularity. These observations indicate that our system would prove effective when considering realistic system sizes, which can be safely assumed to be large.

B. Dynamic settings

We now study the impact on delivery depth and routing table size of dynamic settings. We assume nodes join the network at random point of time.

In this work we do not simulate node departures since our main focus is on the scalability of the overlay network, presented by the average delivery depth and the size of routing table. Moreover, as the incoming links of a node are piggybacked in the heartbeat messages as discussed in Section III, the nodes in the network will be able to update their routing table whenever one of their neighbors leaves or fails.

As the Figure 5.(a) shows, the size of the routing tables is larger than in the static setting: indeed, a node N does not have global knowledge of the network when it builds her routing table, hence the routing tables are not optimal. As the

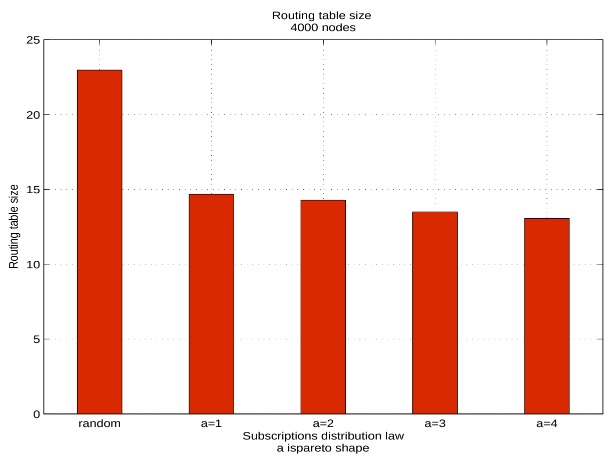


Fig. 3. Routing table size as a function of the subscriptions law.

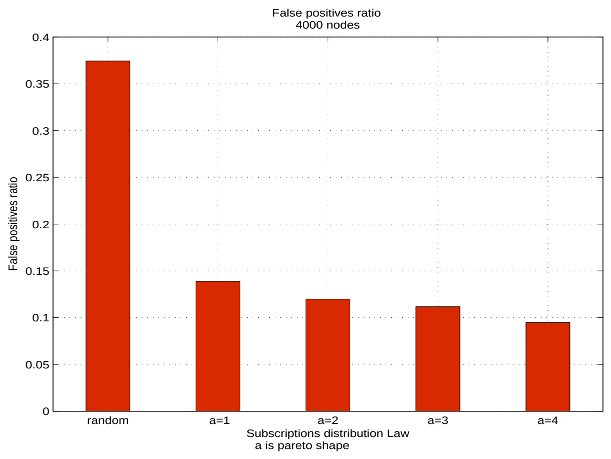
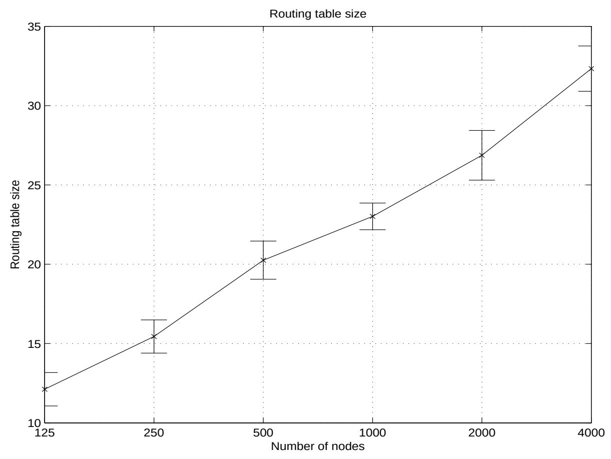
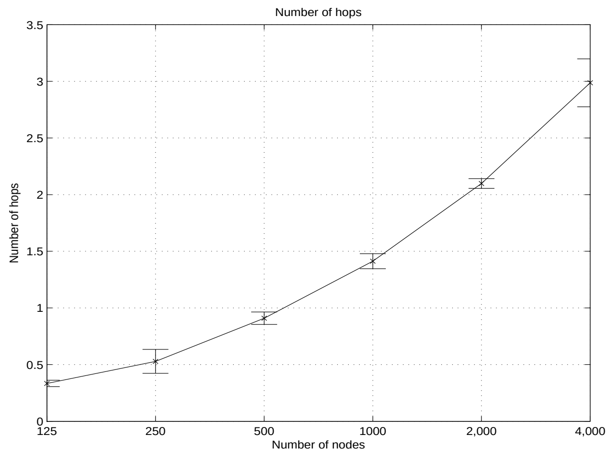


Fig. 4. False positives ratio as a function of the subscriptions law.



(a) Routing table size in dynamic settings.



(b) Average delivery depth in dynamic settings.

Fig. 5. Evaluation of our system in a dynamic setting with uniformly distributed subscriptions and with a size of $ID_{semantic}$ of 1024 bits

number of hops is inversely proportional to the size of routing tables, we notice that the delivery depth of events is slightly smaller than the one observed in the static setting, yet the logarithmic growth is preserved, as Figure 5.(b) illustrates.

These results indicate the ability of our system to cope with system dynamics.

V. Analysis

In this section, we are interested in comparing our approach to the work presented in [5]. We present a theoretical approach to estimate the overhead generated by both schemes. We limit the comparison to [5] as we think it is the closest scheme to ours; both schemes rely on Bloom filters for ID assignment but differ in overlay building and event dissemination. As [5] and our system use both Bloom filters we can safely assume they generate the same amount of false positives that depend on the size of Bloom filter used and the event space.

The work described in [5] might involve un-interested nodes in forwarding events while nodes in our system send multiple messages to correctly join the overlay. We thus estimate the overhead generated by both approaches. We will use the following notation:

- d refers to the size of $ID_{semantic}$.
- n denotes the number of nodes in the overlay.
- λ_{join} and $\lambda_{publish}$ denote the join rate and the publication rate respectively.
- k denotes to the number of join messages sent at each round.

Proposition 2. *On average, our scheme generates $k \times \frac{d}{2} \times \ln n \times \lambda_{join}$ join messages.*

Proof: We assume that the overlay graph is a random graph. In this case the diameter of the graph will be $\mathcal{O}(\ln n)$ where n is the number of nodes. For a node N with m bits in her $ID_{semantic}$ digest set to 1 there will be m join messages generated in the worse case. If the interests are distributed uniformly at random we can safely assume that the probability for a given bit in the $ID_{semantic}$ digest to be set to 1 is 1/2.

Therefore, on average we will have $k \times \frac{d}{2} \times \ln n \times \lambda_{join}$.

■

Proposition 3. *The overhead due to event dissemination in [5] amounts to:*

$$p \times \ln n \times \lambda_{pub}$$

where p is the probability that node N gets an event that she is not interested in.

Proof: A node N will forward an event e that is not interested in if she is in the path of this event. This happens if her ID cover matches the event. This could occur if one of the bits that are set to 0 in her ID digest are set to * in the ID cover.

Let N be a node with m bits set to 1 and p_m the probability that node N gets an event that she is not interested in, given m bits of N's ID are set to 1. Then:

$$p_m = 1 - (1 - \frac{1}{d})^{d-m} \quad (3)$$

where $\frac{1}{d}$ is the probability that one of the d bits is set to *.

Furthermore, let q_m be the probability that m bits of N's ID digest are set to 1. Then:

$$q_m = C_d^m \frac{1}{d}^m (1 - \frac{1}{d})^{d-m} \quad (4)$$

$$event_A = \text{N gets an event that it is not interested in}$$

$$event_B = \text{m bits of N's ID are set to 1}$$

$$Pr(event_A \cap event_B) = p_m \times q_m \quad (5)$$

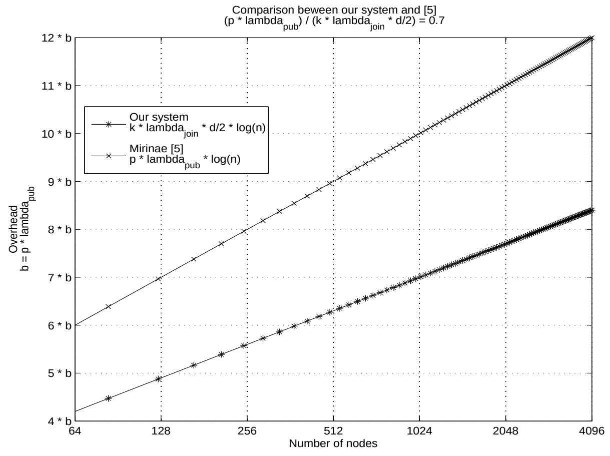
Let p denote the probability that node N gets an event that she is not interested in. Then

$$p = \sum_{m=0}^{d} p_m \times q_m \quad (6)$$

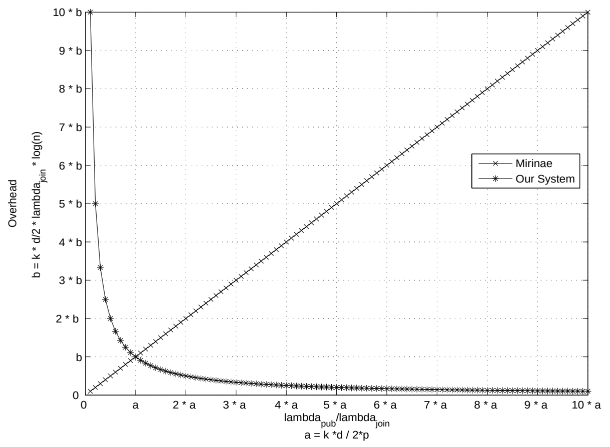
As described in [5], the event dissemination happens in $\ln n$ steps on average therefore: $p \times \ln n \times \lambda_{pub}$ approximates the overhead generated by event dissemination. ■

We now illustrate the impact of system parameters on the overhead generated by our system and the one described in [5]. There two important parameters we focus on are are the join rate λ_{join} and the publication rate λ_{pub}. If we assume that λ_{pub} is larger than λ_{join}, our scheme performs better. Such an assumption holds in practice if we rely on a system model where nodes remain on-line for reasonably long periods of time [3].

Figure 6.(a) shows the overhead as a function of the number of nodes: when $k \times \frac{d}{2} \times \lambda_{join} < p \times \lambda_{pub}$, our scheme generates less overhead than [5]. Moreover, Figure 6.(b) shows the overhead as function of $\frac{\lambda_{pub}}{\lambda_{join}}$: we observe that [5] performs better up to a threshold which corresponds to $\frac{\lambda_{pub}}{\lambda_{join}} = \frac{k \times d}{2 \times p}$ after which our scheme performs better.



(a) Overhead as a function of number of nodes



(b) Overhead as a function of $\frac{\lambda_{pub}}{\lambda_{join}}$

Fig. 6. Comparison between our system and [5]

VI. Conclusion

In this paper, we presented a new peer-to-peer approach for publish/subscribe systems. Our scheme can build a semantic overlay based on nodes' interests and disseminate events using a new interest proximity metric. The novelty of our approach is that it ensures only interested node to be involved in event dissemination while the overhead is low as we do not use any gossiping protocol. Furthermore, simulation results indicate that our scheme is resilient to system dynamics.

Although the simulations we conducted use arbitrary subscription distributions and the parameter space we explored is narrow, we were able to show that our system has a very low false negative rate.

We are currently implementing our system and building a testbed to verify if the properties we observed in this work would carry over in a realistic system deployment.

References

[1] A. Carzaniga, *Architectures for an Event Notification Service Scalable to Wide-area Networks.* PhD thesis, Politecnico di Milano, December 1998.

[2] A.Carzaniga, D. Eosenblum, and A. L. Wolf, “Design and evaluation of a wide-area event notification service,” *ACM transactions on computer systems*, vol. 19, no. 3, pp. 332–383, 2001.

[3] R.Chand and P. Felber, “Semantic peer-to-peer overlays for publish/subscribe networks,” vol. 3648, pp. 1194–1204, 2005.

[4] A. Gupta, O. D.Shahin, D. Agrawal, and A. E. Abbadi, “Meghdoot: Content-based publish/subscribe over p2p networks,” in *Proceedings of the 5th ACM/IFIP/USENIX International Conference on Middleware*, pp. 254–273, Springer-Verlag New York, Inc, 2004.

[5] Y. Choi, H. Lee, K. Park, and D. Park, “A new peer-to-peer overlay network for content-based publish/subscribe systems,” in *Global Telecommunications Conference, IEEE GLOBECOM*, 2005.

[6] Y. Choi, K. Park, and D. Park, “A peer-to-peer overlay architecture for large-scale content-based publish/subscribe systems,” in *Third International Workshop on Distributed Event-Based Systems*, 2004.

[7] S. Voulgaris, E. Riviere, A.-M. Kermarrec, and M. van Steen, “Sub-2-sub: Self-organizing content-based publish subscribe for dynamic large scale collaborative networks,” 2006.

[8] F. Cao and J. P. Singh, “Efficient event routing in content-based publish/subscribe service networks,” in *Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 2, pp. 929–940, 2004.

[9] W. W.Terpstra, S. Behnel, L. Fiege, A. Zeidler, and A. P.Buchmann, “A peer-to-peer approach to content-based publish/subscribe,” in *Proceedings of the 2nd International Workshop on Distributed Event-based Systems*, pp. 1–8, 2003.

[10] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, “Chord: a scalable peer-to-peer lookup protocol for internet applications,” *IEEE/ACM Transactions on Networking*, vol. 11, no. 1, pp. 17–32, 2003.

[11] S.Ratnasamy, P.Francis, R.Karp, and S.Shenker, “A scalable content-addressable network,” in *Proceedings of the 3rd International Workshop of NGC*, vol. 2233, pp. 14–29, LNCS, Springer, 2003.