A Genetic Algorithm for Multicast Mapping in Publish-Subscribe Systems*

Mário Guimarães
Universidade de Lisboa
mario.guimaraes@netc.pt

Luís Rodrigues†
Universidade de Lisboa
ler@di.fc.ul.pt

27th February 2003

Abstract

In publish-subscribe systems, multicast is an efficient way to propagate information from the publishers to a group of subscribers. This paper studies the problem of mapping a large set of subscriptions into a fixed, smaller, set of multicast groups in order to support efficiently the dissemination of events. Given the large search space for this problem, it is infeasible to obtain the optimal solution in reasonable time. To address this difficulty, the paper proposes and evaluates a genetic search solution for the mapping problem.

1 Introduction

The publish-subscribe communication paradigm has been recognized as an useful alternative to the client-server model for building flexible and extensible distributed information systems [1, 2]. This paradigm supports distributed and anonymous communication among several processes. Participants have no explicit knowledge of each other and are only required to agree on the format of the data being exchanged. Participants can be classified in two categories: *publishers*, which produce data as a sequence of notifications, and *subscribers*, which inform about the characteristics of the data they are interested in receiving. It is the responsibility of the publish-subscribe middleware [3, 4, 5, 6] to ensure that the published information is delivered to all interested subscribers.

When many subscribers are interested in the same information, the use of a multicast group to disseminate the data is much more efficient than the use of multiple point-to-point connections. However, there is also cost associated to the maintenance of a multicast group. Firstly, the number of available multicast addresses may be limited. Secondly, each multicast group consumes resources at network routers. Therefore, it is likely that only a limited set of multicast groups is available to the publish-subscribe middleware. In this case, one of the

*Selected section of this report will be published in the proceedings of the 2nd IEEE International Symposium on Network Computing and Applications April 16-18, 2003, Cambridge, MA, USA.

†This work has been partially supported by the FCT project INDIQoS: (POSI/ 41473/ CHS/ 2001).

roles of the publish-subscribe middleware is to map the existing subscriptions into the available multicast groups [7, 8]. We call this problem, the *multicast mapping* problem.

To optimize the use of network resources, a mapping algorithm may cluster in the same multicast group subscriptions which are similar but not exactly equal. This means that subscribers may receive some amount of undesired notifications that have to be filtered locally. The goal of a mapping algorithm is to reduce the amount of undesired information that is received and filtered at subscribers, as a result of the subscription clustering. Since this is an NP-complex problem with a huge search space, it is infeasible to obtain the optimal solution in reasonable time. To address this difficulty, the paper proposes and evaluates a genetic search solution for the mapping problem. The algorithm applies a cost function to each mapping solution, and returns the best one seen after applying a number of genetic transformations to a set of mapping candidates over a number of predefined search iterations.

The rest of the paper is organized as follows. Section 2 makes a brief introduction to publish-subscribe systems and to the several subscription models available. Section 3 introduces the mapping problem and the metrics used to assess the quality of the solution. Our genetic algorithm is presented in Section 4 and its evaluation presented in Section 5. Finally, section 6 concludes the paper.

2 Publish-Subscribe Systems

The publish-subscribe interaction paradigm is an anonymous and general style of one-to-many interaction between producers and consumers of events or messages. One of the most important features of the paradigm is the strong decoupling between publishers and subscribers. Publishers send information as packet notifications and subscribers provide information about the data they are interested in receiving. The exchange of information between the publisher and the subscribers is performed by an underline middleware layer, usually named a *message broker*. Typically, the message broker is a distributed entity, composed of proxies that are executed at the publisher and subscriber nodes plus one or more cooperating servers. Publish-subscribe systems differ on the supported subscription models and on the underlying broker architecture.

2.1 Subscription Models

In publish-subscribe communication, a subscriber references data from the information space according to some addressing scheme, usually based on an hierarchical name-space. In *channel-based* systems [9], subscriptions simply use the name of a communication channel where the information is disseminated (for instance, "/stocks/mycompany"), which behaves like a pipe of message notifications. The addressing scheme is more flexible in *subject-based* systems [3, 4, 10, 11], where the support of wildcard characters in the reference to subject names, permits the mapping to multiple subjects at once (i.e. "/sports/football/teams/*" maps to all football teams). More recently, subject-based systems were extended with typed information spaces, introducing the support to *content-based* selection of notifications [5, 6, 12, 13, 14, 15].

In content-based publish-subscribe, information space has multiple dimensions corresponding to each notification attribute. When a notification is published, the system inspects its attributes against each known subscription, and only if there is a match, the notification is delivered to the corresponding subscribers. For example, a subscription like "(title=[/stocks/mycompany]; value > $5)" will match a stock notification of "mycompany" when its value is greater than $5. In this work, we tackle content-based systems.

2.2 Broker Architectures

Basically, there are two main distinct broker architecture solutions when implementing a publish-subscribe network: *server-based* architectures and *serverless* architectures.

The first approach consists of using one or more server nodes organized according to some overlay topology. Each server can have any combination of publishers and subscribers directly connected [5, 6, 8, 10, 13]. All messages flow through the overlay network of servers according to some routing mechanism, and publisher and subscriber nodes do not exchange data directly. Typically, routing of messages in the overlay network takes into account the existing subscriptions, such that data is only disseminated to the servers that have interested subscribers connected. The knowledge required to build the routing tables may be disseminated by broadcasting the subscriptions in the overlay network. Some systems also support the dissemination of publisher advertisements to avoid the broadcast of subscriptions, under the assumption that subscriptions are more frequent than changes in the advertisements [6]. Finally, servers can inspect notification attributes, and therefore, support content-based routing.

In the second model, dissemination of messages is not based on the availability of specialized servers [3]. Instead, the message broker is made exclusively of proxies that run on the publisher and subscriber nodes. These proxies exchange notifications using low level network broadcast or multicast. In the multicast case, this approach requires the use of some algorithm to map subjects into group addresses. Because of the lack of a network of servers capable to filter and route notifications efficiently, systems like these generally support subject-based addressing. However, content-based filtering can be done by the application.

2.3 On the Role of Multicast

The discussion of the advantages and disadvantages of each of the previous architectures is outside the scope of this paper (a proposal of an architecture that combines the positive aspects of both architectures can be found in [16]). However, it can be observed that the use of multicast groups to optimize the dissemination of information can be applied to both architectures.

On server-based architectures, multicast groups can be used between servers and their locally connected subscribers. In the serverless architectures, multicast groups can be used to disseminate information from publishers directly to subscribers. In both cases, multicast groups can save significant processing and network resources by preventing the same message from being sent several times to each of the subscribers (most likely, over the same physical link).

Subjects	Subscribers	Subjects	Subscribers
a	A B G	i	C D
b	A	t	E H I
c	B G	u	F J K
d	A	v	B G
e	D	x	F J K
g	C	z	E H I
h	C		

Table 1: Subscription set S.

The importance of the use of multicast has been recognized before in the literature. Levine et. al. [7] argue that content that is broadcast without regard to receiver interest is inefficient, and so, suggest the partitioning of receivers into groups based on interests in application content. They conclude that a mixture of group addressing and filtering are an advantage when multicast addresses are scarce and an efficient mapping algorithm is used. The solution in this paper can be seen as a way to make efficient use of multicast groups. T. Wong et al. [17], present a clustering algorithm to group data sources and receivers, but show only the applicability to subject-based.

In the following section, the problem of mapping subscriptions into multicast groups is addressed.

3 The Multicast Mapping Problem

This section addresses the mapping problem. First, we introduce the problem using some simple example. Then we introduce a number of concepts that are necessary to define a metric to assess the quality of a given mapping solution.

3.1 Overview

Although the use of multicast groups can save significant network resources, its use does not come for free. Firstly, in practical settings, the number of available multicast addresses may be limited. More important however, is that each multicast group requires state to be maintained at networks routers. Therefore, it is likely that any practical implementation of a message broker can only rely on a limited number of multicast groups to support the dissemination of notifications. The multicast mapping problem consists of finding an appropriate mapping between the existing subscriptions and the set of available multicast groups. This mapping can be denoted by the relation $R \subset \{(s, g) : s \in S \wedge g \in G\}$, where S is the set of all subscriptions, and G the set of all available multicast groups.

To illustrate the problem, we use an example based on subject-based subscriptions (in the remaining of the paper we address the more general case of content-based subscriptions). Table 1 depicts a set of thirteen different subjects and the identifiers of the subscribers for each of these subjects (for instance, subject a is subscribed by A, B, and G). Naturally, if a multicast group is used

g_1	a	A B G
g_2	b d	A
g_3	c v	B G
g_4	g h	C
g_5	e i	C D
g_6	u x	F J K
g_7	t z	E H I

(a)

g_1	a c v	B G
g_2	a b d	A
g_3	g h i	C
g_4	e i	D
g_5	u x	F J K
g_6	t z	E H I

(b)

Table 2: Mapping examples.

for each subject, thirteen multicast groups would be required to support the message dissemination.

However, it is possible to save resources by clustering similar subscriptions. For instance, since subjects t and z are only subscribed by nodes E, H, and I, the same multicast group can be used to disseminate the messages from these two subjects. The same reasoning can be easily applied to subjects u and x. However, some mappings are not so obvious: Subjects a, c, and v are subscribed by both B and G; However, a is also subscribed by node A which is not interested in c or v. Different mapping solutions, using less multicast groups are illustrated in Table 2. These mappings illustrate a significant (potential) shortcoming of the mapping process: As a result of the mapping, some nodes may receive, and be forced to filter, unwanted subscriptions. For instance, in the mapping of Table 2a, node C receives notifications from topic e which it is not subscribing. Therefore, the role of a mapping algorithm is to find a "good" mapping that minimizes this effect.

Table 2 also illustrates another disadvantage of the mapping process: some notifications may have to be sent on different groups. For instance, in the mapping of Table 2b, notifications from subject a need to be disseminated in groups g_1 and g_2. This is a minor cost compared with the problem identified above, therefore in this paper we consider only algorithms to minimize the reception of unwanted information. However, before we introduce such algorithm, it is important to derive a precise metric to measure the quality of each mapping. In order to define this metric, we need to introduce a number of auxiliary definitions.

3.2 Covering Relations

Content-based subscription can be correlated using covering relations, in a similar way as described in [6]:

- $\phi \subset_f^n \alpha$: the covering relation between an attribute filter ϕ and the corresponding notification attribute α. This relation is true if

  $\phi.name = \alpha.name \wedge \phi.type = \alpha.type \wedge \phi.operator(\alpha.value, \phi.value)$, is true;

- $s \subset_s^n n$: the covering relation of a notification n by a subscription s. This relation is true for notifications $N_S(s) = \{n \in N_a : \forall \phi \in s, \exists \alpha \in n : \phi \subset_f^n \alpha\}$, where N_a is the set of all notifications of subject a;

- $s \subset_s^s s'$: the covering relation between two subscriptions is true for $N_S(s) \supseteq N_S(s')$. This relation is transitive $\forall s, s', s'' : s \subset_s^s s' \wedge s' \subset_s^s s'' \Rightarrow s \subset_s^s s''$;

3.3 Subscription Volume

A subscription s_a issued by m is represented by the pair $(f(a), m)$, where $f(a)$ defines a filter condition on the notifications of subject a. Also, any filter can be decomposed as

$$f(s_a) = f_1(a) \vee \ldots \vee f_k(a)$$

where $f_k(a) = l_{k,1}(a_1) \wedge \ldots \wedge l_{k,n}(a_n)$, $l_{k,n}(a_n)$ is a logical condition on the attribute a_n of a, and $\forall j, k : f_j(a) \wedge f_k(a) = false$. After this, s_a can be expressed and substituted by the reunion of disjoint subscriptions, $s_a = \bigcup_k s_{a_k}$, where $f(s_{a_k}) = f_k(a)$.

We can now introduce the notion of the *subscription volume* of s_a, denoted by $V(s_a)$, as the subset of the information space of a defined by s_a. We also denote its measure by

$$\|V(s_a)\| = \sum_k \|V(f_k(a))\| = \sum_k \|V(l_{k,1}(a_1) \wedge \ldots \wedge l_{k,n}(a_n))\| = \sum_k \prod_n \|l_{k,n}(a_n)\|$$

where $V(f_k(a))$ represents the n-dimensional volume of a limited on axis a_i by $l_{k,i}$, and $\|l_{k,i}(a_i)\|$ is the size of subset $l_{k,i}(a_i)$ of a_i.

As an example, if we consider

$$class(a) = (string\ symbol \in \{ibm, msft, sun\}, float\ price \in [0.0; 1000.0])$$

and the filter

$$f(a) = (symbol = ibm \vee symbol = sun) \wedge price > \$50.0$$

we have

$$\|V(f(a))\| = \|symbol = ibm \vee symbol = sun\| \cdot \|price > \$50.0\| = 2 \cdot (1000.0 - 50.0) = 1900$$

Then, the volume of subject a is given by,

$$\|V(a)\| = \|symbol = any\| \cdot \|price = any\| = 3 \cdot 1000.0 = 3000$$

3.4 Notification Probability

The notification probability of a subscription, denoted by $P(s_a)$, is a measure of the relation between the number of received notifications after s_a, to the number of all notifications received for subject a, at a server node. A simplified formulation of the relation could be:

$$P(a) = 1$$

$$P(s_a) = \frac{\|V(s_a)\|}{\|V(a)\|}$$

Note that for the example introduced in the previous section, we would have $P(s_a) = 0.63$.

However, the previous definition is only accurate when notifications are uniformly distributed in the notification space. In practice, this distribution depends heavily on the nature of each subject and may change along time. Therefore, subject behavior must be considered when computing a more accurate value for $P()$. It is possible to estimate $P()$ based on the notifications received in the past. This can be done as follows.

When a notification n is received, the *hit* counter of every subscription is incremented where $s \subset_s^n n$ is true. From this statistical information, a better approximation for $P(s_a)$ can be computed as the quotient between s_a's hit counter and the total number of received notifications for a. So, for a new subscription s_a a search is made for earlier subscriptions s_a^- e s_a^+ such that,

$$s_a \subset_s^s s_a^- \wedge (\forall s \in S_a: \quad s_a \subset_s^s s \Rightarrow hits(s_a^-) \geq hits(s))$$

$$s_a^+ \subset_s^s s_a \wedge (\forall s \in S_a: \quad s \subset_s^s s_a \Rightarrow hits(s_a^+) \leq hits(s))$$

By definition of s_a^- and s_a^+ the following is always true,

$$P(s_a^-) \leq P(s_a) \leq P(s_a^+).$$

Being the probability a function of the hits between notifications and subscriptions,

$$hits(s_a^-) \leq hits(s_a) \leq hits(s_a^+).$$

Since the values of $hits(s_a^-)$ and $hits(s_a^+)$ limit the value for $hits(s_a)$, they serve as a good basis for computing $hits(s_a)$,

$$hits(s_a) = \left\lfloor hits(s_a^+) \cdot \frac{\|V(s_a)\| - \|V(s_a^-)\|}{\|V(s_a^+)\|} + hits(s_a^-) \right\rfloor.$$

Eventually, when searching for s_a^- and s_a^+ the following can occur:

- if $s_a^- = \emptyset$ (s_a^- doesn't exist): $hits(s_a^-) = 0$ and $\|V(s_a^-)\| = 0$;
- if $s_a^+ = \emptyset$ (s_a^+ doesn't exist) we can have:
  - if s_a is the first subscription: $hits(s_a) = 0$;
  - if $s_a \subset_s^s s_a^{root}$, s_a is the new root as defined by $\subset_s^s$: $hits(s_a) = hits(s_a^{root})$;
  - if $s_a^{root} \not\subset_s^s s_a$, $s_a^{root'}$ (the smallest cover of s_a and s_a^{root}) is the new root[1]:

[1] In this case there are no subscribers for the root.

$$hits(s_a^{root'}) = hits(s_a^{root})$$

$$hits(s_a) = \begin{cases} 0 & \text{if } V(s_a) \cap V(s_a^{root}) = \emptyset \\ \lfloor hits(s_a^{root}) \cdot \frac{\|V(s_a) \cap V(s_a^{root})\|}{\|V(s_a^{root})\|} \rfloor & \text{if } V(s_a) \cap V(s_a^{root}) \neq \emptyset \end{cases}$$

Finally $P(s_a)$ can be formulated as

$$P(s_a^{root}) = 1$$

$$P(s_a) = \begin{cases} \frac{\|V(s_a)\|}{\|V(s_a^{root})\|} & \text{if } hits(s_a^{root}) < n \\ \frac{hits(s_a)}{hits(s_a^{root})} & \text{if } hits(s_a^{root}) \geq n \end{cases}$$

where n is the minimal number of notifications in a sample. It is important to note that $P(s_a)$ is computed at each access server and depends on locally issued subscriptions and received notifications. The time cost of $P()$ is negligible, and its quality is heavily dependent on the sample taken. If necessary, state can be kept between server restarts for quality improvement.

3.5 Subject Weight

Notification probability $P()$ describes notification density across subject information space. However, to address different notification rates between subjects, we introduce the notion of *subject weight*, $W(a)$, expressed as bytes or notifications per unit time. After $W(a)$, the notion of *subscription weight* can be formulated as,

$$W(s_a) = W(a) \cdot P(s_a).$$

Note that $W(a)$ can be statically defined or computed dynamically as notifications are received by an access server for a, and it is locally defined.

3.6 A Quality Metric

We use the subject weight as a basis for our quality metric. Our quality metric captures the aggregate cost of delivering unwanted notifications using a given *map*. This function is formulated as,

$$C(map) = \sum_{g=1}^{G} C_g(map),$$

where G is the number of used groups in *map* and $C_g(map)$ is the total cost for group g,

$$C_g(map) = \sum_{m_{g,k} \in M_g} GW(g, m_{g,k}).$$

In $C_g(map)$, M_g is the set of subscribers joined at g, and $GW(g, m_{g,k})$ is the cost taken by $m_{g,k}$ for being member of g, and is given by

g_1	a	A B G	
g_2	b d	A	
g_3	c v	B G	
g_4	g h	C	
g_5	e i	C D	$W_c(E)$
g_6	u x	F J K	
g_7	t z	E H I	
Total cost			$C = W_c(E)$

(a)

g_1	a c v	B G	
g_2	a b d	A	
g_3	g h i	C	
g_4	e i	D	
g_5	u x	F J K	
g_6	t z	E H I	
Total cost			$C = 0$

(b)

Table 3: Mapping examples (with metric).

$$GW(g, m) = \sum_{a \in A_g} W(s_{a,g} \setminus s_{a,m}),$$

where A_g is the set of all subjects that have subscriptions mapped in g, $s_{a,g}$ is the union of all subscriptions $s_{a,p}$ issued by p and mapped in g, and $s_{a,m}$ the subscription of a issued by m.

Table 3 illustrates the results of applying our quality metric to the mappings of Table 2. It is interesting to observe that map (b) not only uses less groups but also exhibits no cost.

Since $C_g(map)$ is a cost function, the lower its value the better the mapping. The purpose of the genetic mapping algorithm described in this paper is to find low cost mappings for a given set of subscription and number of available multicast groups.

4 A Genetic Algorithm

As we have noted before, a mapping is a relation R in $S \times G$, where S is the set of all subscriptions and G the list of available groups. Observing the possible huge dimension of $S \times G$, an algorithm is necessary to find not the best solution, but a good one in useful response time. Our algorithm uses a genetic search approach [18, 19]. Genetic algorithms use selection techniques inspired in the biological evolution: starting from an initial population of candidate solutions, each successive generation inherits the best features from the previous generation, by applying selection, crossover and mutation between candidate solutions.

The algorithm tries to aggregate on a same group subscriptions from different subscribers, that have close subscription volumes. For each candidate mapping solution found during the search process, a fitness function is evaluated as the cost of delivering unwanted notifications. In our case, the fitness function is the quality cost function introduced in Section 3.6.

The structure of the genetic algorithm is depicted in Figure 1. Note that what differentiates most genetic algorithms is the particular set of genetic operators for each problem domain. The algorithm starts from an initial set of solutions (l1), and evaluates the fitness function for each of them (l2). Next, by applying at each generation, selection, crossover and mutation operators, new

l1: [**Start**]
Generate a first population of n candidate solutions.
l2: [**Fitness**]
Evaluate fitness function for each candidate solution.
l3: [**Acceptance**]
Place candidate solutions in population.
l4: [**Test**]
If stop condition is true, terminate and return best candidate solution found.
l5: [**Generate new population**]
Create next generation of solutions by applying the :
following steps till new population is filled:
l5.1: [**Selection**]
Select two solutions from current population according to its fitness
(the better the fitness is, the more chances a solution has to be chosen).
l5.2: [**Crossover**]
If some probability is meet,
create new candidates by exchanging parts of the two solutions above.
If probability is not meet, the new candidates are copies of the solutions.
l5.3: [**Mutation**]
Change each new candidate if some probability is meet.
l6: Return to l2.

Figure 1: General structure of a genetic algorithm.

solution K								
g1	*s1*				*s5*		*s7*	
g2			*s3*			*s6*		*s8*
...	...							
gN		*s2*		*s4*				

Figure 2: Mapping solution representation.

solutions are derived from each population and the best are retained. At the end, a solution is chosen that more reduces the chances of delivering unwanted notifications to local subscribers.

Solution Encoding In any genetic algorithm each individual in a population must contain necessary information respecting to the solution it represents (commonly known as the genetic material). Considering our problem nature, each individual must list subscriptions for each group, as shown in the Figure 2. In this figure, each line represents a group and lists subscriptions mapped into it. Here, blank spaces stand for subscriptions mapped into other groups. Crossover is made easier after this encoding and by orderly listing subscriptions in groups.

Initial population The choice of good mapping candidates to constitute the initial population can speed up the search process. The idea consists in starting with candidate solutions closer to the final map. One immediate possibility is to aggregate on the same group all subscriptions of a common subject. This explores subscription volume proximity. Other possibility is to group subscriptions based on subscriber geographical neighborhood. This has applications in

collaborative systems, distributed games, and generally on geographical based applications. The initial population must be selected according to the application problem domain and expected subscription patterns.

Selection During selection, a pair of candidate solutions is taken from the current population, according to their better fitness value when compared to other candidates. However, choosing only the best individuals can lead to a quicker population specialization, and soon throw the search into a local minimum of the cost function $C(map)$. That's why selection should give a chance for weaker solutions to be chosen, at least in the initial stages of the search. Here, the algorithm tries to improve population diversity on initial generations, which conduces to a greater scope search of the solution space. Considering these facts, the algorithm initially selects solutions by tournament, and on later iterations, roulette-wheel selection is used to promote candidate population specialization [18].

Crossover This operation consists in exchanging solution parts between two individuals in a population, hoping that the new candidates are better than their parents. Crossover is accomplished by exchanging subscription sequences between groups of each parent. Finding sequences is made easier by the encoding. Before crossover, a pair of subscriptions is chosen as crossover points. Since subscriptions are ordered by their identifier inside each encoded group, the crossover points establish the limits of the sequences to exchange. So, as shown in Figure 3, each new solution is obtained from two parents in the following way: a new solution inherits the subscriptions at the left of the first crossover point and at the right of the second crossover point from one parent, and the middle subscriptions from a random permutation of the correspondent subscriptions of the other parent. Note that, from a pair of initial solutions, a new pair is created. It is useful to note an important property of the used encoding: the crossover of a valid pair of solutions results in another valid pair of solutions, without lost or repetition of subscriptions. Also, this encoding easily supports multiple crossover points.

Mutation Mutation changes a particular solution, hoping that it jumps from a local fitness optimum to a more promising position in the search space. Mutation promotes changes to avoid population stagnation in successive generations. As Figure 4 shows, mutation happens by exchanging subscriptions between a pair of groups in a solution. In each group, subscriptions are chosen by a random selection mask. This operation always produces a new valid candidate.

Acceptance During acceptance, created candidates are selected according to their fitness, to form the next population. Eventually, this process can drop some good solutions if care is not taken. To avoid this, generally an elitist decision is made, and the best solutions from the current population are also kept in the new population.

Test The algorithm must stop after a predefined number of iterations without any improvement, or when a significant improvement over the current mapping

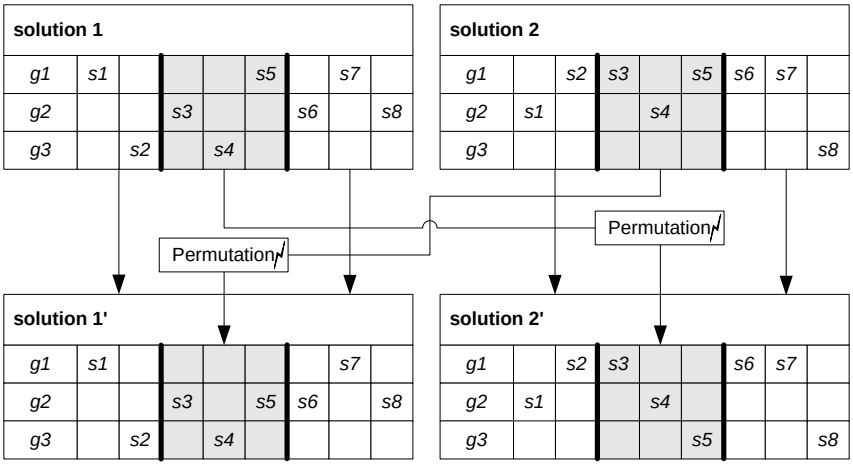

Figure 3: Crossover between solutions 1 and 2.

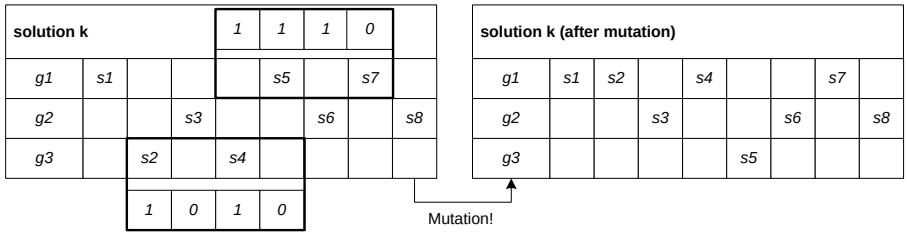

Figure 4: Mutation of a solution.

has been achieved. At the end, the best solution seen is made the new map. If a tie happens, the solution with fewer groups is chosen.

5 Evaluation

The algorithm evaluation was done in Mathlab 6.0, using scripts and a C extension for quicker computation of the fitness function. This tool was chosen because it provides an interpreter for rapid algorithm writing, refinement and test. We have evaluated the efficiency of the mapping solutions, measured as the ratio between desired and total received notifications by all subscribers. We ran an initial set of tests to find good values for the genetic parameters. Afterwards, these parameters were used to configure the algorithm used to run the remaining tests. Subsequent tests evaluate the efficiency of the solutions in subject-based and content-based scenarios. We also compare our solution with the two most common approaches in current publish-subscribe systems: *i)* broadcast [3] and; *ii)* round-robin [4] (where subscriptions are distributed to groups by order of arrival).

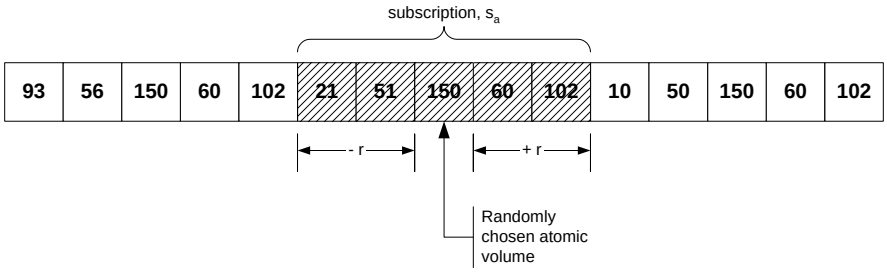

Figure 5: Atomic volumes for a subject and a subscription.

5.1 Modelling Content-Based Subscriptions

Content-based subscriptions are modelled by dividing the information space of each subject in a set of indivisible disjoint volumes of different sizes, which we call *atomic volumes*. Figure 5 illustrates a possible division of the information space, where the number in each block indicates the size of the correspondent volume. The size of each volume is randomly chosen between 1 and 255. Each individual subscription corresponds to a subset of consecutive atomic volumes, formed by the r volumes at the left and the right of a randomly selected volume, where r is the *subscription radius*. A subscriber is simulated by generating a total of k subscriptions, between its selected subjects. This strategy allows us to model in a generic way a wide range of subscription patterns (note that subjects can be decomposed in any required number of atomic volumes). In our tests, there were 64 atomic volumes on each subject, with sizes from 1 to 255. The subscription radius varied between 10 and 20.

5.2 Common Simulation Parameters

In all tests, subjects were grouped into one or five categories, and each subscriber randomly selected one category for issuing all subscriptions. Subjects were chosen by an uniform distribution, and all subscribers selected an equal number of subjects. All subject-based subscriptions were selected for multicast distribution because of their small number. Since there were a lot more of content-based subscriptions, only 70% of the heavier ones (as of $W(s_a)$) were selected for group mapping, and the remaining 30% were considered for point-to-point delivery. Initial populations were formed by a flooding candidate mapping (one group only), many of round-robin subject to group mapping, and those of randomly mapped subscriptions. The number of genetic search iterations was set to 500. Before evaluating the quality of the solutions returned by our algorithm, we conduced some tests to find out a good combination of genetic parameters. These tests revealed good results for a population of 20 individuals, a crossover probability of 90%, a mutation probability of 20%, and an elitism of 5 individuals. Finally, the quality of every returned mapping solution was determined by simulating the emission of 1000 notifications, according to subject throughput.

5.3 Efficiency Tests

We now present the efficiency of the different mapping algorithms under different conditions. In all figures, the caption at the bottom of the figure indicates the following simulation parameters: number of categories (c), relative traffic throughput, measured as the maximum ratio between any subjects's traffic throughput, stated as a power of 10 (p) and number of available groups (g). For instance, the graph in Figure 6(a) was obtained using 1 category (c1), a relative traffic throughput of 1 (p0) and 16 available groups (g16). In all graphs, the xx axis represents the number of subscribers, and the yy axis represents solution efficiency, expressed as the ratio between desired and total received notifications by all subscribers. In a system with no restrictions on the number of available groups, this value would be 1 (i.e., all received notifications would correspond to subscribed notifications).

Efficiency of broadcast, round-robin, and genetic solutions in subject-based. We compare, for different subject-based scenarios, the efficiency of the solutions returned by the broadcast, round-robin, and genetic algorithms. The number of subscribers (xx axis) were varied from 20 to 160. A set of 100 subjects has been defined, and each client subscribed for 10 subjects from a chosen subject category. All mappings were done with 16 groups. The results are depicted in Figure 6.

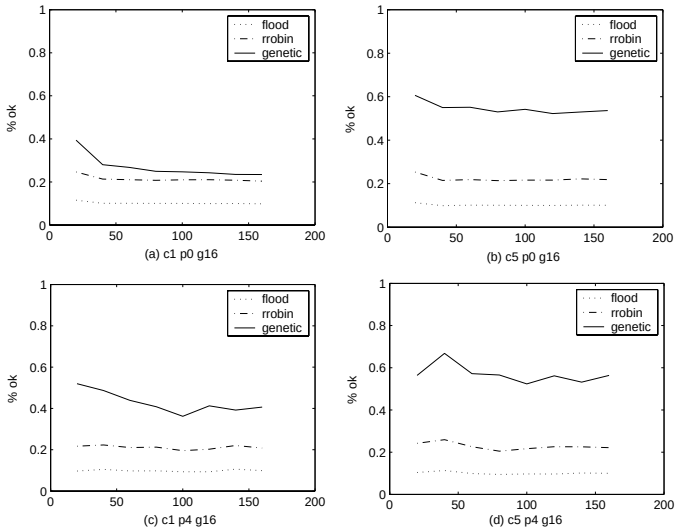

Figure 6: Efficiency of different solutions in subject-based.

As expected, broadcast provides the the worst results in all scenarios. From (a) to (b), we increased the number of subject categories from 1 to 5, and the genetic algorithm returned approximately 3 times the quality of round-robin. From (a) to (c), we increased the maximum relative traffic throughput from 0 to 10^p, and again, genetic mappings were 2 times better than round-robin. It can be observed that the genetic algorithm clearly outperforms the broadcast and the round-robin approaches when subjects are more correlated, subscribers have closer interests, and subject relative throughput is preponderant. An example of

such applications are those where geographic factors are determinant in subject selection.

Efficiency of broadcast, round-robin, and genetic solutions in content-based. We compare the efficiency of the solutions returned by the broadcast, round-robin, and genetic algorithms, for different content-based scenarios. We took the same subject-based scenario configurations, but in this case, we set the subscription radius from 10 to 20 volumes of a 64 atomic volume information space. Each client issued between 4 and 8 subscriptions from each subject selected.

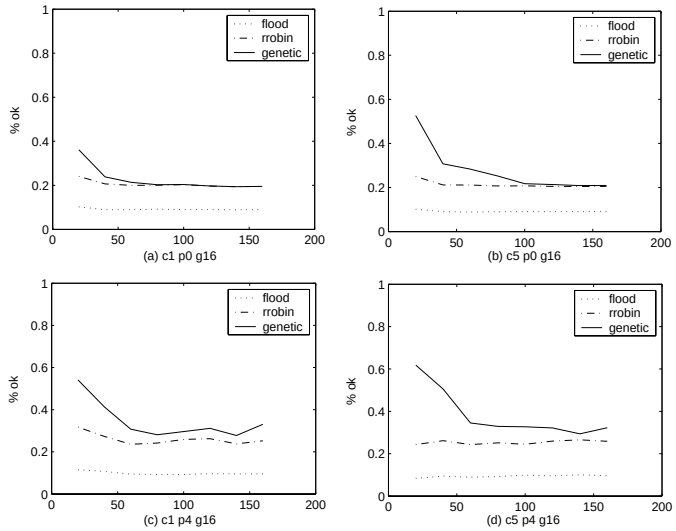

Figure 7: Efficiency of different solutions in content-based.

From the results depicted in Figure 7, we conclude that only the combined effect of subject correlation and relative throughput (d), contribute for the genetic algorithm to return better mappings than round-robin. Since in these tests there were about 450 subscriptions instead of 100 in subject-based ones, the results show that in the presence of a large number of diverse subscription patterns, the efficiency of the mapping is limited if only a small number of groups is available. This fact is confirmed by the tests presented in the following paragraphs.

Number of groups. We evaluate the influence of the number of available groups on the genetic algorithm's efficiency. For this, we reused some of the subject-based and content-based configurations, and changed the number of available groups. As it would be expected, a larger number of groups improves the genetic solutions's efficiency in both subject-based (Figure 8a) and content-based (Figure 8b) subscriptions.

Figure 9 compares the impact of the number of available groups in the broadcast, round-robin, and genetic algorithms, for a content-based scenario. It can be observed that an increase on group quantity produces better mappings for the genetic case than for round-robin.

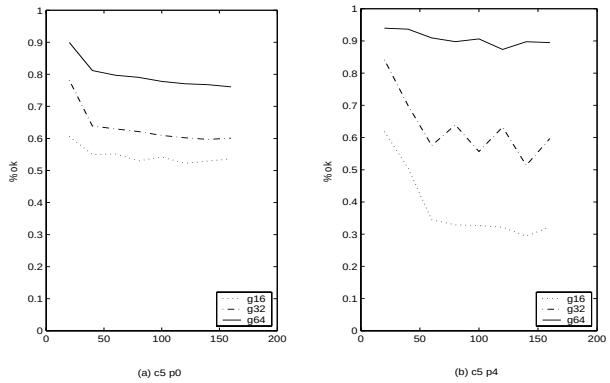

Figure 8: Group availability influence on genetic solutions's efficiency.

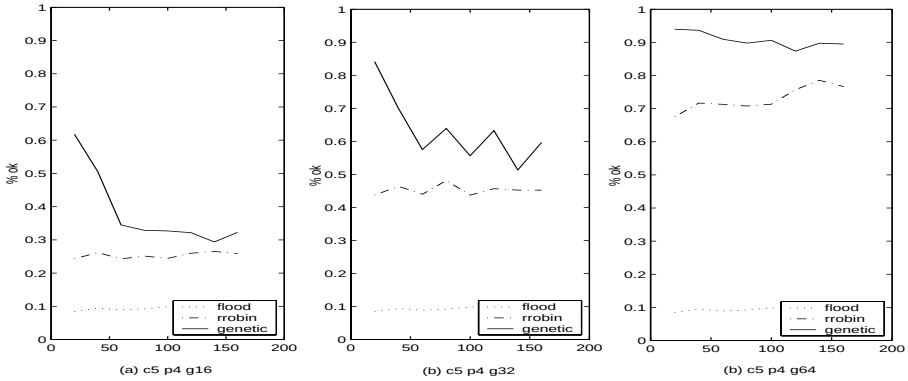

Figure 9: Comparison of group availability influence between various algorithms.

Number of subscribers. We measure the impact of the number of subscribers in genetic solutions's efficiency. Tests were done for both subject-based (Figure 10a) and content-based (Figure 10b), and only the number of subscribers changed. After an increase on the number of subscribers, solution efficiency tend to be constant when other parameters remain constant, even with an increase on notification misdeliveries. This happens because, the greater the number of subscribers is, the greater is the probability of issuing similar subscriptions.

Number of subjects. We evaluate the impact of the number of subjects subscribed by each client in genetic solutions's efficiency. In both subject-based (Figure 11a) and content-based (Figure 11b) scenarios, clients subscribed from the same category, the equivalent of a percentage of all available subjects. We conclude that, either if we only increase the number of total subjects (lines $10\% - 100$ and $10\% - 200$), or we only reduce the percentage of subscribed subjects (lines $10\% - 200$ and $5\% - 200$), we observe a decrease on solution efficiency. This is related to the reduction in similarities between issued subscriptions, in the first case because subscriptions are more dispersed in information space, and in the second case because there are less subscriptions and so these tend to be more dispersed.

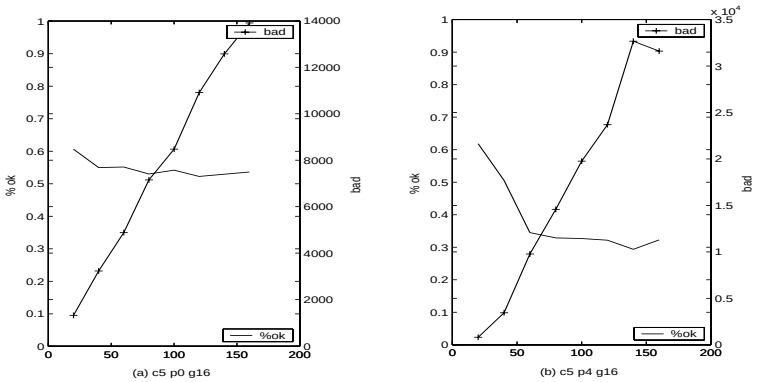

Figure 10: Efficiency stability.

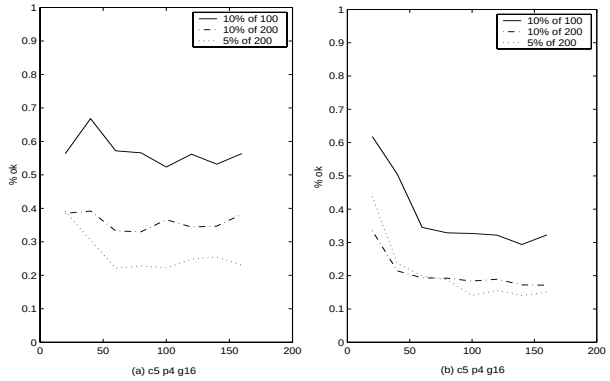

Figure 11: Influence of subject quantity on genetic solution efficiency.

Other tests. We also found that tournament selection should be used between the first 40% $\sim$ 50% iterations, and the best solution probability should be between 70% and 90%. In the subsequent iterations, roulette-wheel selection permits a quicker population specialization, converging into a good local solution. Random permutation at crossover can be used in the first 5% iterations to promote a breather search of the solutions space. In subsequent iterations, when search convergence is required, the permutation technique should be avoided.

5.4 Execution Time

This evaluation was run on an Intel Pentium III 600 Mhz with 128 MB RAM. All Mathlab 6.0 scripts were compiled by MCC (the Mathlab compiler), and the fitness function was written as a C MEX-file extension for quicker execution.

From the population's fitness computation time records shown in the table, if search iterates 250 times, which by our experiments is sufficient, total time for the first line is approximately $28.25s$. An optimized implementation of the algorithm, detached from the Mathlab simulation environment, would perform significantly faster.

P	G	R	S	T (sec)
20	16	160	100	0.113
20	32	160	100	0.106
20	16	320	100	0.220
20	16	160	300	0.640
40	16	160	100	0.251

Table 4: Population's fitness computation times (**P**opulation, **G**roups, **R**eceivers and **S**ubscriptions).

5.5 Discussion

Generally, results show that the genetic search algorithm finds better maps than round-robin, specially when subscriptions are similar and traffic throughput is different between subjects. Also, even a simple algorithm such as round-robin offer better performance than the use of broadcast, confirming the usefulness of multicast in publish-subscribe systems. As observed in Figure 7, the greater the number of subscriptions, the greater is the difficulty to return good maps. Nevertheless, as shown in Figure 9, the genetic algorithm makes a better use of an increase in the number of available groups. The evaluation suggests that it is important to reduce the number of subscriptions used as an input to the algorithm. One way to achieve this goal is to decompose each subject's subscriptions into a disjoint set, and then, use unicast to disseminate low-throughput subscriptions and notification with a small number of subscribers.

6 Conclusions

This paper presents a novel algorithm to map subscriptions into a limited number of multicast groups. The algorithm increases the efficiency of data dissemination in publish-subscribe systems. The large space dimension of this search problem makes infeasible to find optimal solutions in due time. We address this issue by proposing a genetic search algorithm to derive efficient mapping solutions. The paper proposes the fitness, encoding, selection, crossover, mutation and acceptance functions for this algorithm. The efficiency of the proposed solution has been extensively evaluated using simulations and its results compared with simpler approaches such as broadcast and round-robing. Results have show that the genetic search algorithm clearly outperforms round-robin or broadcast in many scenarios, namely when subscriptions are similar and traffic throughput is different between subjects.

Acknowledgments

The authors are grateful to Graça Gaspar for commenting earlier versions of this paper.

References

[1] David Cheriton. Dessimination-oriented communication systems. Technical report, Computer Science Department of Stanford University.

[2] Brian Oki, Manfred Pfluegl, Alex Siegel, and Dale Skeen. The information bus - an architecture for extensible distributed systems. *Operating Systems Review*, 27(5), December 1993.

[3] TIBCO. Tib/rendezvous white paper. Technical report, TIBCO. Available from http://www.rv.tibco.com/whitepaper.html.

[4] David Glance. Multicast support for data dissemination in orbixtalk. In *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 1996.

[5] Bill Segal and David Arnold. Elvin has left the building: A publish/subscribe notification service with quenching. In *Proceedings of AUUG97*, Brisbane, Australia, September 1997. Available from http://www.dstc.edu.au/.

[6] Antonio Carzaniga. *Architectures for an Event Notification Service Scalable to Wide-area Networks.* PhD thesis, Politecnico di Milano, December 1998. Available from http://www.cs.colorado.edu/~carnaziga/papers.

[7] Brian Neil Levine, Jon Crowcroft, Christophe Diot, J. J. Garcia-Luna-Aceves, and James F. Kurose. Consideration of receiver interest for IP multicast delivery. In *INFOCOM (2)*, pages 470–479, 2000.

[8] Guruduth Banavar, Tushar Chandra, Bodhi Mukherjee, Jay Nagarajarao, Robert E. Strom, and Daniel C. Sturman. An efficient multicast protocol for content-based publish-subscribe systems. In *International Conference on Distributed Computing Systems (ICDCS '99)*, June 1999. Available from http://www.research.ibm.com/gryphon/.

[9] Object Management Group. *Object Management Group. CORBAservices: Common Object Services Specification - Event Service Specification.*, March 1995. Available from http://www.omg.org/library/csindx.html.

[10] B. Kantor and P. Lapsley. Network news transfer protocol - a proposed standard for the stream-based transmission of news. Technical report, IETF, February 1986. RFC 977.

[11] G. Cugola, E. Di Nitto, and A.Fuggetta. Exploiting an event-based infrastructure to develop complex distributed systems. In *Proceedings of the 20th International Conference on Software Engineering (ICSE 98)*, Kyoto, Japan, April 1998.

[12] Robert Strom, Guruduth Banavar, Tushar Chandra, Mark Kaplan, Kevan Miller, Bodhi Mukherjee, Daniel Sturman, and Michael Ward. An information flow based approach to message brokering. In *International Symposium on Software Reliability Engineering '98*, 1998. Available from http://www.research.ibm.com/gryphon/.

[13] Hewlett Packard Keryxsoft. *Keryx Version 1.0a Release Notes and Documentation*, 1997. Available from http://keryxsoft.hpl.hp.com/keryx-1.0a/html/index.html.

[14] Object Management Group. *Object Management Group. CORBAservices: Common Object Services Specification - Notification Service Specification.*, March 1998. Available from http://www.omg.org/library/csindx.html.

[15] Sun Microsystems. *Sun Microsystems. Java Message Service.* Available from http://java.sun.com/products/jms.

[16] Mário Guimar aes. Técnicas para aumento da capacidade de escala em sistemas de publicação e subscrição de informação. Master's thesis, Faculdade de Ciências da Universidade de Lisboa, February 2002.

[17] Tina Wong, Randy Katz, and Steven MacCanne. A preference clustering protocol for large-scale multicast applications. In *Proceedings of the First International Workshop on Networked Group Communication*, November 1999.

[18] Melanie Mitchel. *An Introduction to Genetic Algorithms.* The MIT Press, Cambridge, MA, 1996.

[19] D. E. Goldberg. *Genetic algorithms in search, optimization, and machine learning.* Addison-Wesley, Reading, MA, 1989.