MICS: An Efficient Content Space Representation Model for Publish/Subscribe Systems

Hojjat Jafarpour, Sharad Mehrotra, Nalini Venkatasubramanian
Department of Computer Science
University of California
Irvine, CA 92697
{hjafarpo,sharad,nalini}@ics.uci.edu

Mirko Montanari
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801
mmontan2@illinois.edu

ABSTRACT

One of the main challenges faced by content-based publish/subscribe systems is handling large amount of dynamic subscriptions and publications in a multidimensional content space. To reduce subscription forwarding load and speed up content matching, subscription covering, subsumption and merging techniques have been proposed. In this paper we propose MICS, Multidimensional Indexing for Content Space that provides an efficient representation and processing model for large number of subscriptions and publications. MICS creates a one dimensional representation for publications and subscriptions using Hilbert space filling curve. Based on this representation, we propose novel content matching and subscription management (covering, subsumption and merging) algorithms. Our experimental evaluation indicates that the proposed approach significantly speeds up subscription management operations compared to the naive linear approach.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Design Studies; Modeling Techniques; I.6.3 [**Simulation and Modeling**]: Applications; I.6.4 [**Simulation and Modeling**]: Model Validation and Analysis

General Terms

Algorithms, Experimentation, Performance

Keywords

Event notification, publish/subscribe

1. INTRODUCTION

Content-based Publish/Subscribe (pub/sub) is a customized many-to-many communication model that can satisfy requirements of many modern distributed applications [1]. In a pub/sub scheme, subscribers express their interest in content. Whenever a content is produced, it is delivered to the interested subscribers . By decoupling communication parties, a pub/sub system provides anonymous and asynchronous communication making it an attractive communication infrastructure for many applications. Such applications include selective information dissemination, location-based services, sensor networks and workload management [1].

In order to distribute the load of publications and subscriptions a distributed content-based pub/sub system uses a set of network brokers. Routing protocols for publications and subscriptions in brokers aim to reduce network cost that results from publication and subscription forwarding. Subscriptions typically are broadcast to all brokers. This reduces network traffic by enabling filtering of publications close to their sources. To reduce subscription forwarding traffic, subscription covering, subsumption and merging techniques are used to prevent propagation of redundant subscriptions. It has been shown that exploiting subscription covering, subsumption and merging results in significant reduction in subscription traffic [2, 4–7]. Li *et al.* show that depending on subscription set, covering and merging can result in up to 75% reduction in subscription traffic [6]. Ouksel *et al.* also show that subscription subsumption, where a set of existing subscriptions cover a new subscripition, has a significant effect in reducing subscription traffic [4]. However, existing algorithms for evaluating covering, subsumption and merging have at least linear execution time with respect to the number of subscriptions. Also, most of the existing systems including Siena, PADRES and REBECA only exploit pairwise subscription comparison or merging and it is not clear how to extend them to exploit subsumption relation among subscriptions [2, 6, 7].

In this paper we propose, MICS, an efficient approach to evaluate subscription covering, subsumption and merging and perform publication matching. In MICS, subscriptions are mapped to a single dimensional space using Hilbert space filling curve and are represented using a set of ranges. Based on this representation, we organize subscriptions in a $B^{+}Tree$ and propose a novel algorithm that efficiently detects if a subscription is covered or subsumed by existing subscriptions or if it can be merged with a set of existing subscriptions. We also propose an efficient publication matching algorithm. The basic MICS approach is described for lower dimensional spaces. To scale to higher dimensional spaces, we propose a hybrid approach based on combining

MICS and previously proposed approaches that significantly reduces load of subscription management.

The paper is organized as follows. In the next section we provide a brief overview of distributed content-based pub/sub model that we assume in this paper. Section 3 describes MICS, our content space representation model. In Section 4 we present our novel subscription subsumption and merging and content matching algorithms. Our hybrid approach for content space with higher dimensions is presented in Section 5. Section 6 discusses issues in the proposed model and addressing them in MICS. Section 7 presents our experimental results. Section 8 describes related work and Section 9 concludes the paper.

2. DISTRIBUTED PUBLISH/SUBSCRIBE SYSTEM

To keep the paper self-contained, we provide a brief overview of distributed content-based pub/sub system that we assume in this paper. Different architectures for distributed content-based pub/sub have been proposed in the literature [2, 3, 19, 20]. Our proposed system is based on the architecture used in Siena, PADRES and REBECA [2, 6, 7].

The pub/sub system consists of a set of brokers interconnected through transport-level links which form an *acyclic* overlay network. Each client is connected to one of the brokers. When a client issues a subscription, it sends it to the broker that it is connected to. The broker acts as a subscriber with respect to the rest of the brokers and propagates the subscription to all other brokers. When a client publishes an event, it sends the event to its corresponding broker. This broker also behaves as publisher with respect to other brokers and forwards the event through the broker overlay network toward brokers that have subscribed for the event. Then each of these brokers delivers the event to their clients that actually subscribed for the event. Figure 1 depicts a sample broker overlay network with 11 brokers and clients connected to one of these brokers.

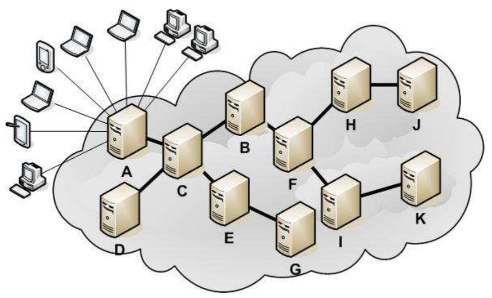


Figure 1: A sample broker overlay network.

Subscriptions are broadcast to all brokers in the overlay network and brokers store them in their subscription table. Upon receiving a publication from a neighbor broker or one of its clients, the broker matches it to the subscriptions in the subscription table and forwards the publication to a neighbor broker if and only if it has received a matching subscription from that neighbor. Since the content matching operation is performed in each broker along the path from publisher broker to subscribers, matching time has a significant effect on the speed of publication dissemination. Efficient matching techniques have been proposed in the literature to reduce matching time [10, 13].

To prevent unnecessary dissemination of subscriptions and reduce the size of subscription tables, pub/sub systems use subscription covering, subsumption and merging techniques. A subscription s_1 covers subscription s_2 if and only if all publications matching s_2 also match s_1. When a broker, N_i receives a new subscription from one of its neighbors, B_j, that is covered by a subscription previously received from B_j, since it does not affect publication routing, B_i does not forward the new subscription to other neighbors. Since the covered subscriptions are not disseminated to all brokers, it results in lower network traffic and compact subscription tables. Note that the receiver of the covered subscription stores the subscription in its *passive subscription list* since it may forward this subscription later should the covering subscription be canceled by an unsubscription request from its subscriber. In this case, the covering subscription is removed from the corresponding subscription table and the covered subscription(s) in the *passive subscription list* are moved to the subscription table and forwarded to the neighbors along with the unsubscription request for the covering subscription.

The more efficient utilization of covering is *subscription subsumption.* While a new subscription may not be covered by a single existing subscription, it is possible that it is covered by a set of existing subscriptions. In this case also forwarding the new subscription does not have any effect on publication routing and is redundant. In fact subscription covering process is a subset of subscription subsumption process and if a subscription is covered by an existing subscription, the subsumption process can detect it. However, subscription subsumption in general has been proved to be a co-NP complete [14].

The other technique that is used to further minimize subscription table size in brokers is subscription merging [6]. The goal of subscription merging is to replace a set of subscriptions with one subscription that have the same effect on publication routing. This results in reduced subscription table size and speeds up matching operation in brokers. In fact, the result of merging a set of subscriptions is a subscription that represents the union of these subscriptions.

In general, the goals of these subscription management operations are to 1) Reduce subscription dissemination traffic. 2) Reduce subscription table size that results in reducing subscription table memory consumption and speeding up the content matching operation.

3. MICS: CONTENT SPACE REPRESENTATION MODEL

In this section we present the basic MICS approach to represent content space in a pub/sub system. We begin by first introducing the notation that we use.

3.1 Notations

We assume the content space consists of d attributes, $A = \{a_1, a_2, ..., a_d\}$, that form a d-dimensional space. The maximum and minimum boundaries of each attribute in the system is known. We represent the domain of attribute a_i with $[l_i, u_i]$ where l_i is the lower bound of the attribute domain and u_i is the upper bound of the attribute domain. Note that $l_i \geq -\infty$ and $u_i \leq \infty$. Each publication represents a point in the content space that is represented as $p = (v_1, v_2, ..., v_d)$ where v_i is the value of attribute a_i in the publication and $v_i \in [l_i, u_i]$. A subscription s is represented as a *conjunction* of predicates where each predicate represents the subset of the corresponding attribute domain that the subscriber is interested in. We assume that each predicate in subscription s_j is represented as $[low_i^j, up_i^j]$ that indicates the boundaries of the subscription for ith attribute. Thus, a subscription corresponds to a *d-dimensional rectangle* in the content space. Note that $l_i \leq low_i^j$ and $up_i^j \leq u_i$.

We say that publication p matches(satisfies) subscription s_j if and only if for each v_i in p, $v_i \in [low_i^j, up_i^j]$.

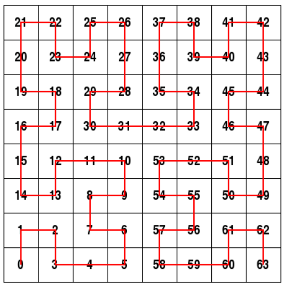


Figure 2: $\mathcal{H}_3^2$, a 2 dimensional space partitioning and indexing using Hilbert curve.

3.2 Multidimensional indexing using Hilbert curve

To be able to efficiently support subscription management, we map the multi dimensional content space to a single dimensional representation using Hilbert curve [16]. The first step is to partition the multi dimensional space. Each dimension domain is divided into 2^k intervals and these intervals are numbered from 0 to $2^k - 1$. We represent ith interval in j'th domain as $\mathcal{I}_{i_j}$. The result of this partitioning is $(2^k)^d$ d-dimensional cubes that we refer to as *cells*. We use $C(\mathcal{I}_{i_1}, ..., \mathcal{I}_{i_d})$ to represent the cell composed by i_jth interval in jth dimension in the space. The partitioned d-dimensional space is represented by $\mathcal{H}_k^d$. The next step is to index these cells in the space. Sophisticated indexing functions have been proposed in the literature including functions based on z-curve, Gray-coded curve and Hilbert curve [16]. Since Hilbert curve preserves locality better than other space filling curves [17], we use *Hilbert curve* for mapping the content space.

Hilbert curve. A d-dimensional Hilbert space filling curve in $\mathcal{H}_k^d$ is a one to one mapping from $[0, 2^k - 1]^d$ into $[0, 2^{kd} - 1]$ that assigns an integer to each cell in the partitioned space and is defined as follows.

$$\mathcal{H}ilbert : [0, 2^k - 1]^d \rightarrow [0, 2^{kd} - 1]$$

The index of cell $C(\mathcal{I}_{i_1}, ..., \mathcal{I}_{i_d})$ that has been produced using Hilbert space filling curve is represented by $\mathcal{H}ilbert(\mathcal{I}_{i_1}, ..., \mathcal{I}_{i_d})$ that is an integer in $[0, 2^{kd} - 1]$. Figure 2 depicts partitioning and indexing of a 2-dimensional space using Hilbert curve.

3.3 Publication/Subscription representation in MICS

Publication Representation: As discussed above, every publication is a point in the content space that is represented as $p = (v_1, v_2, ..., v_d)$. In the mapping of a publication into single dimensional space, we represent the publication by the index of the cell that the publication falls in. Formally, $p = (v_1, v_2, ..., v_d)$ is represented by $\mathcal{H}ilbert(\mathcal{I}_{i_1}, ..., \mathcal{I}_{i_d})$ where $v_j \in \mathcal{I}_{i_j}$ for $j \in \{1, ..., d\}$.

Therefore, the vector representation of a publication is reduced to just an index number. Figure 3 shows three sample publications, p_1, p_2 and p_3, in the two dimensional space, $\mathcal{H}_3^2$, that are represented as 28, 52 and 50 respectively.

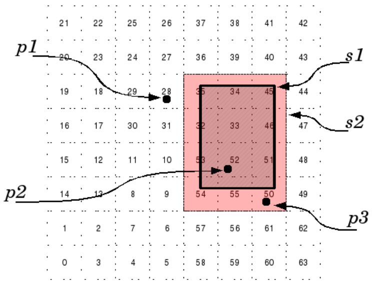


Figure 3: Sample publications and subscription in a two dimensional space.

Subscription Representation: To represent a subscription in the partitioned space, the first step is to augment the subscription rectangle to a rectangle that is composed of cells in the partitioned space. The resulting rectangle from the subscription augmentation step is the minimum bounding rectangle of the original subscription. This minimum bounding rectangle consists of all cells in the partitioned space that have intersection with the subscription. Subscription s_2 is the augmentation of subscription s_1 in figure 3. The formal representation of minimum bounding rectangle for a subscription s in the event space $\mathcal{H}_k^d$ is as follows.

$$\mathcal{H}ilbert(s) = \{C(\mathcal{I}_{i_1}, ..., \mathcal{I}_{i_d}) \in \mathcal{H}_k^d \text{ where } C(\mathcal{I}_{i_1}, ..., \mathcal{I}_{i_d}) \cap s \neq \varnothing\}$$

This augmentation covers some parts of the content space that are not requested by the subscription and may result in false positives in publication dissemination. The amount of resulting false positives directly is related to the space partitioning granularity. The finer granules and smaller partitions result in fewer false positive.

After finding the indexes for the cells in the augmented subscription, the next step is to form the intervals representing these indexes. In this step augmented subscription indexes are clustered based on their sequence and represented by a set of intervals (ranges) based on indexing the space by Hilbert curve. Therefore, the result of mapping a subscription into the one dimensional space is a set of discrete ranges. We represent the number of ranges for subscription s by $\mathcal{N}_s$ and the set of these ranges by $s = \{\mathcal{R}_1, .., \mathcal{R}_{\mathcal{N}_s}\}$. If the mapping process results in large number of intervals, subscription representation in one dimensional space may not be efficient. We discuss this in section 5. The number of resulting ranges for a subscription depends on the number of dimensions and the indexing technique that is used for space partitioning. It has been shown that the indexing technique based on the Hilbert space filling curve results in the least number of intervals in most circumstances [17]. Moon *et. al.*, have shown that in a d-dimensional space $\mathcal{H}_k^d$, the average number of intervals for a rectangle is evaluated by the following formula [16].

$$\lim_{k\to\infty} \mathcal{N}_s = \frac{S}{2d} \quad (1)$$

In this formula S represents the total surface area of the augmented subscription. Since subscriptions are rectangles in our system and assuming that a rectangle for subscription s_j is represented by $\prod_{i=1}^{d}[low_i^j, up_i^j]$ the surface of the subscription rectangle is calculated as follow.

$$S = 2\sum_{k=1}^{d}\left(\frac{1}{[up_k^j - low_k^j]}\prod_{l=1}^{d}[up_i^j - low_i^j]\right) \quad (2)$$

Therefore, the average number of intervals for such query is as follow.

$$\lim_{k\to\infty} \mathcal{N}_s = \frac{1}{d}\sum_{k=1}^{d}\left(\frac{1}{[up_k^j - low_k^j]}\prod_{l=1}^{d}[up_i^j - low_i^j]\right) \quad (3)$$

As an example of subscription representation consider the subscription s_1 in figure 3. The augmented version of this subscription in the sample two dimensional space is s_2. The set of cells that form the augmented subscription is {32, 33, 34, 35, 45, 46, 50, 51, 52, 53, 54, 55}. This results in three intervals for representing the subscription that are {[32,35], [45,46], [50,55]}.

3.4 Content matching in MICS

In the new one dimensional space, we say publication p matches subscription s if and only if the Hilbert index of the cell for the publication is included in one of the intervals representing s. Formally, in the new representation of the content space, matching is defined as follow.

p matches s iff $\exists \mathcal{R}_i \in s$ such that $\mathcal{H}ilbert(p) \in \mathcal{R}_i$

Based on the new matching definition, the problem of content matching converts to finding if a point is covered by at least one of ranges in a set of ranges representing subscriptions. As an example consider matching of p_1=28, p_2=52 and p_3=50 as three publications with subscription s_1 in figure 3. Since we use the augmented subscription to represent a subscription in our representation model, the content matching process uses s_2 to perform matching operation. Therefore, 28 $\notin$ {[32,35], [45,46], [50,55]} means that p_1 does not match s_2. On the other hand, since 52, 50 $\in$ {[32,35], [45,46], [50,55]}, p_2 and p_3 match s_2. Note that p_3 does not match s_1 and is a false positive.

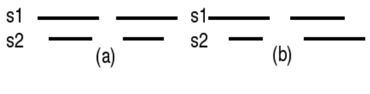


Figure 4: Covering relation between subscription ranges.

3.5 Subscription covering in MICS

To reduce the subscription dissemination traffic and condense the subscription information stored in brokers and speed up matching operation, brokers exploit covering relation between subscriptions. The MICS representation of the content space simplifies the use of covering relation between subscriptions. Based on the one dimensional representation of subscriptions, the covering relation is defined as follows. Subscription s_1 covers subscription s_2 if and only if for all range $\mathcal{R}_i \in s_2$, there exists a range $\mathcal{R}_j \in s_1$ such that $\mathcal{R}_i \subseteq \mathcal{R}_j$. Since in our representation subscriptions are ranges, the above definition in fact shows that all ranges in s_2 are covered by ranges in s_1.

Partial Covering: In the existing covering algorithms for content-based pub/sub a subscription can either cover another subscription or can be covered by another subscription. We call this relation *total covering* relation. Our proposed scheme enables us to define the novel concept of *partial covering*. In partial covering relation between two subscriptions, if even just one of the ranges of a subscription, which represents one part of it, covers one of the ranges of the other subscription, we can remove the covered range from the subscription table. The partial covering relation enables our approach to exploit overlap between subscriptions to reduce required storage size for subscriptions in brokers. Figure 4 shows the difference between partial covering and total covering relations between two subscriptions. As it can be seen in figure 4.a, all ranges of s_2 are totally covered by ranges of s_1. However, in figure 4.b one range of s_1 covers one range of s_2 while the other range of s_2 is not covered by the other range of s_1.

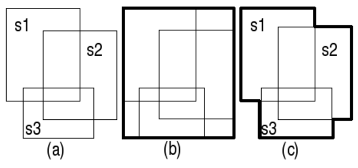


Figure 5: Imperfect and perfect merging of three subscriptions.

3.6 Subscription merging in MICS

Merging subscriptions is a technique to reduce subscription table size in brokers. Subscription merging in multidimensional space is not a trivial process. There are different techniques for merging subscriptions in a multidimensional space [15]. Figure 5 depicts samples for two main approach for subscription merging in two dimensional space. The most straightforward way to merge a set of subscriptions is to represent them by their *minimum bounding rectangle*. The result of merging in this way is a new rectangle that includes all the subscriptions that are being merged. Figure 5.b shows merging of three subscriptions in a two dimensional space using minimum bounding rectangle. The main advantage of using minimum bounding rectangle for merging subscriptions is that the result of merging is also a rectangle that can be easily produced for a known set of subscriptions. However, a disadvantage is that the resulting new subscription may also cover parts of the content space that there is no subscription for them. This may result in false positives in publication dissemination and increases publication dissemination traffic. This merging method is imperfect. The more sophisticated merging technique is perfect merging where *bounding hyperpolygon* is the result of merging. Figure 5.c depicts merging of three subscriptions in a two dimensional space using their bounding polygon. As it is depicted, this technique does not result in extra false positives, however, the generated hyperpolygon complicates content matching and covering algorithms.

The other important issue in subscription merging is the criteria using which set of subscriptions are merged. Selecting subscriptions to merge can be based on the amount of false positives that may result from combining these subscriptions and amount of resource consumption for maintaining subscriptions and performing content matching. The general efficient subscription merging has been proved to be NP-complete [15].

Partial merging: A novel technique that we use in merging subscriptions is *partial* merging. Similar to partial covering relation that described in section 3.4, in partial merging we consider merging parts of subscriptions. This can be done because of representation of subscriptions as a set of intervals. Since each interval can be considered independently, some of intervals of a subscription can be used for merging while some other intervals are not used.

In MICS we can support both perfect and imperfect merging. In the perfect merging, we merge intervals that either have intersection with each other or there is no space between them. Formally, the perfect merging of interval $\mathcal{R}_i = [min_{\mathcal{R}_i}, max_{\mathcal{R}_i}]$ with interval $\mathcal{R}_j = [min_{\mathcal{R}_j}, max_{\mathcal{R}_j}]$ is performed if one of the following conditions holds $\mathcal{R}_i \cap \mathcal{R}_j \neq \varnothing$ or $min_{\mathcal{R}_i} = max_{\mathcal{R}_j} + 1$ or $min_{\mathcal{R}_j} = max_{\mathcal{R}_i} + 1$. The result of merging is a new range $\mathcal{R}_{merge} = [min_{merge}, max_{merge}]$ where $min_{merge} = MIN\{min_{\mathcal{R}_i}, min_{\mathcal{R}_j}\}$ and $max_{merge} = MAX\{max_{\mathcal{R}_i}, max_{\mathcal{R}_j}\}$. In the imperfect merging we merge two intervals if the gap between them is less than a specified threshold. The imperfect merging is done in the same way as the perfect merging. Note that the result of merging can be used for further merging.

3.7 Subscription subsumption in MICS

We formally define the subscription subsumption as follows.

Assume the set of existing subscriptions in a subscription table is represented as $S = \{s_1, s_2, ..., s_n\}$. Also assume that subscription s_m is the new subscription received by the broker. The result of subsumption checking for s_m is true if and only if $s_m \subseteq \bigcup_{i=1}^{n} s_i$.

This is the general definition of subscription subsumption in d-dimensional content space. In MICS subsumption can be checked as follows.

Assume the set of ranges from all existing subscriptions is represented as $\mathcal{R} = \{\mathcal{R}_1, .., \mathcal{R}_n\}$ and the set of intervals for a new subscription is represented as $s_m = \{\mathcal{R}_1, .., \mathcal{R}_m\}$. The result of subsumption checking for s_m is true if and only if for every range $\mathcal{R}_j \in s_m$, $\mathcal{R}_j \subseteq \bigcup_{i=1}^{n} \mathcal{R}_i$ where $\mathcal{R}_i \in \mathcal{R}$.

Similar to subscription covering and merging, since we represent a subscription as a set of intervals, we can examine subsumption of each of these intervals independently. We call this *partial subsumption* since even if a part of a new subscription is covered by a set of subscriptions, this covering is exploited to reduce the corresponding subscription table size.

4. PROPOSED ALGORITHMS IN MICS

In this section we propose subscription covering, subsumption, merging and content matching algorithms based on MICS model. We first describe the data structure we use for subscription table. Then, we present the subscription management (subsumption and merging) algorithm and the content matching algorithm.

4.1 Subscription table structure

Combining the proposed model for representing subscriptions as set of one dimensional intervals and applying subscription covering, merging and subsumption techniques, it is straightforward to show that the resulting intervals for a subscription are discrete and have no intersection. Because if there are overlapping intervals in the subscription table, we can merge them into one interval, therefore, there is no overlapping range in subscription table.

To maintain these non overlapping intervals we propose a subscription table data structure based on B^+Tree [18]. In the B^+Tree, we organize intervals based on their starting point. Every leaf in the tree points to an interval and leaves are sorted based on their start point.

Algorithm 1 Subscription subsumption and merging.

```
 1: Input ← A new subscription in ranges format $s = \{\mathcal{R}_1, .., \mathcal{R}_{\mathcal{N}_s}\}$
 2: Input ← Subscription ranges organized in a $B^+tree$
 3: Output ← List of ranges to forward $\mathcal{L}ist = \varnothing$.
 4:
 5: L1:
 6: for all $\mathcal{R}_i = [min_{\mathcal{R}_i}, max_{\mathcal{R}_i}] \in s$ do
 7:     Look up for $min_{\mathcal{R}_i}$ in the tree
 8:     Assume the visited leaf is $\mathcal{L}_{min}$
 9:     and its corresponding range is $\mathcal{R}_{\mathcal{L}_{min}}$
10: end for
11: Step 1:
12: if $R_i \in \mathcal{R}_{\mathcal{L}_{min}}$ then
13:     $R_i$ is covered. Goto L1.
14: end if
15: Step 2:
16: $CoveredList = \varnothing$
17: if $(\mathcal{R}_{\mathcal{L}_{min}} \in \mathcal{R}_i)$ then
18:     $CoveredList = CoveredList \cup \{\mathcal{R}_{\mathcal{L}_{min}}\}$
19: end if
20: $\mathcal{L}_j = \mathcal{L}_{min}.next$
21: while $\mathcal{R}_{\mathcal{L}_j} \in \mathcal{R}_i$ do
22:     $CoveredList = CoveredList \cup \{\mathcal{R}_{\mathcal{L}_j}\}$
23:     $\mathcal{L}_j = \mathcal{L}_j.next$
24: end while
25: if $CoveredList \neq \varnothing$ then
26:     remove all $\mathcal{R}_{\mathcal{L}_j} \in CoveredList$ from the tree
27: end if
28: Step 3:
29: $\mathcal{R}_{Merged} = \mathcal{R}_i$
30: if $min_{\mathcal{R}_i} \in \mathcal{R}_{\mathcal{L}_{min}}$ or $min_{\mathcal{R}_i} = max_{\mathcal{R}_{\mathcal{L}_{min}}} + 1$ then
31:     $\mathcal{R}_{Merged} = Merge(\mathcal{R}_i, \mathcal{R}_{\mathcal{L}_{min}})$
32: end if
33: if $max_{\mathcal{R}_i} \in \mathcal{R}_{\mathcal{L}_{max}}$ or
    $max_{\mathcal{R}_i} = min_{\mathcal{R}_{\mathcal{L}_{max}}} - 1$ then
34:     $\mathcal{R}_{Merged} = Merge(\mathcal{R}_{Merged}, \mathcal{R}_{\mathcal{L}_{max}})$
35:     Remove $\mathcal{R}_{\mathcal{L}_{max}}$ from the tree
36:     $max_{\mathcal{R}_{\mathcal{L}_{min}}} = max_{\mathcal{R}_{\mathcal{L}_{max}}}$
37:     $\mathcal{L}ist = \mathcal{L}ist \cup \{\mathcal{R}_{Merged}\}$
38: end if
39:
40: if $\mathcal{L}ist \neq \varnothing$ then
41:     forward ranges in $\mathcal{L}ist$.
42: end if
```

4.2 Subscription subsumption and merging algorithm

Algorithm 1 represents the subscription covering, subsumption and merging procedure in MICS. When a new subscription in the form of set of ranges is received by a broker, it uses this algorithm to insert the ranges into the subscription table data structure if necessary and then forwards the subscription ranges that are not subsumed. The algorithm performs the following for each of new subscription's ranges. The first step is to inspect if the range is covered by previously stored ranges. This is done by looking up the $B^{+}tree$ for the starting point of the new range. Assume the leaf of the tree that is visited in this search is represented by $\mathcal{L}_{min}$. Then, if the new range is covered by the corresponding range of the matched leaf, $\mathcal{L}_{min}$, it means that the new range is covered by one or more subscriptions and the algorithm stops for this range. Otherwise, the algorithm starts inspecting if the new range covers other existing ranges. This step is done by going through the sequence of leaves in the tree and checking if they are covered by the new range. Note that in $B^{+}tree$ data structure, leaves are sorted in ascending order from left to right and the sorted list of leaves can be achieved using the pointers between leaves [18]. The traverse through leaves stops when the first leaf with a range that is not covered by the new range is encountered. We represent this leaf by $\mathcal{L}_{max}$. All the covered ranges are added to the passive subscription list and are removed from the $B^{+}tree$ structure. The last step for a new range in the algorithm is to inspect the possible merging with other existing ranges. This step is done by examining two leaves in the tree, $\mathcal{L}_{min}$ and $\mathcal{L}_{max}$. If the ranges for these two leaves can be merged with the new range, the merge operation takes place and these leaves are removed from the tree. Finally the new range or the result of merging is added to the tree and the new range is added to the list of ranges that should be forwarded to the corresponding neighbor. At the end of the algorithm, if this list is empty it means that the new subscription is covered or subsumed by previous subscriptions and there is no need to forward it to the corresponding neighbor. Otherwise, the list is forwarded to the corresponding neighbors.

Note that in the proposed algorithm since we merge all overlapping ranges, subscription subsumption implicitly is evaluated by the step that evaluates covering. The reason is that the covering does not use ranges of a single subscription but it considers result of previously merged ranges that results in subsumption checking too. Hence, if a range is covered by the range resulted from merging several ranges form different subscriptions, it is subsumed by these subscriptions.

An important property of the algorithm is that for each range the algorithm looks up the subscription table ($B^{+}tree$) only once to detect covering, subsumption and merging and the $B^{+}tree$ look up operation has logarithmic execution time. This results in significantly fast evaluation of covering, subsumption and merging evaluation for subscriptions.

4.3 Publication matching algorithm

In this section we propose our content matching algorithm. The content matching algorithm illustrated in algorithm 2 is very similar to the $B^{+}Tree$ search. The algorithm tries to find the leaf in the tree that the publication may fall in. When a new publication is received, the $B^{+}Tree$ is looked up for the match. However, when search reaches to a leaf and it does not match the starting point of an interval, the search does not stop. The algorithm retrieves the range corresponding to this leaf and examines if the publication point is included in this range. If it is included, this means the publication matches at least one of the subscriptions received from the corresponding neighbor and therefore, the publication is forwarded to this neighbor. Otherwise, it is not forwarded to this neighbor.

Algorithm 2 Publication matching.

```
 1: Input ← Publication point $p$
 2: Input ← Subscription ranges organized in a $B^{+}tree$
 3: Output ← TRUE if $p$ is covered, FALSE otherwise.
 4:
 5: Look up for $p$ in the $B^{+}tree$.
 6: Assume the leaf that the search stops in is $\mathcal{L}$ and its corresponding range is $\mathcal{R}$
    if $p \in \mathcal{R}$ return TRUE
    else return FALSE
 7: if $p \in \mathcal{R}$ then
 8:    return TRUE.
 9: else
10:    return FALSE
11: end if
```

5. SCALING TO HIGHER DIMENSIONS

The number of ranges to represent a subscription considerably increases in spaces with higher dimensions [16]. Therefore, applying basic MICS approach for converting subscriptions into one dimensional ranges may result in large number of ranges. In this section we propose two techniques for applying MICS in applications with high dimensionality without generating enormous amount of ranges.

The first technique, *Content Space Projection*, is to select a subset of attributes (dimensions) and project the content space on these dimensions. In fact, we eliminate evaluation of the rest of attributes and consider the space resulting from the selected attributes. While, this approach keeps the number of ranges manageable, it may result in higher amount of false positives during publication dissemination since some of attributes are not represented in the model. Our experimental results indicate that the increased false positives can be from 3% to even higher that 50% depending on the distribution of subscriptions.

The second technique is a hybrid approach that is based on combining MICS with any of the previously proposed approaches. As mentioned, the main drawback of conventional subscription management approaches is that they have a linear execution time with respect to the number of subscriptions. This results in a large search space for subscription management algorithms in a large scale system with enormous amount of subscriptions. In our hybrid approach, which we refer to as *Subscription Indexing*, we use MICS to narrow down the search space for conventional subscription management algorithms. The main idea behind subscription indexing is to use MICS to index subscriptions using a small subset of dimensions. Based on this index, subscription management and content matching algorithms have two phases. In the first phase MICS is used to filter the subscriptions that do not match the incoming subscriptions or publications in the selected dimensions. This narrows down the subscription set that should be used by conventional algorithms. Then, any of the conventional subscription management algorithms is used to evaluate the resulting subscriptions achieved from the first phase. The selection of dimensions for construction of the index for subscriptions plays an important role in the efficiency of subscription indexing. In this paper we use a greedy approach to select the index dimensions based on the selectivity of the attributes. We use the top k attributes that have the highest selectivity for the subscriptions. Applying more sophisticated techniques for efficient selection of attributes to index subscriptions is subject of our future work and out of the scope of this paper.

Subscription management in Subscription Indexing: Similar to our first technique, a subset of attributes are chosen for content space representation. Each subscription has an extra component that is the resulting ranges from mapping the subscription into one dimensional space using the selected subset of its attributes. The $B^{+}tree$ data structure is used as an index to access actual subscriptions. Each leaf in $B^{+}tree$ consists of a range and the list of subscriptions that this range is resulted from. When a subscription is received, its index ranges are used in a similar way as described in section 4.2 to find the set of previously existing subscriptions that must be evaluated for covering, subsumption or merging with this subscription. The result of the algorithm will be a set of candidate subscriptions that should be used for evaluation of covering, subsumption and merging. Then, these subscriptions are evaluated using the existing methods to check if they cover or subsume the new subscription or if they can be merged with it.

Content matching in Subscription Indexing: After finding the range that matches a publication, the matching algorithm does not stop. Instead, it retrieves all the subscriptions corresponding to the matching range and then performs the content matching using these subscriptions. If at least one subscription matches publication, it is forwarded to the corresponding broker.

Compared to content space projection, in subscription indexing the publication matching algorithm performs more computation, however, it has an important advantage that is eliminating false positives in subscription dissemination process. This is achieved by using exact subscriptions for content matching. However, since we prune all other subscriptions and just select a very small fraction of subscriptions for matching in the second phase, it does not have significant computation overhead compared to the case where all subscriptions are evaluated. As we show in Section 7.2 subscription indexing can significantly reduce the number of subscriptions that should be evaluated in high dimensional spaces.

6. DISCUSSION

6.1 Distributed processing of Hilbert indexes

In our content representation model subscriptions and publications are represented using their corresponding indexes generated by Hilbert curve. Computing these indexes for large amount of subscriptions and publications may require significant amount of computation resources. However, such computation can be done very fast and without requiring significant computing power if it is done in a distributed fashion by the subscribing clients. Table 1 depicts the time for computing one dimensional representation of *one* subscription in different settings using a MacBook laptop machine with a Intel Core 2 Duo processor with 2 GHz speed and 1 GB memory running MacOS 10.4.8. Each dimension is partitioned into 256 intervals here. As it can be seen the computation time in all cases is around 0.1 second. Therefore, to avoid imposing such overhead on brokers, we push the conversion process to clients. By acquiring knowledge about the partitioning, each client can compute its limited number of subscriptions and publications in one dimensional format and send them in the one dimensional format to the corresponding broker. This way, computation of Hilbert indexes for subscriptions and publications is done in a distributed fashion and does not result in higher load on brokers.

Table 1: Hilbert index computation time in different dimensions and different subscription sizes

	2	**4**	**8**	**16**
2d	**0.1ms**	**0.1ms**	**0.3ms**	**1.0ms**
3d	**0.1ms**	**0.3ms**	**1.9ms**	**12.8ms**
4d	**0.1ms**	**1.1ms**	**10.2ms**	**124.1ms**

6.2 False positives

The augmentation of subscriptions to represent them as set of cells in partitioned space may result in forwarding publications to brokers that do not have matching subscriptions which we reefer to as false positives. The main negative effect of false positives is increasing publication propagation traffic. The amount of false positive directly depends on the size of partitions that are generated in the system. The larger partitions of the space results in more inaccuracy and consequently higher false positive rate.

Exploiting covering and merging relations among subscriptions in our model results in considerable reduction of augmentation size for subscriptions. For instance, assume a set of overlapping subscriptions. Since we merge all of these subscriptions, the total augmented size for these subscriptions only results from the augmentation in the surface of the resulting hyperpolygon. This is considerably smaller than the sum of subscription augmented area for each subscription.

6.3 Space partitioning with variable cells

As mentioned, having fine grained partitions reduces inaccuracy in subscriptions and results in less amount of false positives in publication dissemination. So far, we considered partitioning of content space into equal cells. Since all parts of content space have the same chance of having publications and subscriptions when the distribution of publications and subscriptions is uniform, we treat all parts of the space in the same way by having equal size partitions. However, if there is prior knowledge about the distribution of publications, we can exploit it to partition the space in such a way that reduces amount of false positives. For instance, if we know that publications in an application follow Zipfian distribution, space partitioning can be done in such a way that the resulting false positive is reduced considerably compared to the case of having equal size partitioning. In this way, partitions in the areas with large amount of expected publications are finer grained compared to other parts of the space. The number of created cells is the same as the equal size partitioning, however, the cell sizes are different depending on the part of content space they are in. Therefore, for a large number of publications, we considerably reduce amount of false positives by reducing the augmentation size for their matching subscriptions. Figure 6 depicts a variable size partitioning in a two dimensional space.

7. EVALUATION

In this section we evaluate our proposed algorithms. We implemented the modified $B^{+}Tree$ for handling intervals in Java. All of the experiments are done on a machine with

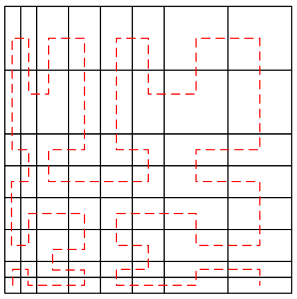


Figure 6: Space partitioning with variable cell sizes.

a 3GHz Intel Pentium 4 processor and 1GB RAM running Ubuntu with Linux kernel 2.6.12. For each simulation run 1GB memory is allocated for JVM. Each attribute domain in the content space is divided into 256 partitions. Similar to previous work in the literature subscriptions and publications are generated using Zipfian and uniform distributions [22]. To create a subscription we first compute the lower bound of the range for each dimension using the selected distribution. Then we add the average subscription size to the computed lower bound to detect the upper bound of the range. We set the default subscription range size to 4. We conducted simulations in 2, 3 and 4 dimensional content space. We compared our approach with a linear implementation of covering, subsumption and merging algorithms.

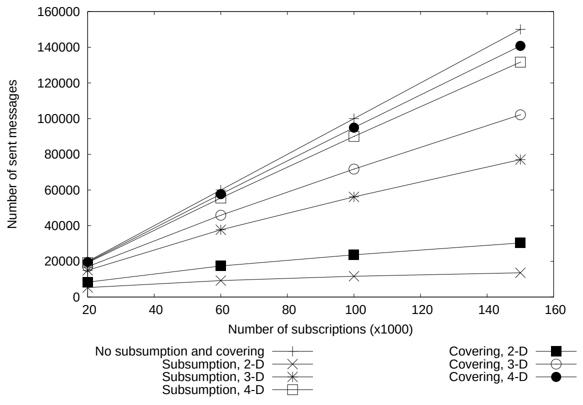


Figure 7: Subscription traffic with and without using covering and subsumption

7.1 Evaluation metrics

We evaluate our proposed content space representation and algorithms based on three main metrics. The first metric is the time required for checking subscription covering, subsumption and merging.

The second metric to evaluate our approach is content matching time. This is a critical factor in content dissemination because content matching is done in every broker along the path of publication dissemination. Therefore, an efficient content matching algorithm plays a critical role in reducing total dissemination time for publications.

The third metric is the amount of ranges stored in subscription table. We consider amount of ranges that should be stored in a subscription table structure in one broker to evaluate the amount of required resource in our approach.

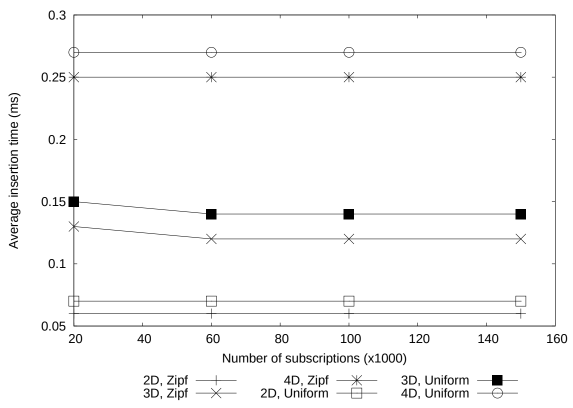


Figure 8: Subscription insertion time with covering, subsumption and merging.

There are two main factors that affect the number of stored ranges. First, representation of each subscription with a set of ranges that increases the number of stored ranges. Second, the partial and total covering, subsumption and merging algorithms we presented in this paper that reduce the number of stored ranges for subscriptions. The number of stored ranges shows the overall effect of these factors.

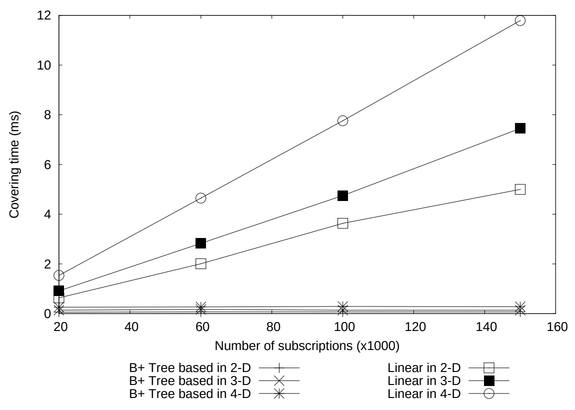
We also evaluate the approach we proposed for dealing with higher dimensions. We evaluate content space projection by measuring the amount of false positives resulted from ignoring subset of dimensions. We then evaluate the effect of subscription indexing approach in MICS to deal with higher dimensionality in the content space. We evaluate this approach by measuring the reduction of the search space for the subscription covering, subsumption and merging.

Our proposed subscription covering, subsumption and merging algorithm also prevents redundant dissemination of subscriptions in the broker overlay network that results in reduced subscription dissemination traffic. The reduced amount of subscriptions depends on the subscription set and is the same for all techniques. However, in order to stress on the importance of subscription subsumption, we present our results in comparing subscription subsumption and covering effects in reducing the subscription forwarding traffic in broker network.

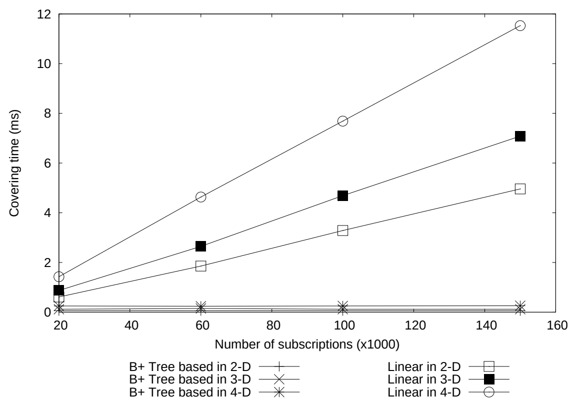
7.2 Experimental Results

Subscription covering and subsumption effect: To show the importance of covering and subsumption we plot the subscription traffic before and after using covering and subsumption in figure 7 for a neighbor broker and different dimensionalities with subscription size 4 and Zipf distribution. The graph depicts three different scenarios. The first scenario is when no covering and subsumption is used. In this scenario all subscriptions are forwarded to the neighbor broker, therefore, the subscription dissemination traffic is the same regardless of the number of dimensions. The second scenario is the effect of using only covering and finally the last one is the effect of using subsumption. As it is shown, exploiting subsumption results in considerable reduction in subscription forwarding traffic compared to the case where only subscription covering is used. The difference is more significant when the dimensionality is smaller because of higher overlapping among subscriptions. The graph also depicts the significant reduction in traffic compared to the case where no covering and subsumption is used.

Subscription insertion time: Subscription insertion process includes checking if the subscription is covered or subsumed by the existing subscriptions. It also exploits all possible merges that can be done by adding the new subscription. Since all of the existing approaches provide separate algorithms for covering, subsumption and merging we only present subscription insertion time for our insertion algorithm described in section 4.2. On the other hand, in order to compare the performance of our proposed approach with the most commonly used linear approach, we adapted our algorithm to just check subscription covering and measured this time for our algorithm and the linear covering algorithm that has been used in [2, 7].



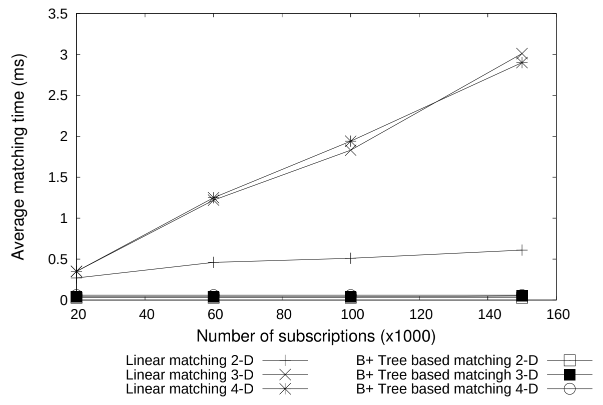
(a) Uniform distribution.



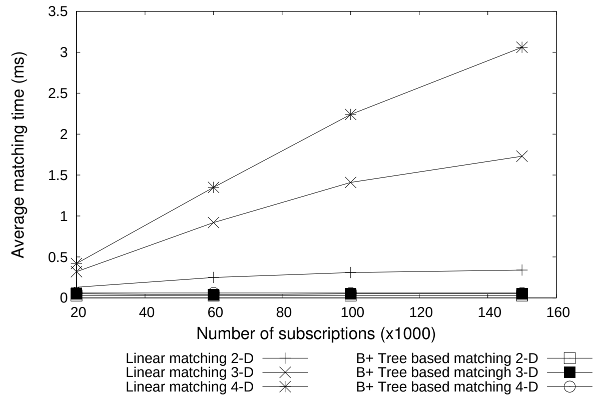
(b) Zipf distribution.

Figure 9: Subscription covering time.

Figure 8 depicts the average subscription insertion time for different subscription table sizes in different dimensionalities with Zipfian and uniform distributions of subscriptions when the subscription size in each dimension is two. Generally, the covering, subsumption and merging time for each subscription in our proposed algorithm and the described setting is in the order of 0.1 ms that is because of logarithmic execution time of the algorithm. As it can be seen, the time is almost the same for different number of subscriptions in the system. The main reason is the $B^{+}Tree$ based subscription management algorithm. However, our measurements



(a) Uniform distribution.



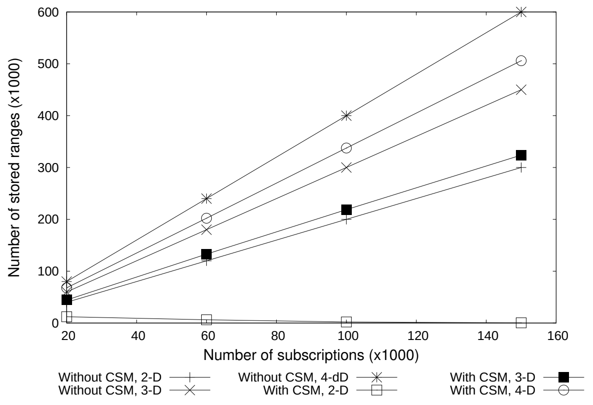
(b) Zipf distribution.

Figure 10: Publication matching time.

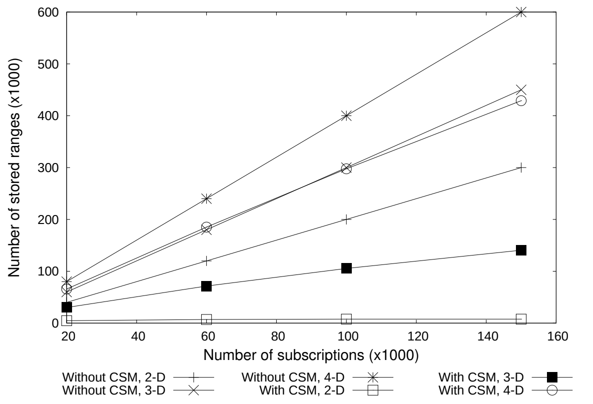
shows that the subscription insertion time considering covering, subsumption and merging increases by dimensionality. The reason for this increase is the increased number of generated one dimensional ranges for each subscription when the space dimension increases. Another interesting fact in the graph is that the insertion time for uniform distribution is more than Zipfian distribution. This is because Zipfian distribution of subscription results in higher covering, subsumption and merging among subscription which results in smaller subscription table size.

In comparing our proposed algorithm to a commonly used linear approach with respect to detection of pairwise covering relation between subscriptions, our approach outperforms the linear one with a significant margin. Figure 9 depicts our results for uniform and Zipfian distributions of subscriptions. As it can be seen, in both cases our algorithm checks subscription covering much faster than the linear algorithm. The other fact depicted in the figure is that the covering detection time for our algorithm does not have considerable fluctuation for different subscription table sizes which is justified by the logarithmic execution time for the algorithm.

Publication matching time: We compare our proposed content matching algorithm with a linear matching algorithm. We evaluate the algorithms based on uniform and Zipfian distributions for subscriptions and uniform distribution for publications. The results are presented in figure 10.

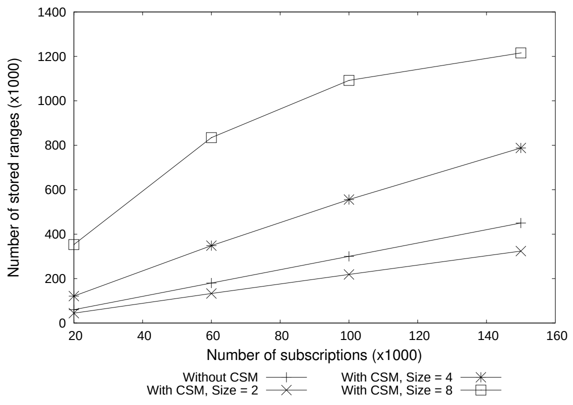


(a) Uniform distribution.

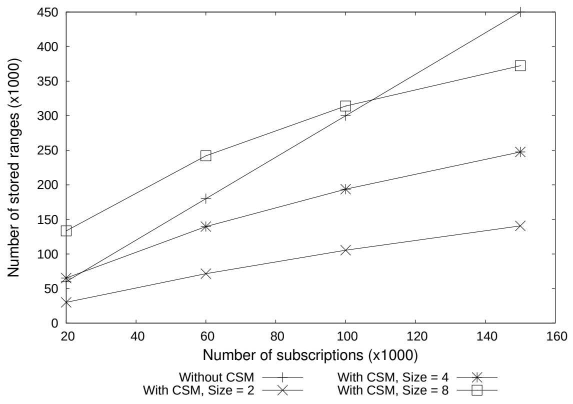


(b) Zipf distribution.

Figure 11: Stored ranges for different dimensionalities.



(a) Uniform distribution.



(b) Zipf distribution.

Figure 12: Stored ranges for different subscription sizes.

The matching time is measured for different subscription table sizes in different dimensionalities for subscription size two in each dimension. The graphs show that our proposed algorithm that uses B^+Tree structure of subscription table significantly outperforms linear content matching. The matching time for linear algorithm increases for larger number of subscriptions, however, since our algorithm has a logarithmic execution time, matching time does not increase significantly.

The other fact that is shown in figure 10 is that the matching algorithm is faster when the subscription distribution is Zipf. This is because of smaller subscription table size resulted from more covering, subsumption and merging among subscriptions.

Subscription storage space: We measure the required storage for subscriptions by counting the number of ranges that should be stored for subscriptions. It is clear that if the number of dimensions in the content space is d, the number of required ranges for a subscription also is d (one range for each dimension). The main parameters affecting the number of generated ranges for a subscription in one dimensional indexing are the number of dimensions and the size of subscriptions. We measure the stored ranges for subscriptions for both of these parameters.

First we consider the effect of the number of dimensions on the stored ranges. Figure 11 depicts the amount of stored ranges after applying subscription covering, subsumption and merging for uniform and Zipfian distributions of subscriptions where subscription size in each dimension is two. Note that in this figure, CSM refers to covering, subsumption and merging. As it is seen by increasing the dimensionality, the number of ranges that should be stored increases. The other fact depicted in the both graphs is that the number of ranges after conversion of subscriptions to one dimensional representation and applying covering, subsumption and merging reduces compared to the actual number of ranges in subscriptions. The reduction is more when subscriptions follow Zipfian distribution that results in higher utilization of covering, subsumption and merging relations among subscriptions.

Increasing subscription size, however, results in larger number of ranges for each subscription that increases the number of stored subscriptions. Figure 12 shows the number of stored ranges for different subscription sizes in a 3-dimensional content space. As it can be observed, by increasing size of subscriptions the number of ranges that should be stored in subscription table increases. This increase is considerable when subscription size in each dimension increases from 2 to 8. However, the increase in the number of stored ranges slows down when the number of subscriptions increases. This is because of the higher probability of overlapping subscriptions that results in more utilization of subscription covering, subsumption and merging. Note that the effect of having more overlapping subscriptions is more considerable when subscriptions follow Zipfian distribution.

Content space projection: As mentioned, one way to use MICS in spaces with higher dimensions is using subset of dimensions and eliminate the rest of them. This results in false positives in content dissemination. Table 2 presents the number of matched publications that must be forwarded and the percentage of false positives in the matched publications for 10K publications and different number of subscriptions with Zipf and uniform distributions when 3 dimensions in a 7-dimension space are used for MICS. As it is seen, very small percentage of publication dissemination traffic results from false positives in Zipf distribution of subscriptions and publications. However, for uniform distribution almost half of the traffic results from false positives. The reason is that in Zipf distribution there is higher chance for most of publications to be matched by real subscriptions. However, the probability of matching is lower for uniform distribution that results in less number of actual matches while increases the portion of traffic resulting form false positives. Therefore, using this approach is more efficient in terms of extra dissemination traffic when subscriptions and publications have Zipf distribution.

Table 2: False positives percentage for 10k publications.

	Traffic (Zipf)	FP% (Zipf)	Traffic (Uniform)	FP% (Uniform)
20K	**5217**	**5.54%**	**209**	**42.11%**
60K	**6472**	**3.71%**	**583**	**42.37%**
100K	**7040**	**3.15%**	**858**	**36.95%**
150K	**7502**	**2.57%**	**1325**	**38.19%**

Subscription indexing effect: To evaluate the effectiveness of subscription indexing in evaluation of covering, merging and subsumption, we measure the reduction in the number of subscriptions that should be evaluated. Table 3 depicts the average number of subscriptions that should be evaluated after looking up the index in a 7-dimensional space for both uniform and Zipfian distributions of subscriptions with subscription size 4. We choose first 3 attributes in the space and index subscriptions using these three attributes. As it is depicted in the graph, the number of subscriptions to evaluate for covering, subsumption and merging significantly reduces. For instance, in Zipfian distribution of subscriptions when there are 100,000 subscriptions, when a new subscription arrives at the broker, it just needs to evaluate 414 subscriptions instead of 100,000. This reduction is more significant when the distribution of subscriptions is uniform. This is because of less overlapping subscriptions in uniform distribution compared to the case where the subscription distribution is Zipf. On the other hand, uniform distribution results in larger subscription table size and more subscription traffic because of less covering, subsumption and merging possibility.

8. RELATED WORK

Most of the existing work have dealt with content matching and subscription covering, subsumption and merging in an isolated way. Several efficient content matching algorithms have been proposed in the literature [10, 13]. A Bloom filter based approach for content matching proposed in [12] that results in high matching efficiency while preserving the expressiveness and flexibility. However, it does not discuss how subscription subsumption or merging can be done in the presented scheme. Tarkoma proposed a chained forest structure for fast matching of profiles and queries [9]. The proposed approach uses chained forests for maintenance and matching of partial orders. Bittner and Hinze proposed an arbitrary Boolean pub/sub model that targets time and space-efficient content matching [11]. In this model, subscriptions and advertisements are presented as arbitrary Boolean expressions and matching, conforming and overlapping relationships among messages, subscriptions and advertisements are defined accordingly.

Table 3: Number of subscriptions to evaluate.

No of Subs	Zipf	Uniform
20000	**64**	**1**
60000	**232**	**1**
100000	**414**	**1**
150000	**600**	**1**

Subscription covering concept in pub/sub systems was introduced in Siena event dissemination system [2]. Siena organizes subscriptions in a partially ordered set (poset) where the order is defined by covering relation. Siena only considers pairwise covering relation between subscriptions and does not exploit subsumption and merging. REBECA is another pub/sub system that not only uses covering, but also considers subscription merging [7]. Subscription covering and merging algorithms in REBECA have linear execution time regarding to the number of subscriptions.

Li *et al.* propose a representation of subscriptions using modified binary decision diagrams in PADRES pub/sub system [6]. They propose subscription covering, merging and content matching algorithms based on this representation. However, these algorithms in worst case may evaluate all the subscriptions. They also do not consider subscription subsumption relation among subscriptions.

Shen *et al.* propose a novel approach for approximate subscription covering detection using Space Filling Curves(SFC) [23]. They convert a d-dimensional subscription into a 2-dimensional point and represent the subscription covering as a point dominance problem which is solved using space filling curve. In this model, if the point for subscription s_1 dominates the point for subscription s_2 then s_1 covers s_2. However, the proposed approach can only detect pairwise covering and it is not clear how to extend it for subsumption and merging.

Ouksel *et al.* present a Monte Carlo type probabilistic algorithm for the subsumption checking [4]. The algorithm has $O(kmd)$ time complexity where k is the number of subscriptions, m is the number of distinct attributes (dimensions) in subscriptions, and d is the number of tests performed to detect subsumption of a new subscription. This algorithm not only has a linear complexity with respect to the number of subscriptions but also may result in false negatives. In this algorithm it is possible that propagation of a subscription is stopped while it is not subsumed by the existing subscriptions. This may result in not delivering publications to some subscribers that may not be acceptable in applications like stock ticker. Compared to this approach our proposed approach not only prevents false negative but also detects subsumption in logarithmic time.

To effectively detect covering subscriptions Triantafillou *et al.* propose an approach based on subscription summaries [8]. Attributes of each incoming subscription are independently merged into their corresponding summary structures. The summaries will ensure reduction in the network bandwidth required to propagate subscriptions and the storage overhead to maintain them. This technique in combination with MICS can be used to deal with higher dimension content spaces. We can partition dimensions into subsets with lower number of attributes and for each attribute subset use our proposed system. Then, we can treat each of these subsets independently similar to the approach proposed in [8] for summarizing indexes and combining matching operation among these attribute subsets.

9. CONCLUSIONS

In this paper we proposed MICS, a new model for representing content space including publications and subscriptions in a content-based pub/sub system. We used multidimensional indexing to represent publications and subscriptions in a one dimensional space. Based on the proposed model, we defined covering, subsumption and merging relationships among subscriptions and proposed an efficient algorithm for evaluating all of these relationships. We also presented a content matching algorithm based on our content representation model. We showed that MICS can effectively be used for systems with larger dimensionality using our proposed subscription indexing technique.

Based on our experimental evaluations, subscription indexing appears to be a promising approach for subscription management in large scale pub/sub systems. As part of future work, we intend to work on different approaches in providing efficient indexing for subscriptions and study the effect of using these indexes in subscription management and content matching algorithms.

10. REFERENCES

[1] P. Th. Eugster, P. A. Felber, R. Guerraoui and A.M. Kermarrec, *The many faces of publish/subscribe.* ,ACM Computing Surveys, Vol 35(2), June 2003.

[2] A. Carzaniga, D.S. Rosenblum, and A.L. Wolf, Design and Evaluation of a Wide-Area Event Notification Service, *ACM Transactions on Computer Systems*, 19(3):332-383, Aug 2001.

[3] P. Costa, M. Migliavacca, G. P. Picco and G. Cugola, Epidemic Algorithms for Reliable Content-Based Publish-Subscribe: An Evaluation. In Proceedings of *ICDCS 2004*, pp. 552-561.

[4] A. M. Ouksel, O. Jurca, I. Podnar and K. Aberer, Efficient Probabilistic Subsumption Checking for Content-Based Publish/Subscribe Systems, In proceedings of *Middleware 2006*, pp. 121-140.

[5] H. Jafarpour, B. Hore, S. Mehrotra and N. Venkatasubramanian. *Subscription Subsumption Evaluation for Content-based Publish/Subscribe Systems*, In proceedings of *Middleware 2008*, pp. 62-81.

[6] G. Li, S. Hou and H.A. Jacobsen, A Unified Approach to Routing, Covering and Merging in Publish/Subscribe Systems Based on Modified Binary Decision Diagrams, In proceedings of *IEEE ICDCS 2005*, pp. 447-457.

[7] G. Mühl. Large-scale content-based publish/subscribe systems. *Ph.D Dissertation*, University of Darmstadt, September 2002.

[8] P. Triantafillou, A. Economides, Subscription Summarization: A New Paradigm for Efficient Publish/Subscribe Systems , In proceedings of *ICDCS 04* , pp. 562-571.

[9] S. Tarkoma, Chained forests for fast subsumption matching. *DEBS 2007*, pp. 97-102.

[10] F. Fabret, H. A. Jacobsen, F. Llirbat, J. Pereira, K. A. Ross, and D. Shasha, Filtering algorithms and implementation for very fast publish/subscribe systems, In proceedings of *ACM SIGMOD 2001*, pp. 115126.

[11] S. Bittner and A. Hinze, The arbitrary Boolean publish/subscribe model: making the case. *DEBS 2007*, pp. 226-237.

[12] Z. Jerzak and C. Fetzer, Bloom filter based routing for content-based publish/subscribe. *DEBS 2008*, pp 71-81.

[13] A. Carzaniga and A. L. Wolf, Forwarding in a Content-Based Network, In proceedings of *ACM SIGCOMM 2003* pp. 163-174.

[14] Srivastava, D., Subsumption and indexing in constraint query languages with linear arithmetic constraints. *Annals of Mathematics and Artificial Intelligence.* 8 (1992), pp. 315343.

[15] A. Crespo, O. Buyukkokten, H. Garcia-Molina: Query Merging, Improving Query Subscription Processing in a Multicast Environment. *IEEE Trans. Knowl. Data Eng.* 15(1): pp. 174-191, (2003).

[16] B. Moon, H. V. Jagadish, C. Faloutsos, J. H. Saltz, Analysis of the Clustering Properties of the Hilbert Space-Filling Curve. *IEEE Trans. Knowl. Data Eng.* 13(1): pp.124-141, (2001).

[17] H.V. Jagadish, Linear Clustering of Objects with Multiple Attributes, In Proceeding of *ACM SIGMOD 1990*, pp. 332-342.

[18] T. Cormen, C. Lieserson, R. Rivest, and C. Stein, "Introduction to Algorithms", second ed. MIT Press, 2001.

[19] P. Costa, G. P. Picco, Semi-Probabilistic Content-Based Publish-Subscribe. In proceedings of *ICDCS 2005*, pp. 575-585.

[20] A. Gupta, O. D. Sahin, D. Agrawal and A. El Abbadi, Meghdoot: content-based publish/subscribe over P2P networks, In proceedings of *Middleware 2004*, pp. 254 - 273.

[21] E. Keogh, K. Chakrabarti, M. Pazzani and S. Mehrotra, Locally adaptive dimensionality reduction for indexing large time series databases, In proceedings of *ACM SIGMOD 2001*, pp. 151 - 162.

[22] A. Riabov, Z. Liu, J. L. Wolf, P. S. Yu and L. Zhang, Clustering Algorithms for Content-Based Publication-Subscription Systems. In proceedings of *ICDCS 2002*, pp. 133-42.

[23] Z. Shen and S. Tirthapura , Approximate Covering Detection among Content-Based Subscriptions using Space Filling Curves. In proceedings of *ICDCS 2007.*