Efficient Probabilistic Subsumption Checking for Content-Based Publish/Subscribe Systems

Aris M. Ouksel[1,⋆], Oana Jurca[2], Ivana Podnar[2], and Karl Aberer[2,⋆⋆]

[1] The University of Illinois at Chicago
Depts. Of Information and Decision Sciences and Computer Science
aris@uic.edu
[2] School of Computer and Communication Sciences
Ecole Polytechnique Fédérale de Lausanne (EPFL)
CH-1015 Lausanne, Switzerland
{oana.jurca, ivana.podnar, karl.aberer}@epfl.ch

Abstract. Efficient subsumption checking, deciding whether a subscription or publication is covered by a set of previously defined subscriptions, is of paramount importance for publish/subscribe systems. It provides the core system functionality—matching of publications to subscriber needs expressed as subscriptions—and additionally, reduces the overall system load and generated traffic since the covered subscriptions are not propagated in distributed environments. As the subsumption problem was shown previously to be co-NP complete and existing solutions typically apply pairwise comparisons to detect the subsumption relationship, we propose a 'Monte Carlo type' probabilistic algorithm for the general subsumption problem. It determines whether a publication/subscription is covered by a disjunction of subscriptions in $O(k\ m\ d)$, where k is the number of subscriptions, m is the number of distinct attributes in subscriptions, and d is the number of tests performed to answer a subsumption question. The probability of error is problem-specific and typically very small, and sets an upper bound on d. Our experimental results show significant gains in term of subscription set reduction which has favorable impact on the overall system performance as it reduces the total computational costs and networking traffic. Furthermore, the expected theoretical bounds underestimate algorithm performance because it performs much better in practice due to introduced optimizations, and is adequate for fast forwarding of subscriptions in case of high subscription rate.

1 Introduction

Content-based publish/subscribe systems are receiving growing interest with a large number of relevant applications such as stock tickers, RSS news feeds, network monitoring, traffic monitoring, and electronic commerce requiring selective information

⋆ Research supported in part by the National Science Foundation grant IIS-0326284.
⋆⋆ Research supported in part by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322 and carried out (partly) in the framework of the EPFL Center for Global Computing.

M. van Steen and M. Henning (Eds.): Middleware 2006, LNCS 4290, pp. 121–140, 2006.

dissemination. Traditional content-based publish/subscribe systems usually employ high-performance servers to handle high rates of publications and serve millions of subscribers in static environments. They have been optimized for fast matching of publications to subscriptions [1,2,3,4] and typically maintain a special subscription index that does not frequently change as the rate of subscription changes is negligible compared to the publication rate.

Distributed content-based publish/subscribe systems traditionally assume static environments and use a network of brokers to divide the publication and subscription load. Brokers implement routing protocols to provide a consistent service with the goal of reducing networking costs generated by publications and subscriptions [5,6]: Subscriptions are typically routed through the network toward publishers to enable filtering of publications close to their sources. Subscription traffic, on the other hand, is reduced by not propagating covered subscriptions, as they are redundant, or by subscription merging [7,8].

Although the importance of subscription set reduction for content-based publish/subscribe systems has been stressed, e.g. in [8], existing deterministic algorithms [9,6,7] focus either on efficient matching of publications to subscriptions only or rely on basic heuristics for subscription set reduction such as pairwise subscription comparison or subscription merging. In this paper we take a more fundamental approach to subscription set reduction for (distributed) content-based publish/subscribe systems. In particular we show that when using general subsumption checking, where the covering of subscriptions by multiple other subscriptions is exploited, important performance improvements can be achieved. However, efficient general subsumption checking is non-trivial. Publications and subscriptions are typically modeled as logical expressions–conjunctions of predicates–where each predicate defines a simple constraint on an attribute. Geometrically, subscriptions can be viewed as convex polyhedra. Therefore, the general subsumption checking problem corresponds to the problem of checking whether a disjunction of subscriptions covers a subscription/publication, which can geometrically be interpreted as checking whether a convex polyhedron is contained within a finite union of convex polyhedra. This problem was proven to be co-NP complete in [10].

Since the general subsumption problem is practically unfeasible, for solving it, we introduce a probabilistic 'Monte Carlo type' algorithm. This is the first probabilistic approach to test the subscription coverage by a union of subscriptions. The algorithm solves the subsumption problem in $O(k \cdot m \cdot d)$, where k is the number of subscriptions, m is the number of distinct attributes in subscriptions and d is the number of tests performed to answer the subscription coverage question. The value of parameter d is dependent on an acceptable predefined probability of error which is problem specific and can be computed in polynomial time a-priori. Using this algorithm a subscription set can be efficiently reduced to a minimized subscription set matching the same set of publications. Experiments show that in practice our algorithmic approach performs much better than the theoretical bound $O(k \cdot m \cdot d)$. The same algorithm can also be used to efficiently match publications from imprecise data sources, by representing publications also as convex polyhedra, as it is advocated in recent publish/subscribe models with approximate matching [11].

The importance of subscription set reduction is highly significant in distributed content-based publish/subscribe system for the following reasons:

- The publish/subscribe systems architecture is increasingly used in environments with highly variable subscriptions, such as MANETs and sensor networks, where the assumption of both network [12] and subscription stability no longer holds. The rate of subscription changes may drastically increase as a consequence of both changing interests and context changes, and also may substantially exceed the publication rate if rare events are monitored; therefore, novel indexing techniques have been investigated that trade-off precision to performance [13], however they do not solve the essential problem of subscription set reduction.
- As publish/subscribe systems mainly target usage scenarios where a subscription space is moderately populated and subscriptions typically overlap due to similar but not equal interest, there is a higher probability of a subscription being covered by a set of subscriptions rather than a single one. Covered subscriptions are redundant. Therefore, they are not propagated further which reduces the total number of subscriptions in the system saving memory and reducing traffic. This in turn reduces computational costs for matching publications to subscriptions and new subscriptions to existing subscriptions as the set of subscriptions is reduced.
- As publish/subscribe systems are growing in scale to very large networks of brokers, the benefit of any reduction in the number of subscriptions forwarded locally by a broker, is amplified exponentially in the network diameter while broadcasting subscriptions in the broker network. Thus even modest local reductions lead to substantial global reductions in network traffic during subscription propagation.

Due to the probabilistic nature of the algorithm a concern about lost publications (false negatives) may be raised. However, many recent applications are tolerant to lost publications, because e.g. the data sources are already unreliable themselves, as in sensor networks. Furthermore, the error probability can be controlled and adapted to application needs, trading off computational cost for precision. Therefore, we expect that for a wide range of important applications the probabilistic nature of the approach is fully acceptable.

To summarize, the algorithm has the potential to significantly decrease costs in terms of computation, memory, and bandwidth consumption in content-based and distributed publish/subscribe systems by fully exploiting the potential subscription set reduction and achieving computational efficiency through a probabilistic approach. In our experimental evaluations we verify both the performance gain with respect to subscription set reduction by comparing to the standard technique of pairwise reduction and the performance characteristics of the algorithm as compared to the pessimistic theoretical bounds.

The remainder of the paper is structured in the following way. We review the basic principles of content-based publish/subscribe communication model in Section 2. To motivate the presentation, Section 3 sketches a usage scenario and formally defines the subsumption problem. Section 4 presents our novel probabilistic algorithm with specific optimizations, and we investigate it's properties in a distributed setting in Section 5. Section 6 presents an evaluation of the algorithm using extensive experimentation, and

in Section 7 we compare it to the related work in the field. We complete the paper with our conclusions in Section 8.

2 Distributed Publish/Subscribe Communication

The publish/subscribe interaction model enables asynchronous communication between information *publishers* and *subscribers*. Subscribers express interest in receiving publications that comply to specific criteria by defining *subscriptions* that change the set of active subscriptions maintained by the publish/subscribe system. When a publisher defines a new *publication*, it is compared against all active subscriptions, and the system notifies subscribers with a matching subscription about the published content. Thus, the publish/subscribe service performs content filtering and enables push-style group communication, where group members are determined dynamically per each publication.

In a distributed system a set of publishers $P_i, 1 \leq i \leq n$ and a set of subscribers $S_j, 1 \leq j \leq m$ interact over a set of nodes, *brokers*, $B_k, 1 \leq k \leq N$. Brokers are responsible for matching publications to subscriptions and for disseminating publications to neighboring brokers with subscribers interested in the published content. A publication matches a subscription if all publication attributes satisfy constraints defined by the subscription. The simplest approach to route publications in a broker network is publication flooding, where end brokers perform publication filtering prior to final delivery to local subscribers. This approach is an obvious solution for scenarios with a densely covered subscription space where most brokers have interested subscribers for all publications; however, it wastes a lot of bandwidth in cases with few or no subscribers interested in a large fraction of publications.

To decrease the publication traffic, subscriptions are disseminated through the network close to publishers to enable publication filtering 'at the source'. Upon receiving a new subscription, a broker will forward it to its neighbors that are potential publishers of content matching the defined subscription. A commonly used technique for subscription dissemination is flooding: A subscription is sent to all neighbors except to the one from which it was received. Note that brokers maintain a routing table with a set of active subscriptions per each neighboring broker, and consider this neighbor to be a subscriber without knowing the 'real' end subscribers. Upon receiving a publication, a broker B_i forwards it to its neighboring broker B_j only if it matches any of B_j's subscriptions. In other words, publications follow the reverse direction of subscriptions. The technique originates from IP muticast and is commonly known as *reverse path forwarding* [5,6].

To reduce the subscription traffic, subscription covering and merging is applied. Informally, a subscription s_1 covers subscription s_2 if all publications matching s_2 will also match s_1, but the opposite does not hold. Since a covered subscription does not influence the propagation of publications, there is no need to forward it to neighboring brokers. Therefore, when a broker B_i receives s_2 which is covered by s_1, it will not forward s_2 to B_j if B_i has previously forwarded s_1 to B_j. Nevertheless, s_2 has to be stored in the *passive set of subscriptions* (s_1 would be an element of the *active set*), because it must be activated in case s_1 expires, i.e. a subscriber unsubscribes from s_2. The process of merging proposes a single merged subscription for similar subscriptions, but will not be discussed in detail as it is beyond the scope of this paper.

3 Problem Statement

Scenario. To motivate the need for an efficient subsumption checking mechanism, we introduce a usage scenario potentially generating a large number of subscriptions. *Resource discovery in Grids* assigns computation requests (jobs) to available services. Current systems use server-based solutions and recently P2P-based solutions have been investigated [14] to deal with the scalability problem caused by a large number of jobs and services. Let us discuss the problem of resource discovery in terms of publish/subscribe. Services offering computational resources may announce their capabilities and availability through subscriptions to enable efficient matching and scheduling of jobs searching for available services. Jobs define their requirements from the services using publications. An example subscription with two publications are presented in Table 1.

Table 1. Subscription and publication examples

	$CPUcycles$	$disk$	$memory$	$service$	$time$
s_1	[3000, 3500]	[40, 50kB]	1GB	a.service.org	[2006-03-31T16:00:00, 2006-03-31T20:00:00]
p_1	3500	45kB	1GB	*.service.org	2006-03-31T16:00:00
p_2	1035	45kB	0.5GB	*.*.org	2006-03-31T12:23:05

The basic characteristic of the presented usage scenario is the potentially large number of services and jobs that generate huge amounts of both subscriptions and publications. Dynamic changes of subscriptions are significant because as the context changes, i.e. services get allocated to new jobs, subscriptions will consequently change. Therefore, this scenario exemplifies a setting where context changes induce higher subscription rate, as it can also be observed in mobile environments. Next, the subscription space may have high dimensionality: Even in our simple example without detailed job and resource descriptions, 5 different attributes have been defined. Thus, we propose a method for reducing the total number of active subscriptions in the system by means of group coverage. Due to large numbers and inherently distributed characteristics of Grid services, the publish/subscribe service for resource discovery would be distributed. As in this paper we are focusing on the subsumption process performed within a single node, we are not assuming neither an underlying network topology nor stability of the broker network. It can be applied with various routing protocols, and our goal is to point out potential impact of the proposed algorithm on the performance of a distributed system regardless of its topology and applied routing strategy.

Let us consider the following example of subscription coverage in a 2-dimensional subscription space. Table 2 defines two existing subscriptions, s_1 and s_2, and new subscription s. We want to determine whether s_1 and s_2 jointly cover s. As it is visible from the graphical representation of subscriptions in Figure 1, the subsumption relationship indeed exists. Even though neither s_1 nor s_2 cover s, their union entirely covers s. Note that constraints in this example define ranges to simplify the presentation, and can straightforwardly be extended to finite sets [15].

Table 3 lists the notation used in the paper.

Table 2. Subsumption example: $s \sqsubseteq (s_1 \vee s_2)$

$Subscription\ s$
$[x_1 \geq 830 \wedge x_1 \leq 870 \wedge$
$x_2 \geq 1003 \wedge x_2 \leq 1006]$
$Subscription\ s_1$
$[x_1 \geq 820 \wedge x_1 \leq 850 \wedge$
$x_2 \geq 1001 \wedge x_2 \leq 1007]$
$Subscription\ s_2$
$[x_1 \geq 840 \wedge x_1 \leq 880 \wedge$
$x_2 \geq 1002 \wedge x_2 \leq 1009]$

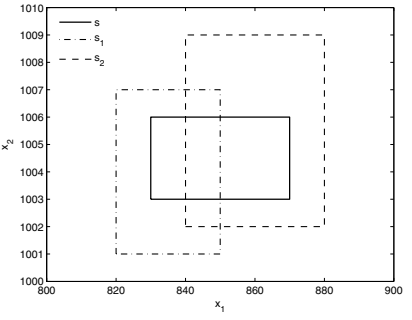

Fig. 1. Graphical representation of subscriptions in Table 2

Table 3. Notations

Symbol	Meaning
s	New subscription
p	Publication
S	Disjunction of existing subscriptions s_i, $1 \leq i \leq k$
k	$\|S\|$
s_i	Existing subscription $s_i \in S$
s_i^j	j^{th} predicate in s_i
x_j	Attribute j
m	number of distinct attributes in S
T	Conflict table
T_i^j	Value in row i, column j of T
t_i	Number of defined elements in row i of T
f_{c_i}	Number of conflict-free elements in row i of T
δ	Error probability
ρ_w	Probability of guessing a point witness

Definition 1. *Subscription* s_i is a conjunction of predicates $s_i = s_i^1 \wedge s_i^2 \wedge \ldots \wedge s_i^{r_i}$ where each s_i^j is a simple predicate, and $r_i \geq 1$, where r_i is the number of simple predicates forming subscription s_i. Let us define m, as the number of distinct attributes in the set of k subscriptions s_i, $1 \leq i \leq k$.

Without restricting the applicability of the algorithm and to simplify the analysis, we consider that each simple predicate defines a constraint on an attribute x_j, $1 \leq j \leq m$, where each x_j has a lower ($x_j \geq low_j$) and upper limit ($x_j \leq high_j$). Each attribute is therefore defined as a range. Furthermore, we assume that all subscriptions define constraints for the same number of attributes $m_1 = m_2 = \ldots = m_k = m$, and since there is a lower and upper bound on each x_j, $r = 2 \cdot m$. In fact, this is not a restriction as the bounds $(-\infty, +\infty)$ mean the attribute is not significant for a particular subscription, and remains undefined.

The *general subsumption* problem tests whether a subscription s is covered by a disjunction of subscriptions, $s \sqsubseteq (s_1 \vee s_2 \vee \ldots \vee s_k)$, where k is the total number of existing subscriptions.

Definition 2. A *conflict table* T is a $k \times (2 \cdot m)$ table relating a subscription s to all simple predicates defined by $S = \{s_1 \vee s_2 \vee \ldots \vee s_k\}$. An element in table T, T_i^j is $\neg s_i^j$ if $s \wedge \neg s_i^j$ is satisfiable or is otherwise *undefined*.

A conflict table points out conflicting and not covered intervals between a tested subscription and a set of subscriptions. To construct the conflict table, we process each subscription $s_i \in S$ to verify the satisfiability of the negation of each simple predicate s_i^j against subscription s. If the condition is true, T_i^j is assigned the value $\neg s_i^j$, otherwise it is assigned the *undefined* value. Thus, the decision whether a specific T_i^j is defined is done in $O(1)$ and the construction of the table requires $O(m \cdot k)$.

For the example in Table 2, $s \wedge \neg s_1^1$ is *not satisfiable*, because the intersection between s and $\neg s_1^1 = \{x_1 < 820\}$ is empty, while $s \wedge \neg s_1^2$ is *satisfiable* because the intersection between s and $\neg s_1^2 = \{x_1 > 850\}$ is non-empty. Both $s \wedge \neg s_1^3$ and $s \wedge \neg s_1^4$ are *not satisfiable* and thus the corresponding table cells are $undefined$. The same procedure is performed to compare s to s_2.

Table 4. Conflict table for the example in Figure 1

s_i	$x_1 < low_i^1$	$x_1 > high_i^1$	$x_2 < low_i^2$	$x_2 > high_i^2$
s_1	$undefined$	$x_1 > 850$	$undefined$	$undefined$
s_2	$x_1 < 840$	$undefined$	$undefined$	$undefined$

The conflict table relating subscription s from Table 2 to the set of subscriptions s_1 and s_2 is given in Table 4. The first row represents a template for the content of the actual conflict table relating s to s_1 and s_2. The first line corresponding to s_1 has only one defined element, $\neg s_1^2 = \{x_1 > 850\}$ because, as it is visible in the graphical representation, s_1 does not cover s for $x_1 > 850$. Analogously, the only defined element in the second line corresponding to s_2 is $\neg s_2^1 = \{x_1 < 840\}$.

Definition 3. A *polyhedron witness* to non-cover is a set of elements from a conflict table T, $\left\{T_1^{j_1}, \ldots, T_k^{j_k}\right\}$, such that $s \wedge \neg s_1^{j_1} \wedge \ldots \wedge \neg s_k^{j_k}$ is satisfiable, defining a convex polyhedron. In other words, a polyhedron witness is a convex polyhedron contained in s, but not in S.

Let us consider the example graphically represented in Figure 2, defining two subscriptions s_1 and s_2 that do not cover subscription s. The *polyhedron witness* to non-cover is a rectangle in this case, and is defined by the intersection of s and the element $\neg s_2^2 = \{x_1 > 870\}$. This rectangle is contained in s, but not in s_1 nor s_2.

Definition 4. A *point witness* to non-cover is a point that satisfies s, but does not satisfy S. A point witness is inside a polyhedron witness, but not inside S.

Table 5. Non cover example: subscriptions

$Subscription\ s$
$[x_1 \leq 890 \wedge x_1 \geq 830 \wedge$
$x_2 \leq 1006 \wedge x_2 \geq 1003]$
$Subscription\ s_1$
$[x_1 \leq 850 \wedge x_1 \geq 820 \wedge$
$x_2 \leq 1009 \wedge x_2 \geq 1002]$
$Subscription\ s_2$
$[x_1 \leq 870 \wedge x_1 \geq 840 \wedge$
$x_2 \leq 1007 \wedge x_2 \geq 1001]$

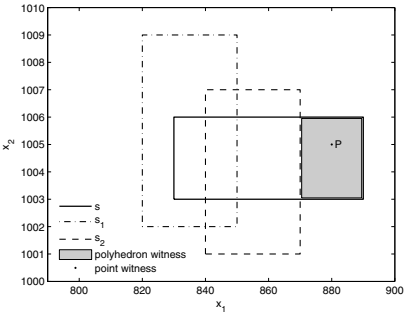

Fig. 2. Non-cover example: graphical presentation of a polyhedron witness and point witness

In the previous example, any point inside the polyhedron witness rectangle defined by $s \wedge \neg s_2^2$ is a point witness. The following 2 corollaries are based on the properties of the conflict table, polyhedron witness and point witness.

Corollary 1. If all T_i^j for $1 \leq j \leq r$ are undefined, then s is covered by s_i.

Proof. If all $T_i^1, \ldots, T_i^r$ are undefined, then $(s \wedge \neg s_i^1, \ldots, s \wedge \neg s_i^r)$ are all not satisfiable, and thus $(s \sqsubseteq s_i^1) \wedge \ldots \wedge (s \sqsubseteq s_i^r)$, or alternatively, $s \sqsubseteq (s_i^1 \wedge \ldots \wedge s_i^r)$. In effect s is covered by s_i. Thus, as a side-effect, the use of the conflict table provides a sufficient condition, tested in $O(m \cdot k)$, to check whether s is covered by any of the subscriptions individually. □

Corollary 2. If all T_i^j for $1 \leq j \leq r$ are defined, then s covers s_i.

Proof sketch. If all $T_i^1, \ldots, T_i^r$ are defined, then $(s \wedge \neg s_i^1, \ldots, s \wedge \neg s_i^r)$ are all satisfiable, and thus s includes s_i on all attributes. □

Corollary 3. Let $t_{i_1}, t_{i_2} \ldots t_{i_k}$ be the list resulting from sorting $t_1, t_2 \ldots t_k$ in ascending order, where t_i represents the number of defined entries in row i of the conflict table T. If all $t_{i_j} \geq j$ for $1 \leq i_j \leq k$, then s is not covered by S.

Proof sketch. If $t_{i_j} \geq j$ for $1 \leq i_j \leq k$, then a polyhedron witness exists. It can be constructed in the following way: Choose any element $s_{i_1}^{j_{i_1}}$ to be part of a polyhedron witness, and then eliminate any conflicting entries from other rows. Since each row will have a maximum of one conflicting element with $s_{i_1}^{j_{i_1}}$, then at most one element in each row will be eliminated. If this step is repeated k times a polyhedron witness will be derived. Thus, s is not covered by S. □

4 Probabilistic Cover Algorithm

In this section we describe the probabilistic cover algorithm to solve the defined subsumption problem. This algorithm has direct implications on the effectiveness of routing both publications and subscriptions in a distributed environment, and the efficiency

to discover matching publications. The probabilistic core of the algorithm is the 'Monte Carlo type' Random Simple Predicates Cover part. It runs in a fixed number of iterations, but may produce an incorrect result with a certain pre-determined probability of error. The probability of error is problem specific, and we show that an upper bound on this error is derived in polynomial time prior to the execution of the algorithm. Thus, the performance of the algorithm can be decided in advance based on particular application requirements. The Random Simple Predicates Cover can be executed independently or in conjunction with the minimized cover set algorithm which reduces the original set of subscriptions S to a non-reducible set against which a new subscription s has to be checked. We also introduce a number of optimizations used for making fast decisions under specific conditions that can be detected from the conflict table.

4.1 Random Simple Predicates Cover

The Random Simple Predicates Cover (RSPC) algorithm exploits the property of *point witnesses*. If the algorithm guesses a point in s that is a *point witness* to non-cover for the set of subscriptions S, then the subsumption problem is solved with a definite NO, i.e. $s \not\sqsubseteq S$. On the other hand, in case a subsumption relationship exists, the algorithm would try in vain to find such a witness. To prevent this situation, we define a threshold d for the number of guesses, and the algorithm may output a probabilistic YES, i.e. $s \sqsubseteq S$ with a predefined probability of error.

Algorithm 1. Random-Simple-Predicates-Cover

```
1: /* Decide whether a subscription s is covered by the existing subscriptions set S */
2: for i = 1 to d do
3:    GUESS a point P inside s
4:    if P does not satisfy subscriptions set S then
5:       RETURN false
6:    end if
7: end for
8: RETURN true
```

Algorithm 1 defines the RSPC algorithm which executes a number of iterations d to randomly generate a point satisfying subscription s and checks whether it is a *point witness*. To generate a point within s costs $O(m)$, and verifying whether it lies inside any of $s_1, s_2, \ldots s_k$ can be done in $O(m \cdot k)$ steps. Overall, the algorithmic complexity of RSPC is $d(m + m \cdot k)$, or $O(d \cdot m \cdot k)$. However, our experiments in Section 6 show that this upper bound is a pessimistic estimate, since at any iteration, RSPC can output a definite NO if the guessed point is indeed a point witness. In addition, the complexity can be greatly reduced using the optimizations presented in Sections 4.2 and 4.3.

Proposition 1. RSPC returns NO when s is definitely not covered by S. It returns YES with a probability error δ upper bounded by

$$\delta = (1 - \rho_w)^d, \tag{1}$$

where ρ_w is the probability that a randomly generated point P inside s is a point witness.

Proof. If RSPC returns NO then a point witness was found, and thus s is definitely not subsumed by S. Therefore, the answer is correct. If s is not subsumed then RSPC returns YES only if none of the guessed points is a point witness. For each trial this happens with probability less than $1 - \rho_w$, therefore for d trials the probability RSPC returns YES is less than $(1 - \rho_w)^d$, since d trials are randomly generated and are thus assumed to be independent. □

In problems with specific probability of error δ, we can compute the necessary number of trials, d, to answer the subsumption question with the required δ, using Equation 1 beforehand in polynomial time. The number of trials increases with a decrease of the error probability. The value of ρ_w depends on the number of existing point witnesses for the particular subscription s related to the set of subscriptions S, and the 'size' (number of integral solutions) of subscription s. Since the probabilistic algorithm may produce a wrong answer only if s is not subsumed by S, the worst situation is to assume that s is indeed not subsumed by the set. To compute the upper bound on d, we need to determine the lower bound on ρ_w, set by the smallest possible polyhedron witness.

Algorithm sketch for computing d. In order to compute d, the algorithm needs the value of ρ_w, which must be approximated, because knowing an exact value is equivalent to solving the subsumption problem. We approximate the lower bound on ρ_w as the product of the minimum distances for each attribute between the new subscription bounds and the bounds of each subscription in the set (possible minimum non-covered ranges). Then, the upper bound on d is extracted from Eq. 1, using the computed value for ρ_w and the given δ.

4.2 Minimized Cover Set of Subscriptions

To further reduce the number of subscriptions against which s needs to be checked, we introduce another algorithm, the minimized cover set algorithm (MCS). From the set of subscriptions S, MCS constructs a non-reducible set of subscriptions, by ignoring those that are redundant for the covering detection problem and filters out duplicate subscriptions (those covering the same parts of s), and subscriptions that do not intersect with s. The remaining subscriptions form the non-reducible set S' (which may not be the minimal covering set) against which s is subsequently checked by RSPC.

Definition 5. Two defined entries in the table, $T_{i_1}^{j_{i_1}}$ and $T_{i_2}^{j_{i_2}}$ are said to be conflicting if $i_1 \neq i_2$, and $s \wedge T_{i_1}^{j_{i_1}} \wedge T_{i_2}^{j_{i_2}}$ is not satisfiable. A defined entry $T_{i_1}^{j_{i_1}}$ is said to be *conflict-free* if it does not conflict with any other defined element $T_{i_2}^{j_{i_2}}$, where $i_1 \neq i_2$.

Conflict free entries are determined by comparing entries from the conflict table related to the same attribute, for different subscriptions. If a constraint conflicts with any other constraint defined by another subscription, the entry is conflicting. It is conflict free otherwise.

Figure 3 visualizes the set of 3 subscriptions, s_1, s_2 and s_3, as well as subscription s defined in Table 6, and Table 7 shows the corresponding conflict table. We can observe

Table 6. Conflict-free example: subscriptions

$Subscription\ s$
$[x_1 \leq 870 \wedge x_1 \geq 830 \wedge$
$x_2 \leq 1006 \wedge x_2 \geq 1003]$
$Subscription\ s_1$
$[x_1 \leq 850 \wedge x_1 \geq 820 \wedge$
$x_2 \leq 1007 \wedge x_2 \geq 1001]$
$Subscription\ s_2$
$[x_1 \leq 880 \wedge x_1 \geq 840 \wedge$
$x_2 \leq 1009 \wedge x_2 \geq 1002]$
$Subscription\ s_3$
$[x_1 \leq 890 \wedge x_1 \geq 810 \wedge$
$x_2 \leq 1005 \wedge x_2 \geq 1004]$

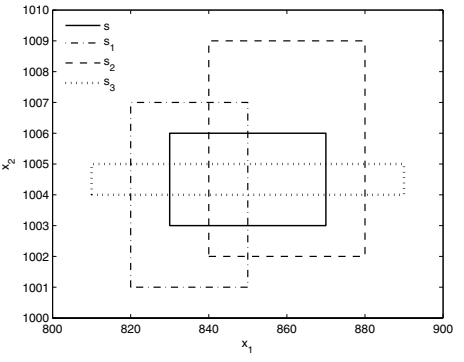

Fig. 3. An example with conflict free entries

Table 7. Conflict table for the example in Figure 3

s_i	$x_1 < low_i^1$	$x_1 > high_i^1$	$x_2 < low_i^2$	$x_2 > high_i^2$
s_1	$undefined$	$x_1 > 850$	$undefined$	$undefined$
s_2	$x_1 < 840$	$undefined$	$undefined$	$undefined$
s_3	$undefined$	$undefined$	$x_2 < 1004$	$x_2 > 1005$

that the defined entries for s_3 are conflict free: they are not conflicting with the entries from s_1 and s_2. On the other hand, s_1 and s_2 have conflicting entries because x_1 cannot simultaneously satisfy both conditions, $x_1 > 850$ and $x_1 < 840$.

Proposition 3. If the number of conflict-free elements in the i-th row of T, f_{c_i}, is greater than or equal to 1, or the number of defined elements in row i, $t_i \geq k$, then s_i is redundant. Proof is given in [15].

The MCS algorithm consists of two main steps, as defined in Algorithm 2. First, starting from the conflict table T, it counts the number of defined elements for each subscription s_i in the corresponding row, t_i and computes the number of conflict free elements, f_{c_i}. Then, it removes from the set all subscriptions for which t_i is equal to or greater than the current number of subscriptions in the set. It also removes subscriptions that have at least one conflict free element in the corresponding row of the conflict table. These two steps are repeated until there are no more subscriptions that fulfill any of the two conditions. The remaining subscriptions form the non-reducible cover set S' for answering the union covering problem.

Considering the conflict table from Table 7, in the first step none of the subscriptions has more defined entries than the total number of subscriptions ($t_1 = t_2 = 1$ and $t_3 = 2$ which is smaller than 3), while only s_3 has conflict free entries. Based on the elimination conditions (in this case, $f_{c_3} = 2 > 0$), MCS can remove subscription s_3 in the first iteration. In the second iteration, still no subscription has more defined entries

Algorithm 2. Minimized Cover Set

```
/* Find the minimized set of subscriptions S' relevant for subsumption detection */
/* Construct and use the conflict table T */
repeat
   S' = S
   for every row i in T do
      compute f_ci /* number of conflict-free elements in row i in T */
      compute t_i /* number of defined entries in row i in T */
      if f_ci ≥ 0 or t_i ≥ k then
         remove row i from T
         remove subscription s_i from S'
         k = k − 1
      end if
   end for
until no s_i can be removed
RETURN S'
```

than the total number of subscriptions ($t_1 = t_2 = 1 < 2$) and there are no conflict free entries, thus the algorithm stops. The minimized cover set is $S' = \{s_1, s_2\}$.

Determining if a table entry is conflict free is $O(m \cdot k)$. Therefore computing each f_{c_i} costs $O(m^2\, k)$, and in turn steps 1 and 2 in each iteration of the MCS algorithm cost $O(m^2\, k^2)$. Steps 1 and 2 may be repeated k times since each time step 2 is performed at least one s_i is filtered out. As a result, the overall cost of the algorithm reduction is $O(m^2\, k^3)$ in the worst case.

4.3 Fast Decisions Based on Sufficient Conditions

To summarize, in order to answer the subsumption problem, the algorithm first constructs the conflict table, runs the MCS algorithm to reduce the subscription set, and then applies the probabilistic RSPC algorithm which produces either a definite NO or a probabilistic YES. Nevertheless, for some specific cases, the algorithm can efficiently give a deterministic answer. Here we briefly present three specific cases.

1. Pairwise subsumption: As stated in Corollary 1, it is possible to detect if a subscription s is entirely covered by another subscription and produce a definite YES by analyzing the conflict table. If the row in the conflict table corresponding to subscription s_i contains only *undefined* values, then s_i covers the new subscription.
2. The outcome of the MCS algorithm can be an empty set, which means that there are no candidate subscriptions that could jointly cover s, and the algorithm will produce a definite NO.
3. Polyhedron witness: Detecting the existence of a polyhedron witness suffices to detect a non-cover relationship and output a definite NO as stated in Corollary 3. Based on the definitions of the polyhedron witness and conflict free entries, we can detect the presence of such a witness, depending on the number of defined entries in the conflict table without using either RSPC or MCS.

5 Subscription Propagation in a Distributed System

As in a distributed system subscription propagation affects the overall system performance, here we analyze the implications of incorrectly declaring a subscription as covered. Equation 1 gives the upper bound for the probability of error in incorrectly withholding the forwarding of a subscription, and therefore, it represents the likelihood of not finding a matching publication if it is available at the next broker. In a distributed publish/subscribe system, data is routed throughout the system, and we need to analyze the influence of our probabilistic algorithm on subscription propagation. We consider in Figure 4 a simple and illustrative case, where the new subscription s should be propagated along a chain of brokers $B_1, B_2, \ldots, B_n$.

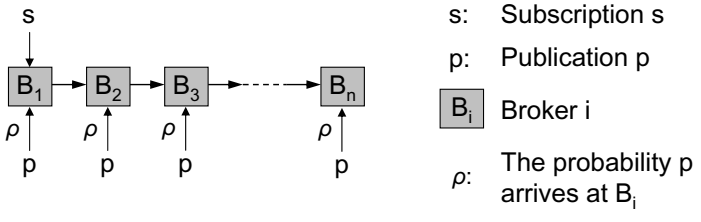

Fig. 4. New subscription propagation

We assume that the new subscription s is issued at broker B_1, while subscriptions $s_1, s_2, \ldots, s_k$ have already been propagated down the path to all brokers. Let ρ be the probability that a matching publication p (matches s but no s_i) is issued at any of the brokers B_i. The overall performance of the probabilistic algorithm is given by the probability of finding the matching publication, wherever it resides.

Proposition 4. The probability of finding the matching publication p under the condition that s is erroneously found to be covered by S, where $s_1, s_2, \ldots, s_k$ have been propagated to all brokers along the path, and all brokers have equal probability of ρ of receiving publication p is:

$$\sum_{i=1}^{n} \rho[(1-\rho)(1-(1-\rho_w)^d)]^{i-1}, \tag{2}$$

where ρ is determined by the network density and the communication distance of two neighboring brokers, and n is the total number of brokers in the path.

Equation 2 gives the lower bound for the overall algorithm performance. However, as we will show in the next chapter, the actual performance is much better in practice, even for loose error probabilities. On the other hand, the reduction in the global subscription traffic is more important for longer broker paths, reflecting the local reduction at each broker, exponentially amplified in the network diameter.

Note that we do not present in this paper the mechanism for dealing with subscriptions cancelation. This issue can be tackled by explicit forwarding of unsubscriptions

between brokers or by associating an expiration time with each new subscription. According to our approach, the canceled subscription can either be covered, and then cancelation has only the effect of removing it from the passive set, either be present in the (active) subscription set, and then its covered subscription must be promoted to this set, to replace it.

6 Experimental Evaluation

In this section, we evaluate the performance of the proposed probabilistic approach in terms of *efficiency* and *effectiveness* using a number of subscription generation scenarios. Efficiency is measured as the number of actual algorithmic steps performed to answer the subsumption question, and effectiveness as the ratio of recognized redundant subscriptions to the total number of redundant subscriptions. Especially, we are interested in potential gains and costs when using the MCS algorithm in specific subscription generation scenarios. Next, we analyze the *number of false decisions* declaring a subsumption relationship when there was no subsumption. Finally, we compare our approach with the existing one for pair-wise coverage detection.

There are two main categories of subscription settings:

(1) Covering: s is covered by the set of subscriptions (with some of $s_i \in S$ being redundant).
(2) Non-cover: s is not covered by the set S (as such, all subscriptions are redundant).

In particular, we have considered the following subscription generation scenarios:

(1.a) Pairwise covering scenario; s is entirely covered by at least one subscription from the set of existing subscriptions.
(1.b) Redundant covering scenario; s is not covered by any single subscription, but is covered by the set, with a lot of subscriptions being redundant.
(2.a) No intersection scenario; s does not intersect with any existing subscription.
(2.b) Non-cover scenario; s is not covered by the set S, but overlaps with existing subscriptions over many attributes.
(2.c) Extreme non-cover scenario; similar to (2.b), but s has only a very small non covered gap.
(1-2) Comparison scenario; generate incoming subscriptions randomly.

Scenario (1.a) is straightforward as the covering relationship is determined efficiently by applying Corollary 1 after the construction of the conflict table, therefore the cost of detecting pairwise coverage is $O(m \cdot k)$. Scenario (2.a) is also straightforward because MCS determines non subsumption after the first iteration by removing all subscriptions from the set S' because all $s_i \in S$ have conflict-free elements in the conflict table. Scenarios (1.b), (2.b), and (2.c) are difficult settings for checking the covering relationship, as there are no pair-wise subsumptions which could help to reduce the set S'. We investigated these scenarios using the following subscription generation principle: Existing subscriptions overlap with a new subscription and each other for many attributes, but there are no pair-wise subsumptions. The last scenario (1-2) simulates a realistic setting

Table 8. Parameters used in simulations

	(1.b)	(2.b)	(2.c)	(1-2)
no. of subscriptions k	10-310 (30)	10-310 (30)	50	1 to 5000
no. of attributes m	10, 15, 20	10, 15, 20	5	10, 15, 20
error δ	10^{-10}	10^{-10}	10^{-3}, 10^{-6}, 10^{-10}	10^{-6}
no. of trial runs	1000	1000	3000	1
gap size	0	random	0.5-4.5% (0.5%)	NA

assuming that user interests are similar, and that the popularity of attributes appearing in subscriptions is Zipfian.

The parameters used in simulations are listed in Table 8. For the redundant covering and non-cover scenarios, the setting is similar, while the extreme non-cover scenario investigates different error probabilities. The comparison scenario is performed in a single run by generating a sequence of subscriptions. In the figures, each plotted point is the average of the values obtained over the number of trial runs.

6.1 Redundant Covering Scenario

This simulation scenario investigates algorithm performance when the subscription set S subsumes s. A high rate of redundant subscriptions is introduced to test the influence of the MCS algorithm on the overall performance: s is covered by 20% of the generated subscriptions and the remaining 80% are redundant and partly cover s.

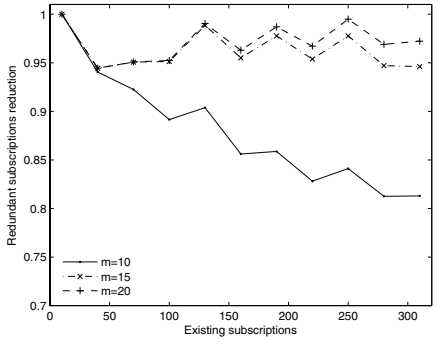

Fig. 5. Reduction for the redundant covering scenario

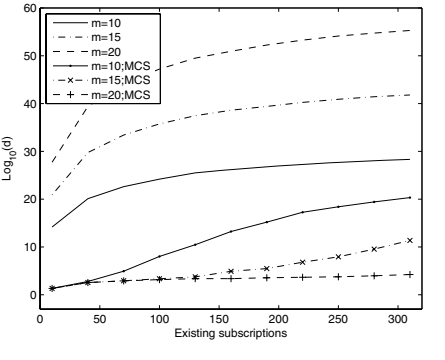

Fig. 6. Theoretical number of iterations for the redundant covering scenario

Figure 5 shows the effectiveness of the MCS algorithm: It successfully removed between 80% and 100% of redundant subscriptions. The performance increases for higher number of attributes because when increasing m, the probability of group coverage increases due to the specific subscription generation scenario.

Figure 6 shows the theoretically predicted number of iterations d needed to answer the subsumption question. The $log(d)$ plot is shown as a function of k, calculated using Equation 1. The plot is given for the initial set of subscriptions S, and the reduced set S' after running MCS. Due to the tight error probability, d is extremely high when using only the RSPC algorithm. However, MCS significantly reduces the number of needed iterations and becomes practically feasible: $d < 10^5$ for 100 subscriptions with 10 attributes, and decreases significantly for larger number of attributes. Further more, as the results obtained for non-cover show, we could reduce the number of trials further.

6.2 Non-cover Scenario

For the non-cover scenario, the experiment is constructed by forcing the non-covering of s by leaving a small range over x_1 uncovered. The values over the other attributes are generated randomly. The whole set of subscriptions S is actually redundant as s is not covered. In this scenario, the algorithm has always detected the non-coverage relationship due to optimizations and a low probability of error.

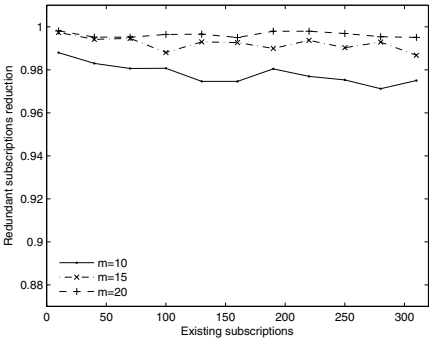

Fig. 7. Reduction for the non cover scenario

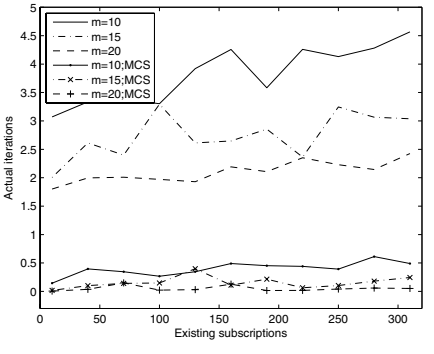

Fig. 8. Actual iterations for non cover

Figure 7 shows the effectiveness of the MCS algorithm which performs even better than for the redundant covering scenario because most of the subscriptions are removed quickly due to the non covering relationship.

Since non-cover can be detected prior to performing all d theoretical iterations, Figure 8 shows the actual number of iterations performed to discover a witness point. The average number of performed iterations is extremely low (< 0.5), due to the fact that in most of the cases, after running MCS, the reduced set is empty, thus $d = 0$. There are some evident fluctuations for this scenario caused by the probabilistic nature of the algorithm.

6.3 Extreme Non-cover Scenario

The extremeness of this scenario consists of covering the new subscription entirely, except for a narrow slice over one attribute, where we enforce a gap. We investigate the

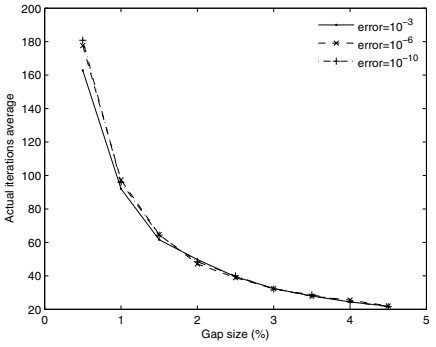

Fig. 9. Actual iterations performed

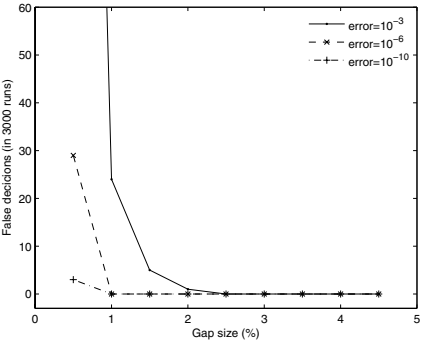

Fig. 10. Number of false negatives

total number of false decisions that result in non forwarding of a non covered subscription and the average number of performed trials.

Figure 9 shows that the average number of guesses is similar for all probabilities of error, even though the theoretical number of guesses increases for tighter error probabilities. This behavior is expected, as the chances of guessing a point witness depend on the ratio between the gap size and the total range size of the non covered attribute, but it does not depend on the error probability. This result suggests that we can also reduce the number of trials after which a subscription is declared covered.

In Figure 10 we can see the total number of false decisions increases with the error probability and decreases with larger gap sizes. In fact, for probabilities of error lower than 10^{-6} and gap sizes of more than 1%, the algorithm always takes the right decision. Even for a looser probability of error (10^{-3}), the number of false negatives remains quite low, if the gap is at least 2%. This shows that an error probability of 10^{-6} is sufficient for detecting non-coverage in most application scenarios because it has a low number of false decisions in case of a small non-covered subscription space while at the same time reducing the theoretical number of iterations d.

6.4 Comparison

Due to the lack of real-world subscription set, we have simulated a setting using power law distributions that are considered as good approximations of popularity both for the selection of attributes and attribute ranges. From the set of m attributes popular ones were chosen using a Zipf distribution (skew = 2.0). Attributes are generated in the following way: The center of a range is generated with a Pareto distribution (skew = 1.0) to simulate similar interests, while range sizes are generated with a normal distribution.

The experiment compares the growth of subscription set sizes in case of the pair-wise ([7,8]) and group subsumption (our approach) reductions.

Figure 11 shows the growth of the total number of active subscriptions when increasing the number of incoming subscriptions. It is interesting to observe the power of subscription set reduction using subscription coverage both for pair-wise and group

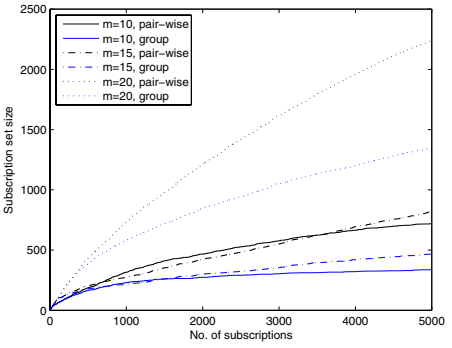

Fig. 11. Evolution of subscription set size

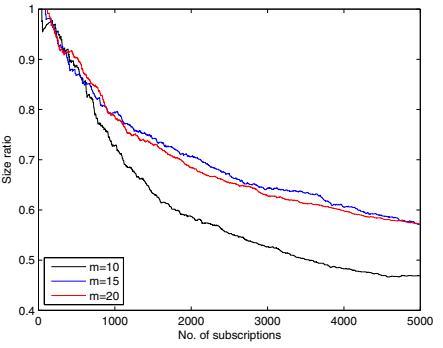

Fig. 12. Ratio of subscription set sizes

coverage in case of partly covered subscription space. The group coverage shows greater reduction compared to the pair-wise algorithm for all values of m. For $m = 10$ and $m = 15$ group coverage has reduced the original set of 5000 subscriptions to less than 10%, and pair-wise coverage to approx. 15% of the entire set, while for $m = 20$ the reduction is still significant (around 33% for group and less than 50% for pair-wise coverage). The set reduction is very important for subscriptions with a large number of attributes which increases complexity because of the absolute subscription set size, e.g. some brokers have limited resources and may not handle more than 1000 active subscriptions. When increasing m, the actual number of active subscriptions is also larger, and this is due to the fact that the probability of subsumption generally decreases in the applied subscription generation scenario when increasing subscription space dimensionality.

Figure 12 quantifies the actual gain of group coverage compared to the pair-wise coverage by showing the ratio between the subscription set sizes obtained with the 2 reduction mechanisms . The obtained results show the extreme reduction potential when increasing the number of incoming subscriptions. In case of 1000 received subscriptions, the ratio is between 70 and 80%, and keeps decreasing with new incoming subscriptions showing a stabilization tendency after 5000 subscriptions. The ratio is larger for large m, but still significant, and is almost similar for 15 and 20 attributes because the actual number of defined attributes does not significantly differ. Of course, the obtained results are highly dependent on subscription generation, but since our distributions follow a realistic popularity-based setting, it can be concluded that group coverage can greatly reduce the subscription set compared to the pair-wise approach.

To conclude, the reduction algorithm is both efficient and effective: It can significantly reduce the size of the subscription set with acceptable error probability and computational costs. RSPC should be used in combination with MCS because it dramatically reduced the number of performed trials. Finally, the comparison shows the supremacy of the group coverage algorithm over the classical pair-wise approach that will in general largely decrease the number of subscriptions in different distributed publish/subscribe systems.

7 Related Work

Most of the research efforts in publish/subscribe systems have so far focused on the problem of efficient matching and forwarding of publications [9,7]. Pairwise covering and merging of subscriptions are typically used to reduce the set of active subscriptions, and all algorithms rely on some version of the *counting algorithm*, originally defined in [16]. The importance of reducing the number of subscriptions in a distributed environment is stressed in [8]. The authors are dealing with a complementary problem—merging a set of subscriptions to reduce their number. In [7], modified binary decision diagrams are employed, to achieve pairwise covering and merging of subscriptions. The trade-off with merging is that the new subscription might contain parts of the subscription space not covered originally by the set, which leads to false positives, delivery of unrequested publications. A recently proposed solution relies on clustering of subscriptions based on a proximity metric in subscription space [17], and would greatly benefit from global subscription set reduction for both the total number of subscriptions and the generated traffic. None of these techniques supports group subsumption, and can filter out fewer subscriptions than the proposed probabilistic algorithm.

8 Conclusion

The paper presents a novel probabilistic algorithm for determining whether a subscription is covered by a set of subscriptions. Theoretically it solves the problem in $O(k \cdot m \cdot d)$. The probability of error is problem specific and very small, and an upper bound on the threshold d is determined in polynomial time prior to the execution of the algorithm. Our experiments have shown that the algorithm performs much better in practice when combing the probabilistic algorithm with the reduction algorithm that removes redundant subscriptions against which a new subscription needs to be checked. Even more, in case of the non covering relationship, it is possible to give a deterministic answer without applying the probabilistic tests. Therefore, we can conclude that the proposed algorithms can efficiently decide whether a subscription is covered by a group of subscriptions which is important for fast subscription forwarding and network congestion control in distributed publish/subscribe systems. The experimental results show that the algorithm performs much better than the pessimistic theoretical bounds even for settings where group coverage is difficult to detect. Finally, compared to the reduction based on the classical pair-wise coverage, the subscription set reduction achieved with our approach is significantly better, which, correlated with its good efficiency and the very tight achievable error probabilities, recommends it for distributed publish/subscribe systems.

References

1. Aguilera, M.K., Strom, R.E., Sturman, D.C., Astley, M., Chandra, T.D.: Matching events in a content-based subscription system. In: Proceedings of the 18th ACM Symposium on Principles of Distributed Computing, ACM Press (1999) 53–61
2. Altinel, M., Franklin, M.J.: Efficient filtering of XML documents for selective dissemination of information. In: The VLDB Journal. (2000) 53–64

3. Campailla, A., Chaki, S., Clarke, E., Jha, S., Veith, H.: Efficient filtering in publish-subscribe systems using binary decision diagrams. In: Proceedings of the 23rd International Conference on Software Engineering. (2001) 443–52
4. Fabret, F., Jacobsen, A., Llirbat, F., Pereira, J., Ross, K., Shasha, D.: Filtering algorithms and implementation for very fast publish/subscribe. In Sellis, T., Mehrotra, S., eds.: Proceedings of the 20th Intl. Conference on Management of Data (SIGMOD 2001), Santa Barbara, CA, USA (2001) 115–126
5. Carzaniga, A., Rosenblum, D.S., Wolf, A.L.: Design and evaluation of a wide-area event notification service. ACM Transactions on Computer Systems **19** (2001) 332–383
6. Mühl, G., L.Fiege, Gärtner, F.C., Buchmann, A.: Evaluating advanced routing algorithms for content-based publish/subscribe systems. In: Proceedings of the 10th IEEE International Symp. on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS'02), IEEE Computer Society (2002) 167–176
7. Li, G., Huo, S., Jacobsen, H.: A unified approach to routing, covering and merging in publish/subscribe systems based on modified binary decision diagrams. In: 25th International Conference on Distributed Computing Systems (ICDCS). (2005)
8. Crespo, A., Buyukkokten, O., Garcia-Molina, H.: Query merging: Improving query subscription processing in a multicast environment. IEEE Transactions on Knowledge and Data Engineering **15** (2003)
9. Carzaniga, A., Wolf, A.L.: Forwarding in a content-based network. In: Proceedings of ACM SIGCOMM 2003, Karlsruhe, Germany (2003) 163–174
10. Srivastava, D.: Subsumption and indexing in constraint query languages with linear arithmetic constraints. Annals of Mathematics and Artificial Intelligence **8** (1992) 315–343
11. Liu, H., Jacobsen, H.A.: Modeling uncertainties in publish/subscribe systems. In: ICDE '04: Proceedings of the 20th International Conference on Data Engineering, Washington, DC, USA, IEEE Computer Society (2004) 510
12. Chen, Y., Schwan, K.: Opportunistic overlays: Efficient content delivery in mobile ad hoc networks. In: Middleware 2005, ACM/IFIP/USENIX, 6th International Middleware Conference. (2005) 354–374
13. Dittrich, J.P., Fischer, P.M., Kossmann, D.: Agile: Adaptive indexing for context-aware information filters. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, Baltimore, Maryland, USA, June 14-16, 2005. (2005) 215–226
14. Hauswirth, M., Schmidt, R.: An overlay network for resource discovery in Grids. In: 2nd International Workshop on Grid and Peer-to-Peer Computing Impacts on Large Scale Heterogeneous Distributed Database Systems (GLOBE'05). (2005)
15. Ouksel, A.M., Jurca, O., Podnar, I., Aberer, K.: Fast probabilistic subsumption checking for publish/subscribe systems. Technical Report LSIR-REPORT-2006-004, EPFL (2006)
16. Yan, T.W., García-Molina, H.: Index structures for selective dissemination of information under the Boolean model. ACM Transactions on Database Systems **19** (1994) 332–334
17. Voulgaris, S., Rivire, E., Kermarrec, A.M., van Steen, M.: Sub-2-sub: Self-organizing content-based publish and subscribe for dynamic and large scale collaborative networks. In: 5th Int'l Workshop on Peer-to-Peer Systems (IPTPS). (2005)