

# On Event Routing in Content-based Publish/Subscribe through Dynamic Networks\*

Antonino Virgillito, Roberto Beraldi and Roberto Baldoni  
Dipartimento di Informatica e Sistemistica  
Università di Roma “La Sapienza”  
Via Salaria 113, 00198, Roma, Italy  
email: {virgi,beraldi,baldoni}@dis.uniroma1.it

## Abstract

*Content-based publish/subscribe communication systems are a popular technology for many-to-many information diffusion over large scale networks. Scalable solutions are obtained considering a network of distributed event brokers, dispatching information (events) from producers (publishers) to consumers (subscribers). Many scalable and efficient solution for routing events in content-based systems exist, that selectively send events only toward the interested subscribers. However, if subscribers having similar interest are present in different parts of the network, the benefits of such routing strategies significantly decrease. In this paper we propose a novel approach for enhancing content-based routing, based on the dynamic reconfiguration of the broker network. The reconfiguration aims at aiding the routing process by placing close to each others brokers that manage subscribers with similar interests. Metrics for measuring similarity of interests are discussed and a reconfiguration algorithm is presented.*

## 1 Introduction

Publish/subscribe (pub/sub) has emerged as a widespread and powerful communication paradigm for message-based distributed infrastructure. It is based on the concept of event service, a piece of software that decouples producers and consumers of events. In particular, consumers *subscribe* to the event service by submitting the features of the events to be notified for, while producers *publish* such events to the event service which will dispatch them to all intended receivers.

Early implementations of the pub/sub paradigm followed the *topic-based* scheme ([5, 7, 11]), where events are grouped in topics (or subject) i.e., a subscriber declare its interest for a particular topic and will receive all events related to that topic. Recently, a more general approach has emerged in the literature, *content-based* publish/subscribe ([2, 1, 3, 6]), where subscriptions are specified through constraints on properties of each event. In other words, each subscription is a predicate composed by a set of constraints on the values of attributes of the event.

Even though content-based pub/sub is more flexible than its topic-based counterpart, it is notably more difficult to implement: for example the set of intended receivers for an event cannot be determined a priori but must be computed on a per-event basis. This makes content-based pub/sub not immediately scalable to a large system size, where a high number of subscribers, publishers and event publications must be supported. To get scalability, distributed implementations of the event service have to be considered: these are based on a set of event brokers that exchange messages over a topology formed by TCP-like connections. In this case, events and subscriptions are submitted by clients to one of the event brokers and the delivery of the events to the intended subscribers passes through two basic functions to be executed at each broker: (*i*) calculating for each event  $e$  the set of receivers and (*ii*) propagating  $e$  to all such receivers. Both functions are based on an *event routing*. To improve the performance of such matching and diffusion mechanisms, a key metric is to reduce as many as possible *the number of TCP hops experienced by an event  $e$  to be delivered to all its intended subscribers*. Content-based routing of Siena for example avoids an event to be diffused in parts of the network where there are not subscribers for that event. This obviously improves the aforementioned metric. However Siena works on a static network topology connecting the brokers (i.e., the network topology graph is given a priori and do not change over the time): this poses an im-

---

\*This work has been partially supported by a grant from Projects “MAIS” (funded by Italian MIUR), “EU-PUBLI.COM” (#IST-2001-35217) and “MIDAS”(#IST-2001-37610) (funded by EU IST).

explicit limit to this improvement due to the fact that two brokers with similar subscriptions could be very far each other in terms of TCP hops, where *similar* means that most of the events matching subscriptions managed by one broker, they also match subscriptions managed by the other.

In this paper we propose a dynamic and acyclic network of brokers which reconfigures itself by removing and adding TCP connections following a *similarity* principle over the set of current subscriptions. More specifically, two brokers which manage similar subscriptions should be placed very close (in the best case they should be neighbors) in the network topology graph. In this way when an event  $e$  matching subscriptions at both brokers will be received by one of them it will be delivered also to the other broker in as few as possible TCP hops.

This reconfiguration algorithm has naturally to work in synergy with an underlying content-based routing protocol as similarity among subscriptions can be found by locally exploiting content-based routing information each time a subscription change occurs at a broker. The algorithm rearranges the network topology on a local base and endows two main mechanisms:

- A similarity metric (*associativity*) local at each broker, which gives an heuristic to count the degree of similarity of the subscriptions managed by a broker with the ones of its neighbors. This function is computed each time there is a change at a broker in its local subscriptions.
- A protocol, initiated at a broker  $B$ , when there is a suspect that the associativity function could be increased by connecting directly  $B$  to another broker  $B'$  with high degree of similarity. This protocol first finds  $B'$  (if any). In the affirmative, it connects  $B$  to  $B'$  and removes the connection between  $B$  and its neighbor which is on the path to get to  $B'$ .

The paper is structured as follows. Section 2 presents the related work, Section 3 introduces the system model. Section 4 describes first the rationale behind the proposed algorithm and then the reconfiguration algorithm itself. Section 5 discusses some improvements over the reconfiguration algorithm and Section 6 finally concludes the paper.

## 2 Related Work

The contribution of this paper relates to two research topics: content-based publish/subscribe systems and self-configurable application-level networks. Content-based routing algorithms have been widely studied in the literature. Systems such as Siena [2], Gryphon [1], JEDI [3] and, more recently, Hermes [6] implement efficient and scalable routing algorithms that reduce network traffic by selectively

dispatching events only to actual subscribers. All these systems are built through an application level network of event brokers but Hermes is the only one providing reconfiguration mechanisms. Specifically, Hermes reconfiguration provide repair mechanisms in case of brokers' faults.

Reconfiguration in application-level networks has been widely studied in the context of structured peer-to-peer overlay network infrastructures, such as Chord [9], Pastry [8] and Tapestry [12]. Such systems offer high-performance point-to-point routing for large scale networks. Their self-reconfiguration capability is exploited for fault-tolerant routing and for adjusting routing paths with respect to metrics of the underlying network.

Overlay network infrastructures represent a general connectivity framework for many classes of peer-to-peer applications. For example, Scribe [7] and Bayeux [11] are two topic-based publish/subscribe systems built on top of two overlay network infrastructures (respectively Pastry and Tapestry). They do not deal with reconfiguration as it is completely managed by the overlay network level.

With respect to the aforementioned systems, our approach to reconfiguration differentiate in two senses: i) it is specific for the content-based routing problem and ii) it operates *on the same level* as content-based routing. That is, rather than basing reconfiguration on the underlying network, we start from an existing routing algorithm (Siena's) and add a dynamic behavior that rearranges the topology considering the distribution of interest on the subscribers.

## 3 Publish/Subscribe Model

This section defines the key concepts related to the model of content-based publish/subscribe event service we refer to in the rest of the paper, including a presentation of the content-based routing algorithm on which our reconfiguration algorithm is based.

### 3.1 Events and Subscriptions

Subscriptions and events are defined over a predetermined event schema, constituted by a set of  $n$  *attributes*, each defined through a name and a type, where the type can be a common basic type such as integer, real or string. The event schema can be graphically represented as an  $n$ -dimensional *event space*. In the following of the paper we consider a structured and fixed event space, that is all events and subscriptions adhere to a same structure defined by the event space itself. We follow this approach for ease of presentation, though the majority of the actual systems use an unstructured event space. However, all the concepts here can be simply generalized to the unstructured case.

An event is a set of  $n$  values for each attribute, whose type is consistent with that attribute's type. If all values

are defined for every attribute, an event can be considered as a point in the event space. Therefore in the sequel for simplicity we abstract the notion of subscription as a subset of events of the event space defined by using a set of *constraints* over numerical attributes. We say that an event *matches* a subscription if the point it represents falls inside the zone identified by the subscription. As far as figures are concerned, for ease of presentation we only consider a 2-dimensional event space.

### 3.2 Publish/Subscribe Event Service

A distributed event notification service is composed by a set of processes, namely *event brokers*,  $\{B_1, \dots, B_n\}$  interconnected through transport-level links. In particular maintains a set of open connections (*links*) with other brokers (its neighbors). The link connecting brokers  $B_i$  and  $B_j$  is indicated with  $l_{i,j}$  (or  $l_{j,i}$ ) and the set of neighbors of  $B_i$  is indicated with  $\mathcal{N}_i$ . A broker can be contacted by clients, that depending on their role, can be divided in *subscribers* or *publishers*. Publishers produce information by firing events to one of the brokers, while subscribers express their interests for specific events by issuing subscriptions to one of the brokers. Each broker stores locally these subscriptions and communicate with other brokers in order to propagate events fired by its publishers. We define as the *zone of interest* of a broker  $B$ , denoted  $Z_B$ , the union of all local subscriptions at  $B$ .

Each time an event  $e$  is received by a broker  $B$  either from its local publishers or from a link, it matches  $e$  against all local subscriptions and then forwards  $e$  through the links which can lead to potential  $e$ 's subscribers (this corresponds to a routing operation). In this case we say that  $B$  acts as a forwarder for  $e$ .

Finally, the following basic assumptions state:

- brokers are initially arranged in an acyclic topology graph where an edge is a TCP-like connection;
- there exists an underlying content-based routing protocol whose specification are given in the next section.

### 3.3 Content-based routing protocol

We consider the content-based routing algorithm for acyclic peer-to-peer topologies introduced in Siena [2]: subscriptions are diffused in order to build paths for routing events; subscription diffusion is limited exploiting coverage relationships among subscriptions. The key aspect of Siena is to dispatch events to all the intended subscribers by excluding parts of the network with no interested subscribers. This allow to largely reduce event traffic with respect to naively flooding each event over the entire network [4].

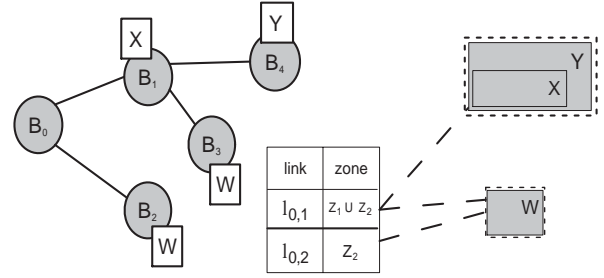


Figure 1. Routing Table Example

Routing tables have to be kept updated whenever a subscription change occurs<sup>1</sup>. When a new subscription  $X$  arrives at a broker  $B_i$ , if  $X$  is not a subset of  $Z_{B_i}$ , it propagates  $X$  to the other brokers and  $X$  is added to  $Z_{B_i}$ . When a broker  $B_j$  becomes aware of  $X$  through a message received from one of its incoming TCP links, namely  $l$ , it updates its routing table by adding  $X$  to the  $l$ 's entry in the routing table.

Figure 1 shows an example of routing table for broker  $B_0$  where outgoing links are denoted  $l_{0,1}$  and  $l_{0,2}$ . The second column of each entry  $l_{0,i}$  represents the union of all the zone of interests of brokers which are connected to  $B_0$  through  $l_{0,i}$ . We denote such union as the zone covered by  $l_{0,i}$ .

## 4 The Reconfiguration Algorithm

We call *reconfiguration* the process through which a broker alters the topology of the network in order to improve the performance of the routing algorithm. This section firsts describes the idea underlying reconfiguration based on subscription similarity and then presents a reconfiguration algorithm.

### 4.1 Basic Idea

Let us consider a random acyclic topology characterized by an average distance  $h$  between two brokers. Let  $B$  be a broker of such a network and  $X$  the subscription it holds. If all brokers can generate an event of interest for  $B$  with the same probability, then the value  $h$  also provides the average cost, expressed in terms of TCP hops, required to deliver the event to  $B$ . Let now  $B'$  be another broker that holds the same subscription  $X$ . The minimum average cost of delivery the events both to  $B$  and  $B'$  is obtained when  $B$  and  $B'$  are directly connected. The simulation results reported in [4] provides a quantitative analysis of the improvements obtained when brokers with similar interests are kept close each other. In particular, it is evident from simulations that

<sup>1</sup>A complete description of Siena is out of the aim of this paper, the interested reader can refer to [2] for a complete description of the algorithm.

when similar subscriptions are scattered over the network, the routing algorithm falls into its worst case, offering little advantages over a simple flooding of events.

The rationale underlying our reconfiguration approach is the following: suppose that the current network of brokers is represented by the graph  $G$ , and let  $B$  be the only broker with subscription  $X$ . When a broker  $B'$  in the network receives the same subscription  $X$  (or one that is “similar” to  $X$ ), the topology of the network is rearranged in a such a way that, if  $G'$  is the new graph then: (i)  $B$  and  $B'$  are neighbors in  $G'$ , (ii) the graph  $G'$  remains acyclic; (iii)  $G$  and  $G'$  have the same average distance between two brokers (i.e. the graphs  $G'$  and  $G$  have the same “randomness”).

## 4.2 Associativity

One main point of our approach to reconfiguration is to define how subscription similarity can be measured. Being  $B_1$  and  $B_2$  two brokers hosting subscriptions that partly overlap, there is some probability that some events in which  $B_1$  is interested also interest  $B_2$  or viceversa. Following this simple interpretation, it is straightforward to measure the communality of interest between two brokers  $X$  and  $Y$  as the area of the intersection of their zones of interest  $X$  and  $Y$ . Then, we can consider as the first objective of reconfiguration the reduction of the distance between brokers having intersecting zones of interest. The metric that a broker will locally try to enhance in order to obtain this behavior is the *associativity*. Given two zones<sup>2</sup>  $X$  and  $Y$ , we define the associativity between  $X$  and  $Y$  as following:

$$AS(X, Y) = |X \cap Y|$$

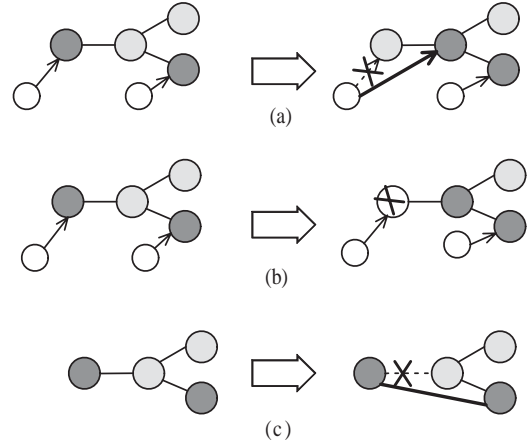
Being  $B_1$  and  $B_2$  two brokers, with zones of interest  $Z_{B_1}$  and  $Z_{B_2}$  respectively, the associativity between  $B_1$  and  $B_2$  is defined as  $AS(B_1, B_2) = AS(Z_{B_1}, Z_{B_2})$ . If  $Z_{B_1} \cap Z_{B_2} = \emptyset$  then  $AS(B_1, B_2) = 0$ . In the following the notation  $a_{i,j}$  will also be used to denote the associativity between brokers  $B_i$  and  $B_j$ . Subsequently, an overall value of the associativity for a broker  $B_i$  can be simply defined as the sum of the associativity values with  $B_i$  and each of its neighbors:

$$AS(B_i) = \sum_{B_k \in \mathcal{N}_i} AS(B_i, B_k)$$

Rearranging the network in order to obtain an increment of the associativity at every single broker, means increasing the probability that two brokers with similar interest get close to each other. This enhances the overall degree of locality in the network, approximating a situation where *all* brokers with same interest are in a same part of the network, that for the discussion in Section 4.1 enhances the

<sup>2</sup>Here with “zone” we refer indifferently to a zone of interest of a broker or a zone covered by a link.

selectivity of the routing algorithm. However, other issues arise when trying to improve the efficiency of the routing process. We will discuss them in Section 5.2.

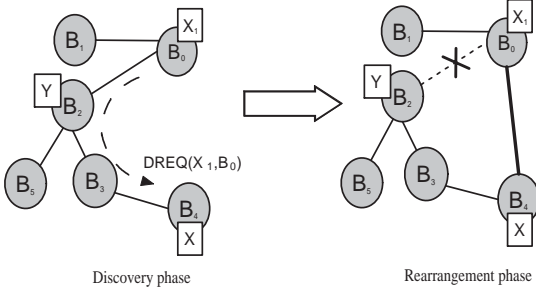


**Figure 2. Possible Approaches to Reconfiguration**

## 4.3 Design Constraints

Different solutions may be considered for placing broker with similar interest close to each other:

- *Moving clients.* A subscriber client may be forced to connect to a broker that already hosts similar subscriptions (Figure 2(a)). This solution can lead to badly distributing load among brokers, because brokers with more subscriptions will also have to maintain a higher number of client connections. Moreover, the number of connections that a client has to maintain can unpredictably grow when it has subscriptions spanning different interests.
- *Moving subscriptions.* Clients remain connected to the broker they contacted first, but their subscription are moved to a different broker (Figure 2(b)). Similar subscriptions can be assigned to neighbor brokers. The idea of moving subscriptions from one broker to another is exploited by topic-based systems such as Bayeux [11], Scribe [7]: such system assigns subscription for a same topic to be hosted on a same broker. Though this is surely an interesting approach, many problems arise when applying it to a dynamic content-based system: defining a criteria for grouping content-based subscriptions and assigning uniquely a group of to a broker is a difficult task, especially when assuming that the number of brokers may change over time. Wang et al. in [10] apply this idea to content-based



**Figure 3. Reconfiguration Example**

systems but only considering subscriptions with equality operators and a fixed number of broker.

- *Moving brokers.* A broker creates a connection to one that has a similar zone of interest (Figure 2(c)). This approach is the one that more easily adapts to the content-based routing algorithm presented in Section 3.3. It can be realized acting only on the connection among brokers without having to involve clients or move subscriptions.

The algorithm presented in this paper exploits the latter solution. The idea of the algorithm is very simple: when a broker changes its zone of interest, it identifies a new neighbor with a high similarity and tries to connect to it. However, the content-based routing mechanism imposes some constraints in the actual realization:

- A broker does not have the exact knowledge of all the subscriptions stored by other brokers in the system. Through its routing table it only keeps an approximated view of the subscription distribution.
- The topology must be maintained acyclic also after reconfiguration occurs. This is a basic assumption for the routing algorithm to work properly and efficiently.

In the remainder of this section we introduce a reconfiguration algorithm that accounts the constraints above.

#### 4.4 Algorithm Description

The reconfiguration algorithm can be triggered when the overall associativity at a broker  $B$  changes due to a modification in its zone of interest, after a subscribe or an unsubscribe request from a client. When the zone of interest grows, it can be an opportunity for  $B$  to increase its associativity:  $B$  has to find a broker  $B'$  in the network such that by connecting with it, a contribution to associativity is gained with respect to the new zone of interest.

The reconfiguration process is structured in two phases, namely *discovery* and *rearrangement*. Each of the phases corresponds to one of the constraints imposed by the content-based routing (Figure 3):

- Since  $B$  does not know all the interests of the brokers, a *discovery* phase is executed to find the possible new neighbors. The discovery phase is based on a query-reply cycle:  $B$  sends a series of Discovery Request messages (*DREQ*) on the network routed through all the possible neighbors. Each broker receiving the message can either reply or forward it further to its neighbors. Eventually, a reply is generated for each request and sent to  $B_i$ . The reply message can be either a positive (*PREP*) or negative (*NREP*) response. A positive response contains the reference of the brokers to which  $B_i$  has to connect.
- When  $B$  opens a new connection, this creates a cycle in the network. Then, when  $B$  adds a new neighbor it also has to *remove* one of its old ones, in a way such that the no cycles appear. In the *rearrangement* phase, a new link is created from  $B_i$ , in correspondence to all the positive reply messages. When a new connection with a broker  $B'$  is created, network topology is maintained acyclic, simply by deleting the link from which the request that reached  $B'$  has been sent. Finally, routing tables are updated.

Figure 3 depicts an example of reconfiguration triggered at broker  $B_0$  when its zone of interest switches to  $X_1$ , with  $X_1 \cap X \neq \emptyset$ . During the discovery phase, the *DREQ* message reaches  $B_4$ . In the rearrangement phase the new link from  $B_0$  to  $B_4$  is created, and the previous neighbor  $B_2$  is disconnected by  $B_0$ . Finally, we specify the behavior of a broker in case its zone of interest shrinks, after an unsubscribe: the reconfiguration process is simply triggered in one of the neighbors, reducing to the described case.

#### 4.5 Algorithm Details

This section analyzes in more details the reconfiguration algorithm, in the case a broker  $B_s$  augments its zone of interest due to a subscribe request from a client.

**Starting the Reconfiguration** When a client issues a subscribe request, the zone of interest at a broker  $B_s$  remains unchanged if the zone added by the subscriber is already covered by other subscriptions. Obviously in this case neither reconfiguration nor routing table update are triggered.

When  $B_s$ 's zone of interest increases from  $X_{old}$  to  $X$ ,  $B_s$  will try to substitute its neighbors in order to enhance its associativity. Intuitively, a reconfiguration starts when  $B$  can find a more convenient neighbor. Thus,  $B_s$  will remove

neighbor  $B_k$ , whose associativity with  $B_s$  is  $a_{s,k}$ , if it exists a broker  $B_x$ , reachable from link  $l_{s,k}$ , whose associativity with  $B_s$  is higher than  $a_{s,k}$ . The presence of such a broker can be inferred by  $B_s$  only from the routing table. That is, if  $B_x$  exists, its zone of interest will be included in the zone covered by  $B_s$ 's links.

That is, a discovery request message,  $DREQ(B_s, X, t_k)$ , is sent through any link  $l_{s,k}$  covering a zone  $Z_k$  such that  $AS(X, Z_k) > a_{s,k}$ . The message  $DREQ$  contains the new zone  $X$ , the reference of the source broker  $B_s$  and the *associativity threshold*  $t_k = AS(X, Z_k) - a_{s,k}$ .

**Discovery Phase** The discovery phase consists in forwarding the request message through the brokers' network to greedily reach the first broker whose associativity with  $B_s$  exceeds the associativity threshold.

When a broker  $B_f$  receives a  $DREQ(B_s, X, t)$  from a neighbor  $B_j$ , it can propose itself as the new neighbor or forward the request again. To this end,  $B_f$  calculates the associativity  $a = AS(X, Z_{B_f})$  between the zone  $X$  and its current zone of interest  $Z_{B_f}$ . If  $a > t$  then  $B_f$  sends the positive reply message  $PREP(B_f)$  to the broker  $B_s$  and does not propagate the request any further. The  $PREP$  message contains the identification of the candidate brokers  $B_f$ .

Otherwise, the request is forwarded in order to let  $B_s$  get as closer as possible to a source of higher associativity.  $B_f$  checks whenever some of its neighbor links (except the one connecting  $B_j$ ) cover a zone whose associativity with  $X$  is higher than  $t$ . If no links exist that satisfy such a condition, then  $B_f$  sends a negative reply message  $NREP$  to  $B_s$ . Otherwise, the  $DREQ$  message is sent to the link that covers the zone with higher associativity with  $X$ .

**Rearrangement Phase** After a broker  $B_s$  has issued the discovery requests, it waits for all the replies.  $B_s$  can start the reconfiguration as soon as a reply is received, as at most one request is issued for each link, and all corresponding replies can be treated separately. If a  $NREP$  message is received, nothing happens. If a  $PREP(B_r)$  is received by broker  $B_s$ ,  $B_s$  creates a new link with the broker  $B_r$  specified in the message.

Each  $PREP$  is associated to the link  $l_{s,k}$  from which the corresponding request started: such a link is dropped, in order to maintain an acyclic topology. Deleting  $l_{s,k}$  keeps the network completely connected, provided no reconfiguration request has been issued by other brokers in the meantime. A request flowing on a same path travelled by another request currently in progress has then to be blocked. Each time the network configuration changes, due to the creation or the deletion of a link, routing tables have to be updated on all the brokers on the path from  $B_s$  to  $B_r$ .

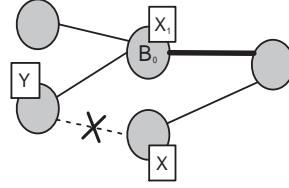


Figure 4. Alternative Mechanism for Link Removal

## 5 Improvements on the Basic Algorithm

This section discusses some inefficiency in the definition of the associativity metrics and the reconfiguration algorithm, as presented in the previous section. Possible solutions to overcome such problems are presented.

### 5.1 Link Removal

The algorithm described above simply removes the link on which a request is sent. This can obviously cause a broker to disconnect from a neighbor, also if its associativity is greater than 0. Revisiting the example in Figure 3, if broker  $B_0$  had non-zero associativity with broker  $B_2$  ( $X_1 \cap Y \neq \emptyset$ ), this would be lost after reconfiguration. Though the overall associativity increases, with a simple modification to the algorithm the reconfiguration process can be made much more effective, avoiding associativity losses with removed neighbors. It is sufficient to consider a different link removal criteria in the rearrangement phase: when connecting to a new neighbor, the link to be deleted is chosen not necessarily from the source broker but on the whole path travelled by the  $DREQ$  message that connects the source with the new neighbor. This link is chosen as one that connects two brokers with no associativity. Performing a reconfiguration with this new criterion, the previous example leads to the topology depicted in Figure 4. This modification to the basic algorithm tend to preserve neighbors that contribute to the overall associativity and discard neighbors that do not, obviously enhancing the associativity at all the brokers involved in the process.

### 5.2 Refining Associativity Definition

Considering only the intersection of the zones of interest as a metric for subscription similarity is not sufficient for obtaining an efficient topology: for example, brokers with small zones of interest will be excluded from the reconfiguration as they cannot provide high values for intersection. Also normalizing the intersection against the smallest zone creates undesired behavior: small zones will be easily con-

tained in biggest one and brokers which host them subsequently obtain high value for the metric. The result is that broker with small zones will easily connect to other ones and will more probably receive more events they do not need.

Contrarily, brokers with smaller zones of interest have to receive *less* events: being  $B_i$  and  $B_j$  two brokers with zones of interest  $Z_{B_i}$  and  $Z_{B_j}$ , respectively, if  $Z_{B_i} \in Z_{B_j}$  it is convenient for  $B_i$  to be connected *only* with  $B_j$  from which it will receive all and only the events it needs. In general, we can say that the probability for a broker of being a forwarder for others' events increases as its zone of interest is smaller with respect to the number of its neighbors. This observation leads to relating associativity both to the relative dimension of the zones of interest and to the number of neighbors. First, we give an alternative definition of associativity between two zones:

$$AS'_X(Y) = \frac{|X \cap Y|}{|X|}$$

This definition can be obviously extended to two brokers. If  $X \cap Z_{B_i} = \emptyset$  then  $AS'_B(B_i) = 0$ . Also, if  $X \subset Z_i$  then  $AS'_B(B_i) = 1$ . In the other cases, the value is in the range (0..1). Then, an overall value of associativity for a broker  $B_i$  can be defined as:

$$AS'(B_i) = \frac{AS(Z_{B_i}, \bigcup_{B_k \in \mathcal{N}_i} Z_{B_k})}{|\mathcal{N}_i|}$$

This value represents an average of the associativity with each neighbor with respect to the overall number of neighbors. Each zone is counted only once. The lower the value of  $AS(B_i)$ , the higher the probability for  $B_i$  to be connected with neighbors that will forward it events it is not interested in. It is then convenient for each broker only to maintain links that contribute to the overall associativity (provided the overall connectivity of the network is preserved). Trying to maximize associativity for a broker will lead to topologies where *i*) brokers with intersecting zones of interest will tend to stay close to each other and *ii*) brokers with large zones of interest will have more neighbors and viceversa.

## 6 Conclusions and Future Work

In this paper the concept of reconfiguration in content-based publish/subscribe system has been introduced. Some metrics and policies for reconfiguration have been discussed together with a reconfiguration algorithm that groups similar subscribers in a same part of the network, augmenting the probability that best-case scenarios occurs for a selective content-based routing algorithm, such as the one used in Siena. We believe that augmenting a content-based event system with reconfiguration capabilities that accounts the distribution of subscriber is a promising idea. However, the

basic steps in this direction that we presented in this paper require further investigation, especially for what concern the evaluation of costs and benefits of reconfiguration. In particular, we have to determine if some circumstances exist in which the traffic generated by the control messages of the reconfiguration algorithm may outweigh its advantages. An extensive simulation analysis is currently in progress.

## References

- [1] 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 International Conference on Distributed Computing Systems*, 1999.
- [2] A. Carzaniga, D. Rosenblum, and A. Wolf. Design and Evaluation of a Wide-Area Notification Service. *ACM Transactions on Computer Systems*, 3(19):332–383, Aug 2001.
- [3] G. Cugola, E. D. Nitto, and A. Fuggetta. The JEDI Event-Based Infrastructure and its Applications to the Development of the OPSS WFMS. *IEEE Transactions on Software Engineering*, 27(9):827–850, September 1998.
- [4] G. Muhl. *Large-Scale Content-Based Publish/Subscribe Systems*. Phd thesis, Technical University of Darmstadt, 2002.
- [5] B. Oki, M. Pfluegel, A. Siegel, and D. Skeen. The Information Bus - An Architecture for Extensive Distributed Systems. In *Proceedings of the 1993 ACM Symposium on Operating Systems Principles*, December 1993.
- [6] P. Pietzuch and J. Bacon. Hermes: A Distributed Event-Based Middleware Architecture. In *Proceedings of the 1st International Workshop on Distributed Event-Based Systems (DEBS'02)*, 2002.
- [7] A. Rowston, A. Kermarrec, M. Castro, and P. Druschel. SCRIBE: The Design of a Large-Scale Notification Infrastructure. In *3rd International Workshop on Networked Group Communication (NGC2001)*, 2001.
- [8] A. Rowston and P. Druschel. Pastry: Scalable, Decentralized Object Location and Routing for Large-Scale Peer-to-Peer Systems. In *Proceedings of International Conference on Distributed Systems Platforms (Middleware)*, 2001.
- [9] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In *Proceedings of ACM SIGCOMM*, 2001.
- [10] Y. Wang, L. Qiu, D. Achlioptas, G. Das, P. Larson, and H. J. Wang. Subscription Partitioning and Routing in Content-based Publish/Subscribe Networks. In *16th International Symposium on Distributed Computing (DISC'02)*, October 2002.
- [11] S. Q. Zhuang, B. Y. Zhao, A. D. Joseph, R. Katz, and J. Kubiatowicz. Bayeux: An Architecture for Scalable and Fault-tolerant Wide-area Data Dissemination. In *11th Int. Workshop on Network and Operating Systems Support for Digital Audio and Video*, 2001.
- [12] S. Q. Zhuang, B. Y. Zhao, A. D. Joseph, R. Katz, and J. Kubiatowicz. Tapestry: An Infrastructure for Fault-Tolerant Wide-Area Location and Routing. Technical Report UCB/CSD-01-1141, University of California at Berkeley, Computer Science Division, April 2001.