Clustering Algorithms for Content-Based Publication-Subscription Systems

Anton Riabov*
IEOR Department
Columbia University
New York, NY 10027, USA
E-mail: anton@ieor.columbia.edu

Zhen Liu, Joel L. Wolf, Philip S. Yu and Li Zhang
IBM T.J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598, USA
E-mail: {zhenl,jlwolf,psyu,zhangli}@us.ibm.com

Abstract

We consider efficient communication schemes based on both network-supported and application-level multicast techniques for content-based publication-subscription systems. We show that the communication costs depend heavily on the network configurations, distribution of publications and subscriptions. We devise new algorithms and adapt existing partitional data clustering algorithms. These algorithms can be used to determine multicast groups with as much commonality as possible, based on the totality of subscribers' interests. They perform well in the context of highly heterogeneous subscriptions, and they also scale well. An efficiency of 60% to 80% with respect to the ideal solution can be achieved with a small number of multicast groups (less than 100 in our experiments). Some of these same concepts can be applied to match publications to subscribers in real-time, and also to determine dynamically whether to unicast, multicast or broadcast information about the events over the network to the matched subscribers. We demonstrate the quality of our algorithms via simulation experiments.

1 Introduction

Publication-Subscription systems (*pub-sub* for short) provide information on specific real-time *events* from *publishers* to interested *subscribers*. The subscribers express their interest in the form of multiple *subscriptions*. The publishers and subscribers may be located at arbitrary nodes in a distributed network.

Pub-sub systems can be characterized into two broad types based on the degree of generality, usability and personalization allowed to the subscribers. We focus in this paper on the more sophisticated of these, namely *content-based* pub-sub. See, for example, the pioneering work of the Gryphon project [2, 4, 14], as well as NEONet [21] and READY [8]. Borrowing extensively from the classic stock market example used by Gryphon, a subject-based pub-sub system would allow subscriptions based on broad criteria only. A subscriber might request all events related to IBM, for instance. Such a system is powerful but relatively inflexible. The subscriber might receive many publications involving IBM stock market events of little interest. Expanding on the stock market scenario originally discussed by Gryphon, a content-based pub-sub system would allow subscriptions which are based on the conjunction of potentially multiple predicates concerning different attributes. The motivating Gryphon subscription example was based on three distinct attributes:

- The first was an equality predicate based on a character string referring to the stock name. *name*=IBM would be an example, though one could also imagine categories ("blue chip", for instance) being specified instead.
- The second could be one or two inequality predicates based on a two decimal numerical attribute referring to the stock price, such as $90.00 < price \leq 110.00$. Alternatively one could normalize these by the price of the stock at the start of the day, so that a subscription might look for changes in price within 10 percent in a day.
- The third could be one or two inequality predicates based on an integer attribute referring to the volume: $volume > 10{,}000$ might be an example. One could easily imagine translating these volumes into dollars to ensure a common basis amongst differing stocks, so that one might look for volume whose monetary equivalent exceeds one million dollars.

A client with this (expanded) subscription would receive information about all trades of "blue chip" stocks whose price stays within a 10 percent range during the day and involves more than a million dollars in start-of-day units.

So content-based pub-sub is more general and personalizable than subject-based pub-sub. In general, we can assume that content-based pub-sub systems allow each predicate to be *range-based*, composed of *intervals* in the underlying domain of the predicate. (Because computers can handle only inherently finite and discrete attribute values, one can assume without loss of generality that all possible intervals are actually open on the left and closed on the right. This assumption allows the intervals to 'fit together' more cleanly.) By decomposing a subscription with multiple such ranges into multiple subscriptions consisting of single ranges we can see that it is sufficient only to consider intervals, albeit at a cost of more subscriptions. And even attributes such as name, not typically thought of as numerical, can be indexed and therefore linearized in some fashion. A predicate involving "blue chip"s can thus be decomposed into the union of several conjunctions.

This perspective allows us to think of a subscription as a half-open, half-closed aligned rectangle in space, each dimension corresponding to a different attribute. (The term *aligned* means that the rectangle is parallel to the various axes.) A published event becomes a point in the underlying space.

*This work was done while the author was at IBM Research.

Proceedings of the 22 nd International Conference on Distributed Computing Systems (ICDCS'02)
1063-6927/02 $17.00 © 2002 IEEE

Clearly, the extra function involved with pub-sub systems comes at a price. It is technically challenging for a content-based pub-sub system to efficiently distribute the many publication events to the interested subscribers over the network. And it is technically challenging to do so in a manner which scales with the dimensionality of the underlying event space, the number of publishers and the number of subscriptions.

There are two key dynamic problems that a content-based pub-sub system needs to solve:

1. One must *match* a real-time event to determine the set of relevant subscribers. This is the so-called *matching* problem.
2. One must decide to unicast, multicast and/or broadcast information about the event over the network efficiently to the matched subscribers (or possibly to a superset of those describers, if that is more algorithmically reasonable, to be filtered out as necessary). We shall call this the *distribution method* problem.

Both of these problems must be solved in real time as new events are published. However, there is a related, static "preprossessing" problem that should be solved in order to enable the real-time algorithms to function efficiently. Basically we must precompute a set of high quality multicast groups having as much commonality as possible, based on the totality of subscribers' interests. We shall call this the *subscription clustering* problem. This paper will be focused on the clustering problem, whereas the matching problem will be analyzed in more detail in a companion paper [16].

The Gryphon papers do tackle these problems in elegant ways. In [2] and [4], for instance, the authors consider the matching problem. The interested reader is referred to [7] for a detailed description of the matching algorithm, a performance analysis and a comparison with existing matching algorithms. In [4] and [14] the authors consider multicasting techniques. It seems fair to say, however, that the authors base their algorithms primarily on subscriptions in which each dimension is based on either equality or wildcard predicates. (A *wild-card* (*) indicates that the subscriber does not care about the value in that dimension.) While their algorithms will certainly work in full generality, we believe that they are optimized for their motivating predicate types.

However, the above mentioned earlier work of Gryphon project (see, for example, [14]) concluded that multicast mechanism does not provide substantial benefits in many pub-sub systems, and that given the multicast overhead, unicast and broadcast are sufficient for these systems. We believe that the conclusions would be very different depending on the network configurations, distribution of publications and subscriptions. In this paper we consider larger networks with fewer number of subscriptions from each node. We think this setting could be closer to the real-world environments. This leads to the potentially large advantage of forming multicast groups. We will evaluate the benefits of employing (both network-supported and application-level) multicast mechanisms. We shall show that the communication costs depends crucially on how we form the multicast groups. Specifically, in this paper:

- We examine the various assumptions in the Gryphon framework and investigate quantitatively different impacts from several aspects of the pub-sub framework.
- We introduce a general framework that allows to adapt partitional data clustering algorithms for pub-sub systems in which subscriber preferences are described more generally than in previous work.
- We devise a number of new clustering algorithms and enhance some others. Among the algorithms we now consider are *K-means*, a variant called *Forgy K-means*, a hierarchical clustering algorithm based on *Minimum Spanning Trees (MST)*, the *Pairwise Grouping* algorithm and a variant called *Approximate Pairwise Grouping*, and finally a so-called *No-Loss* algorithm. (The no-loss strategy implies that a publication never needs to be filtered out in the dynamic stage: Every subscriber that receives a publication is indeed interested in that message. The other algorithms described in this paper do *not* have this property.) Our comparisons show new leaders among these algorithms, and our results are more robust and realistic.
- We evaluate our algorithms on a large network model. Our subscriptions are based on three different models of interest, and the same is true for our publication model. We analyze the effects of regional subscriber preferences relative to the network topology. We show that the conclusions in this paper depend dramatically on assumptions about the size and structure of the network.

We will consider two flavors of multicasting in this paper, network supported and application level schemes. The interested reader is referred to [3] for a description of the various tradeoffs. See also [15] for a description of application level multicasting.

The remainder of this paper is organized as follows. We introduce some notation in Section 2. In Section 3 we describe some preliminary investigations which illustrate potential impacts of communication algorithms on the communication costs. Then in Section 4 we describe the algorithms for solving common interest clustering problems. In Section 5 we present the results of our experiments and comment on the relative performance of the algorithms. Finally, in Section 6 we summarize the results and discuss future work. The many items there are indicative of the newness and importance of this area of research.

2 Problem Notation

In this section we define some key parameters. Let Ω denote the publication event space. Each event being published within the system can be uniquely described with a single value ω such that $\omega \in \Omega$. Let N denote the number of dimensions (or attributes) in Ω, so that $\Omega \subseteq \mathbf{R}^N$. Let $p_p(x)$ be the probability that publications events are within set $x \subseteq \Omega$. Define the underlying network topology as an undirected graph $G = (V, E)$. Define the communication costs to be $c_e \geq 0$ for each edge $e \in E$. Let $V_P \subseteq V$ be the set of nodes containing publishers. Let $V_S \subseteq V$ be the set of nodes containing subscribers. Let N_S be the number of subscribers. In this paper we shall assume that each subscriber $v_i \in V_S, i = 1, ..., N_S$, has a set of subscription preference expressed by (possibly infinite) rectangles $\mathcal{I}_i = \{b_{ij}\}_{j=1}^{r_i}$ associated with it. Each $b_{ij} \subseteq \Omega$ is an aligned rectangle in space Ω. We define $\mathcal{I} := \bigcup_{v \in V_S} \mathcal{I}_v$ to be the set including all subscription rectangles. We define $k := |\mathcal{I}|$ to be the number of subscriptions.

The size of the clustering problem depends on the dimension of the event space N and the number of subscriptions k. We are interested in algorithms that scale well with respect to these values.

Typically the number K of available multicast groups is known in advance. In the case of network-supported multicast, this is the

Proceedings of the 22 nd International Conference on Distributed Computing Systems (ICDCS'02)
1063-6927/02 $17.00 © 2002 IEEE

number of multicast IP addresses purchased. In the case of application level multicast this number is limited by the amount of memory and processing power of participating computers.

3 Preliminary Analyses

Earlier Gryphon project work [14] demonstrated that the multicast mechanism does not provide substantial benefits in many pub-sub systems. Given the well-known structural and performance overhead for applying multicast, unicast and broadcast are sufficient for these systems, having little or no overhead and insignificant increase in the delivery cost. We examine the various assumptions in Gryphon and investigate quantitatively different impacts from several aspects of the pub-sub framework. The conclusions are very different depending on the network configurations, distribution of publications and subscriptions.

In our model, events are generated in 4 dimensions, the first one corresponding to "regional attribute". When a publication event occurs, the publication is always set to the identification number of originating subnet ("stub") for this message. The *degree of regionalism* parameter is the probability that in a subscription this attribute equals the corresponding subnet number. Zero degree of regionalism corresponds to no regionalism, and degree 1 to absolute regionalism. Regional subscriptions in this table are generated with this parameter set to 0.4. Non-regional subscriptions have this parameter set to 0 during the generation of subscriptions.

The other 3 attributes of events can take on integer values between 0 and 20, according to either uniform or gaussian distributions. Preference in each parameter can be either a "don't care" parameter, denoted "*", which means that all values of this parameter are of interest, or can be specified as an interval. In the uniform case, the probabilities of not having "*" in position 2 is 0.98, and then decreases at a rate of 0.78 for subsequent parameters. Thus the parameter 3 preference is specified with probability $0.98 \cdot 0.78$, and parameter 4 with probability $0.98 \cdot 0.78^2$. If parameter preference is specified, then two random numbers between 0 and 20 are generated, sorted if needed, and assigned to the ends of the preference interval. For the gaussian distribution each of the 3 parameters preference can have a value of "*" with probability q_1, can be a left-ended interval with probability q_2, a right-ended interval with probability q_3 and an interval with both ends with probability $1 - q_1 - q_2 - q_3$. If both ends of the interval are specified, the center of the interval follows a gaussian distribution with parameters (μ_3, σ_3) and the length of the interval follows a Pareto-like distribution with a given mean. If the interval is one-ended, then the end of the interval is chosen from gaussian distribution with parameters (μ_1, σ_1) and (μ_2, σ_2) for left-ended and right-ended intervals. The parameters in the experiment are chosen from the following table, to simulate stock name, price and trading volume:

Para	q_1	q_2	q_3	μ_1, σ_1	μ_2, σ_2	μ_3, σ_3	mean
2	0.1	0	0	8,2	10,2	9,6	1
2	0.15	0.1	0.1	8,1	10,1	9,2	4
2	0.35	0.1	0.1	8,1	10,1	9,2	4

The networks were generated by the GT-ITM package [20] using the transit-stub model with one transit block and the following parameters:

Node	Trans node	Stubs/trans node	Nodes in a stub
100	4	3	8
300	5	3	20
600	4	3	50

More details can be found in section 5.1.

Tables 1 and 2 show the communication costs for broadcast, unicast and ideal multicast, where ideal multicast stands for the distribution scheme in which a multicast group is formed for each publication event and is composed only of the subscribers interested in this event. The ideal multicast could thus use all the possible 2^{N_S} groups. We observe that for the same network, the difference in cost between the broadcast and ideal multicast is small for cases with a large number of subscriptions, and becomes larger as the number of subscriptions decreases. This gap can be as large as 4 times the ideal solution. Thus there is a need to investigate more sophisticated content delivery mechanisms.

Table 1. Degree 0.4 regionalism

Node	Sub'n	Dist'n	Unicast	Broadcast	Ideal
100	5000	uniform	31351	1430	1334
100	5000	gaussian	48805	1430	1415
100	1000	uniform	5846	1430	867
100	1000	gaussian	9513	1430	1134
100	80	uniform	750	1430	310
100	80	gaussian	548	1430	287
300	5000	uniform	38612	5079	3453
300	1000	uniform	8181	5079	867
300	350	uniform	3638	3880	1065
600	10000	uniform	92178	10235	6720
600	10000	gaussian	139020	10235	8212
600	5000	uniform	45320	10235	4820
600	5000	gaussian	69179	10235	6431
600	1000	uniform	5477	10235	1350
600	1000	gaussian	9408	10235	1823

Table 2. No regionalism

Node	Sub'n	Dist'n	Unicast	Broadcast	Ideal
100	5000	uniform	50737	1430	1377
100	5000	gaussian	81779	1430	1428
100	1000	uniform	9409	1430	1039
100	1000	gaussian	16314	1430	1301
100	80	uniform	816	1430	328
100	80	gaussian	1580	1430	545
300	5000	uniform	61513	5079	4019
300	5000	gaussian	98735	5079	4751
300	1000	uniform	13384	5079	2026
300	1000	gaussian	21167	5079	2918
300	80	uniform	61513	5079	1113
300	80	gaussian	6113	5079	1598
600	10000	uniform	151270	10235	7993
600	10000	gaussian	232405	10235	9382
600	5000	uniform	73830	10235	6184
600	5000	gaussian	116952	10235	8000
600	1000	uniform	8276	10235	1791
600	1000	gaussian	14428	10235	2502

The Gryphon framework has a 100 node network, with an average of 125 subscriptions for each of the 80 nodes. For networks with a small number of nodes, each node having many subscriptions per publication, there is a very high probability that at least one of the subscriptions at this node includes this publication. Therefore, the number of nodes interested in this publication will either be very

Proceedings of the 22 nd International Conference on Distributed Computing Systems (ICDCS'02)
1063-6927/02 $17.00 © 2002 IEEE

high or very low. This means that the system needs to deliver the publication either to almost every node or to a very small set of nodes. We believe that this accounts for the conclusion by Gryphon that broadcast is sufficient for such systems. For larger networks with relatively fewer subscriptions, on the other hand, it is unlikely that publications need to be delivered to almost every node in the network. Multicast is most beneficial in this kind of setting, because messages must be delivered to only part of the network.

In Tables 1 and 2 the unicast and ideal multicasts are in general larger for the gaussian distributions than for the uniform. This is due to the fact that more nodes are likely to be interested in the publication events for the gaussian case, resulting in greater communication costs. This shows that the publication distributions also affect the multicast benefits.

Furthermore, Tables 1 and 2 show that the communication costs for regional-specific subscriptions are smaller than those for non-regional subscriptions. More generally, the dependence of the subscriptions by different nodes can have a big impact on the multicast benefits. Consider independent subscriptions by the nodes for a particular publication. The nodes that are interested in this publication would likely be scattered around the network evenly. The multicast benefit of delivering messages to such a scattered network would not be significant. On the other hand, if the subscriptions have a regional concentration, the interested nodes of a publication would very likely be more localized. It would not be surprising to observe substantial benefits from employing multicasts.

In addition, if the probability that each node subscribes to a given message is independent of the other nodes, and there is no concentration of common interest in the event space, then it is difficult to form only a few groups and greatly improve communication efficiency – each of $(2^{N_S} - 1)$ possible multicast groups would be needed with roughly equal probability.

The rest of this paper considers larger networks with fewer number of subscriptions from each node. We think this setting is closer to real-world environments. This leads to the potentially large advantages in forming multicast groups. For such pub-sub systems, using broadcast to deliver messages would not be appropriate due to the large communication overhead. We will evaluate the benefits of employing multicast mechanisms. The communication benefits of multicast depends crucially on how one forms the multicast groups. Due to the overhead of managing a large number of multicast groups, we need to consider forming a limited number of groups. We develop several algorithms for constructing multicast groups and evaluate their performance benefits. Algorithmic complexity is also a key factor for these real-time applications. We further study the cost benefit and running time trade-offs of these algorithm and discover good algorithms for practical applications.

Before going into details on the algorithms, it is important to describe the assumptions of our studies. First, we assume that the peaks in density of subscriptions follow peaks in density of the messages. It seems likely that multicast will not provide comparable improvements in communication costs without this assumption.

We further assume that the subscriber preferences are regional in the network topology. In our experiments, for example, stock name preference (mean of the distribution) was assigned according to the transit block in the network. In addition we assume subscriptions themselves are unevenly distributed in the network, with higher concentrations of interest in some areas, and lower in others.

Under these assumptions forming a limited number of groups using subscription clustering algorithms can potentially lead to a large reduction in communication costs when compared to unicast and broadcast. While relatively restrictive, the assumptions still seem to be practical. Indeed, in many real-life pub-sub systems we would expect that events, in which more people are interested, are typically published more often, than the less interesting events. Also, we can expect the regionalism of subscriptions, with more concentration of users in certain parts of the network, and regionalism of interest, with interest distribution being different in different parts of the network. The model formally presented in section 5.1 follows the above assumptions.

4 Algorithms for Subscription Clustering

There are two distinct categories of subscription clustering algorithms that we present. First category, the *Grid-Based* clustering algorithms, extends earlier work on subscription clustering in content-based pub-sub systems for the case of rectangular preference sets. Work by the Gryphon group [4] and by Katz *et al* [19] employ similar data algorithms for clustering of point subscriptions. In the first subsection we describe the framework that allow us to use data clustering algorithms for clustering subscriptions. In the following subsections we illustrate our approach by describing how four clustering algorithms: *K-means*, *Forgy K-means*, *MST*, and *Pairwise Grouping* can be used for subscription clustering.

The second category of subscription clustering algorithms presented in this section includes just one algorithm – *No-Loss*. While grid-based algorithms sometimes can "lose" messages, sending them to subscribers who are not interested in the particular message, but only happen to have close interests, the No-Loss algorithm guarantees that all subscribers receiving a message are interested in it, thus avoiding redundant communication in the network.

Subscription clustering algorithms form multicast groups, as well as produce data structures that are used for matching events to multicast groups. The last subsection describes matching algorithms for the two categories of clustering algorithms.

4.1 Grid-Based Clustering Framework

Grid-based subscription clustering algorithms (or *cell clustering* algorithms) apply data clustering heuristics to the cells of a regular grid in event space Ω. Data clustering algorithms are widely used for grouping "similar" objects. Similarity of objects is determined based on the value of a distance function. In this subsection we define feature vectors and distance function for clustering, and describe the application of several data clustering algorithms to our model in the following subsections.

Feature Vectors. Each cell $a \subset \Omega$ has a subscriber membership vector $\mathbf{s}(a) \in \{0,1\}^{N_S}$ associated with it. By definition,

$$\mathbf{s}(a)_i := \begin{cases} 1 & \text{if there exists index } j \text{ s.t. } b_{ij} \cap a \neq \emptyset, \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

In this vector non-zero elements correspond to subscribers interested in the cell.

The most commonly used strategy for partitional clustering is the square-error minimization criterion, which amounts to minimizing the sum of the squared Euclidean distances between feature vectors corresponding to objects over the entire set of objects. We use membership vectors as feature vectors of cells instead of using, for example, the coordinates of the cell center in event space Ω. Using

Proceedings of the 22 nd International Conference on Distributed Computing Systems (ICDCS'02)
1063-6927/02 $17.00 © 2002 IEEE

coordinates in Ω for this purpose would lead to poorer solutions, since our goal is to create groups based on common as opposed to *similar* interest. Grouping based on similar interest is considered, for example, in the work by Katz *et al* [19]. In our model, however, we assume that subscribers are only interested in events for which they have directly expressed interest, and there is a performance penalty to be paid when a message is delivered to a subscriber who not interested in it. On the other hand, when several subscribers are interested in the same events, possibly scattered over Ω, it is still logical to assign these subscribers to the same multicast group. Therefore we conclude that the comparison of sets of interested subscribers is more suitable for identifying common interest than comparison of coordinates in the event space.

Distance Function. Squared Euclidean distance between cells a and b can be computed as $d_e^2(a,b) = \sum_i \big(\mathbf{s}(a)_i - \mathbf{s}(b)_i\big)^2 = \sum_i \big(\mathbf{s}(a)_i \oplus \mathbf{s}(b)_i\big)$, where "$\oplus$" denotes a binary operator of exclusive *or*. In our model, the probability density function of publications can be taken into account in order to better characterize our objective. We define the distance function d as: $d(a,b) := p_p(a) \sum_{i \in V_S} \max\big\{[\mathbf{s}(a)_i - \mathbf{s}(b)_i], 0\big\} + p_p(b) \sum_{i \in V_S} \max\big\{[\mathbf{s}(b)_i - \mathbf{s}(a)_i], 0\big\}$. The value of $d(a,b)$ is the expected number of messages sent to subscribers who are not interested in them, if the cells a and b are combined in one group. Note that we can similarly define membership vectors and distance functions for sets of cells, simply replacing cells a and b with sets of cells instead.

The function d characterizes *expected waste*, the notion first introduced in work by Gryphon group [4]. The objective of clustering in this formulation is to form groups in a way that minimizes expected waste. It is known that the heuristics used for clustering are not guaranteed to achieve a global optimum, but in practice the quality of the solution may be sufficient.

Implementation Notes. If for cells a and b it is true that $\mathbf{s}(a) = \mathbf{s}(b)$, then the cells can be combined in one group inducing zero expected waste. Our implementation during preprocessing stage scans continuous blocks of cells searching for repeating sets of subscribers, and joining the cells into *hyper-cells*, in other words sets of cells having the same membership vector.

Since the number of cells might become too high for the algorithm to handle, it would be helpful if we could somehow select the "most popular" cells for clustering, leaving the rest for unicast. Our implementation sorts hyper-cells (after the grouping step described above) by the "popularity rating" $r(.)$ defined by $r(a) := p_p(a) \sum_{i \in V_S} \mathbf{s}(a)_i$ and keeps only a fixed number of cells having the largest values of popularity rating.

Our experiments show that after some number of cells, the improvement gained from feeding more cells to the algorithm becomes negligible. In fact, the more cells are given to clustering algorithm, the worse the quality of solution becomes. This justifies the need for the implementation of outlier removal algorithms for detection of cells that have rather unique combination of subscribers. On the other hand, even without the outlier removal algorithm, clustering a large enough fraction of cells can lead to good results.

4.2 K-Means and Forgy K-Means Clustering

The use of k-means clustering algorithm in pub-sub systems (with different objective, and therefore different feature vectors and distance functions) was proposed by Katz *et al* in [19]. We have studied the performance of the original algorithm by McQueen and as well as the one of its variants by Forgy. See [9] for details.

0. Form initial K groups.
1. Re-assign each cell to a closest group.
2. Repeat step 1 until no cell can be moved.

Figure 1. K-Means Clustering

Figure 1 shows generic pseudo-code for a k-means clustering algorithm to form K multicast groups. In step 0, the initial partition is formed. For this purpose, K hyper-cells with the highest popularity rating $r(a)$ are chosen to be centroids of groups, and the rest of hyper-cells is assigned to closest groups, based on the expected waste distance function. In step 1 each of the hyper-cells is examined, and assigned to the "closest" cluster. We measure the distance between the membership vector of the hyper-cell and the membership vector of the cluster. The K-means algorithm updates the membership vector of the cluster each time a hyper-cell is moved. The algorithm by Forgy updates the membership vectors of all clusters after step 1 is finished. A hyper-cell cannot be moved to another cluster if it is the last hyper-cell in its current cluster.

K-means type algorithms converge to a local optimum in a number of iterations, and in practice they converge quickly. Nevertheless, the processing can be stopped after any iteration, resulting in a feasible partition into K groups. This also provides an easy way to accommodate changes in cell membership, simply running a number of re-balancing iterations, when new subscribers arrive or subscription rectangles are changed in some other way.

4.3 Pairwise Grouping

Pairwise grouping for clustering point interest (or *pairs* for short) was proposed in the work of Gryphon [4], and we have extended these ideas to the case of interest rectangles. It is a top-down clustering algorithm, which starts with each hyper-cell assigned to its own cluster. If the total number of clusters is larger than the required number K, the two groups with the minimum distance to each other are chosen, and combined. The membership vector for this group becomes a combination of vectors of the joined groups. The process is repeated until no more than K groups are left. The algorithm is summarized on Figure 2.

0. Given l cells, form l groups, one cell in each.
   Group $g_i = \{a_i\}$, for each cell a_i, and $\mathbf{s}(g_i) = \mathbf{s}(a_i)$.
1. Find i and j such that $i \neq j$ and $d(g_i, g_j)$ is minimized.
   Reset $g_i \leftarrow g_i \cup g_j$, update $\mathbf{s}(g_i)$, and remove g_j.
2. Repeat step 1 until there are only K groups left.

Figure 2. Pairwise Grouping

An approximate version of this algorithm inspects a fraction $1/e$ of the total number of the group combinations during the search for best distance, stores the pair with the shortest distance from this fraction of groups, and terminates the search after that, if a smaller distance is found. This heuristic derives from a well-known solution to the secretary problem ([17, p. 114]) for maximizing the chance of choosing the maximum value. The algorithm modified in this way works faster, but it may obtain a poorer solution.

Proceedings of the 22 nd International Conference on Distributed Computing Systems (ICDCS'02)
1063-6927/02 $17.00 © 2002 **IEEE**

4.4 Minimum Spanning Tree Clustering

The use of the minimum spanning tree (*MST*) for clustering was proposed by Zahn (1971). Suppose we have a graph G in which nodes correspond to hypercells, and there is an edge between each pair of nodes i and j of length $d(i,j)$. We will say that each node by itself is a component. We process the edges in non-decreasing order of length, if the edge connects different components, combine the components into one, and proceed to the next edge. We continue until there are K components left.

0. Given l cells, form l groups, one cell in each.
1. `For` each pair in the order of increasing distance:
2. If the hyper-cells of the pair are not in same group, combine the groups corresponding to cells.
Repeat step 2 (for next pair) until there are only K groups left.

Figure 3. Minimum Spanning Tree Clustering

This algorithm is similar to Kruskal's algorithm [11] for finding MST in graph G, except that this version stops when exactly K connected components are formed. Pairwise grouping proceeds in the similar fashion as the MST algorithm, but in the pairwise grouping case the distances are calculated between groups, not between cells. Therefore it is impossible to sort pairs by distance in advance, which in effect leads to greater running time of the pairs algorithm, compared to MST on the same data.

4.5 No-Loss Algorithm

The grid-based family of cell clustering algorithms works with cells of a regular grid, and each cell of the grid can be associated with one of the multicast groups. As a result, each subscriber whose interest overlaps with the cell is assigned to the multicast group. Since interest rectangles are not aligned on cell borders, it is possible that an event sent to a multicast group will reach subscribers that are not interested in this particular event, as well as the ones for which this event was intended. The idea behind the *No-Loss* algorithm is to avoid this kind of wasted communication completely, forming multicast groups corresponding to areas aligned to interest rectangles borders. The algorithm (Figure 4) tries to find the "most popular" intersections of interest rectangles. The popularity (or *weight*) of an area in event space is measured by the number $|u(s)|$ of subscribers interested in this area multiplied by the probability of publication in the area: $w(s) = p_p(s)|u(s)|$.

4.6 Matching Subscriptions To Events

Each time an event arrives in the system it must be matched to multicast groups formed by a clustering algorithm in order to find out how to deliver the message. Matching must done efficiently, since the delay caused by the matching algorithm directly affects the maximum throughput of the system. In this subsection we briefly introduce matching algorithms to illustrate how the data structures produced by subscription clustering algorithms are used in real time, and refer the reader to our companion paper [16] for more details.

Each group produced by a clustering algorithm can be described as a set of aligned rectangles in the event space Ω. Therefore the problem of matching an event ω to multicast groups can be reduced to the problem of searching among aligned rectangles in event space

0. Set of rectangles: $S := \mathcal{I}$;
Rectangle weights $w(b_{ij}) := p_p(b_{ij})$ for each $b_{ij} \in S$;
Subscriber node lists: $u(b_{ij}) := \{i\}$ for each $b_{ij} \in S$.
1. Sort elements of S by w, such that:
if $s_l, s_m \in S$ and $l < m$, then $w(s_l) > w(s_m)$.
2. Retain only first T elements
in sets S, w and u, discarding the rest.
3. For each $b_{ij} \in \mathcal{I}$ and $s \in S$, such that $t := s \cap b_{ij} \neq \emptyset$,
and $i \notin u(s)$ do:
if $\exists\, r \in S$ such that $r \equiv t$,
$u(r) \leftarrow u(r) \cup u(t)$; $w(r) \leftarrow p_p(r)|u(r)|$.
else $S \leftarrow S \cup t$; $u(t) \leftarrow u(t) \cup \{i\}$; $w(t) \leftarrow p_p(t)|u(t)|$.
4. Repeat from step 2 at most k times.
5. Sort S as in step 1.
6. Form K multicast groups corresponding to
first K elements of S, grouping subscribers
according to $u(s_l)$ lists, $l = 1..K$.

Figure 4. No-Loss Algorithm

1. Given an event ω, find the corresponding cell of the grid in the event space Ω.
2. `If` the event is matched to a cell:
2.1. Denote the associated multicast group $\mathcal{G}$.
Find the number (or proportion) of members of $\mathcal{G}$, interested in ω.
2.2. `If` the number is above a predefined threshold:
2.2.1. Send the message to group $\mathcal{G}$.
`Else`
2.2.2. Send it only to interested subscribers.
3. `Else` (if the event is not matched)
Send it to the list of interested subscribers.

Figure 5. Matching for Grid-Based Algorithms

Ω for the rectangles that contain a given point ω. This general problem is most commonly solved using a special pre-built data structure called an R^*-tree (see [5]). Alternatively, the S-tree algorithm described in [1] can be used instead.

Matching in Grid-Based Algorithms. Multicast groups formed by a grid-based algorithm are associated with cells of the grid in the event space. Each cell of the grid is either associated with one group or is not associated with any group. Therefore the matching algorithm should find which cell the event falls into, and take different actions according to whether the cell is associated with a group or not. If there is no group associated with the cell, the message must be delivered using unicast. If there is an associated multicast group, the message is usually delivered via multicasting to this group. However, if we can determine how many of subscribers included in the matched group are actually interested in the message, we may be able to avoid unnecessary communication by sending the message via unicast only to those interested in it. This optimization can help to noticeably reduce communication. We shall further discuss the effects of this optimization, as well as present experimental results, in a companion paper [16]. Figure 5 summarizes the matching algorithm in pseudo-code.

Matching in No-Loss Algorithm. The No-Loss interest clustering stage forms a list $A \subseteq S$ consisting of the first n elements of S in the order of decreasing density w. When an event e arrives,

Proceedings of the 22 nd International Conference on Distributed Computing Systems (ICDCS'02)
1063-6927/02 $17.00 © 2002 **IEEE**

the matching algorithm in Figure 6 is applied. The algorithm (using an S-tree, for example) finds multicast groups, as well as individual subscribers that are not included in the groups, whose interest rectangle includes the message.

1. If $e \in s$ where s is a rectangle, $s \in A$
   let s be such that $\forall t \in A : e \in t, w(t) \leq w(s)$
   (i.e. it is the rectangle with greatest density).
   Send message to multicast group formed of $u(s)$.
   Send message via unicast to subscribers in
   $(V_S \setminus u(s))$ that are interested in e.
2. Otherwise send message via unicast
   to subscribers interested in e.

Figure 6. Matching for No-Loss Algorithm

5 Experiments

5.1 Experiment Model

Results have been obtained on 3 different models of subscription interest and message distribution, but on the same network consisting of 600 nodes and the same distribution of subscribers. In all three models 1000 subscription rectangles were generated.

We adopted the GT-ITM package from Georgia Institute of Technology [20] to generate a network with six hundred nodes. This tool generates a hierarchical topology with transit blocks on top, stubs in the middle and nodes at the bottom. In our experiments, we first generate three transit blocks, with an average of five transit nodes in each block. Each transit node is connected on average to two stubs, and each stub has an average of twenty nodes.

For a given network topology, we generate subscriptions for each node. We first generate one thousand subscriptions with a $\{40\%, 30\%, 30\%\}$ breakdown for the three transit blocks. Within each transit block, there is a Zipf-like distribution for the number of subscriptions between all the stubs connected to this transit block. Subscriptions are distributed according to another (common) Zipf-like distribution within each stub.

The generated interval subscriptions are of the form $\{bst, name, quote, volume\}$. The first field bst, which could stand for *buy, sell*, and *transaction*, takes value B, S and T with probabilities $0.4, 0.4$, and 0.2, respectively. The center of the interval for the $name$ field follows a normal distribution, with mean centered around the points specific to transit block number ($3, 10$ and 17), and standard deviation of 4. The length of this interval also follows a Zipf distribution. The intervals for the $quote$ and $volume$ fields are generated according to the same parametric distribution with different parameters. This parametric distribution takes values as follows:

$(-\infty, +\infty)$, with probability q_0,
$[n, +\infty)$, with probability q_1, and $n \sim N(\mu_1, \sigma_1)$,
$(-\infty, n]$, with probability q_2, and $n \sim N(\mu_2, \sigma_2)$,
$[n_1, n_2]$, otherwise, center of interval $\sim N(\mu_3, \sigma_3)$, interval length follows a Pareto distribution.

The parameters are given in the following table:

	q_0	q_1	q_2	μ_1, σ_1	μ_2, σ_2	μ_3, σ_3	c, α
price	0.15	0.1	0.1	9, 1	9, 1	9, 2	4, 1
vol	0.35	0.1	0.1	9, 1	9, 1	9, 2	4, 1

The generation of the subscriptions are intended to mimic the real-life scenario [16] that people's interests in stocks are centered around the current prices, the popularity of the information for different stocks has a Zipf-like distribution, and the popularity of the participants also has a Zipf-like distribution.

The publications are points in the subscription space, which are generated according to a mixture of multivariate normal distributions. The different peaks in the distributions represent the multiple hot spots where events are published more frequently. We study three scenarios, which are mixtures of one, four and nine multivariate normal distributions. The means and standard deviations for the single mode multivariate normal distribution are $(1, 1), (10, 6), (9, 2), (9, 6)$ for each of the four dimensions. The four mode distribution is constructed by sampling independent mixtures of multivariate normal distributions in each dimension. The mean and standard deviations for the first and fourth dimensions are $(1, 1)$ and $(9, 6)$, respectively. The second dimension is a normal random variable with parameters $(12, 3)$ with probability 0.5, and with parameters $(6, 2)$ with probability 0.5. The third dimension is a normal random variable with parameters $(4, 2)$ with probability 0.5, and with parameters $(16, 2)$ with probability 0.5. Similarly, for the nine mode distribution, the parameters for the first and fourth dimensions remain the same. The third dimension is N(4,3) with probability 0.3, N(11,3) with probability 0.4 and N(18,3) with probability 0.3. The fourth dimension is N(4,3) with probability 0.3, N(9,3) with probability 0.4 and N(16,3) with probability 0.3.

It should be noted that this experimental framework is flexible enough to accommodate other probability distributions for the subscriptions and publications. In the following study, we constructed 1000 subscriptions for the network with 600 nodes generated by the GT-ITM package.

We performed experiments on the generated testbed to evaluate the performance of the different schemes for forming multicast groups under two multicast frameworks: network supported and application-level multicast schemes. Network supported multicast requires the network routers to have the multicast capabilities, to be able to recognize multicast groups, and to forward the information to the proper members of the group. There are two types of multicast algorithms currently used in routers: *dense mode* and *sparse mode* multicast. The implementations differ in the amount of state information and in the structure of the routing tree. We assume the *dense mode* multicast where the routing tree is a shortest path tree rooted at publisher. The amount of state information is proportional to both the number of publishers and the number of groups. In recent years, the study of multicast mechanisms has focused on the application level multicast, which does not require full support at the network routers. The members of a multicast group communicate through unicasts. They form a minimum spanning tree and forward the messages from one member to another through the tree. We also evaluated the impact of the application level multicasts for the different algorithms.

5.2 Experiment Results

The following plots summarize simulation results in which the cost of communication was computed by summing up the edge costs (generated by GT-ITM package) on the links on which communication takes place. Since absolute values of costs differ for different networks, we normalize the costs to make comparison easier. Thus the vertical axis in most plots shows "improvement percentage" over unicast. In other words, 0% communication cost improvement is achieved by using unicast to deliver each message.

Proceedings of the 22 nd International Conference on Distributed Computing Systems (ICDCS'02)
1063-6927/02 $17.00 © 2002 IEEE

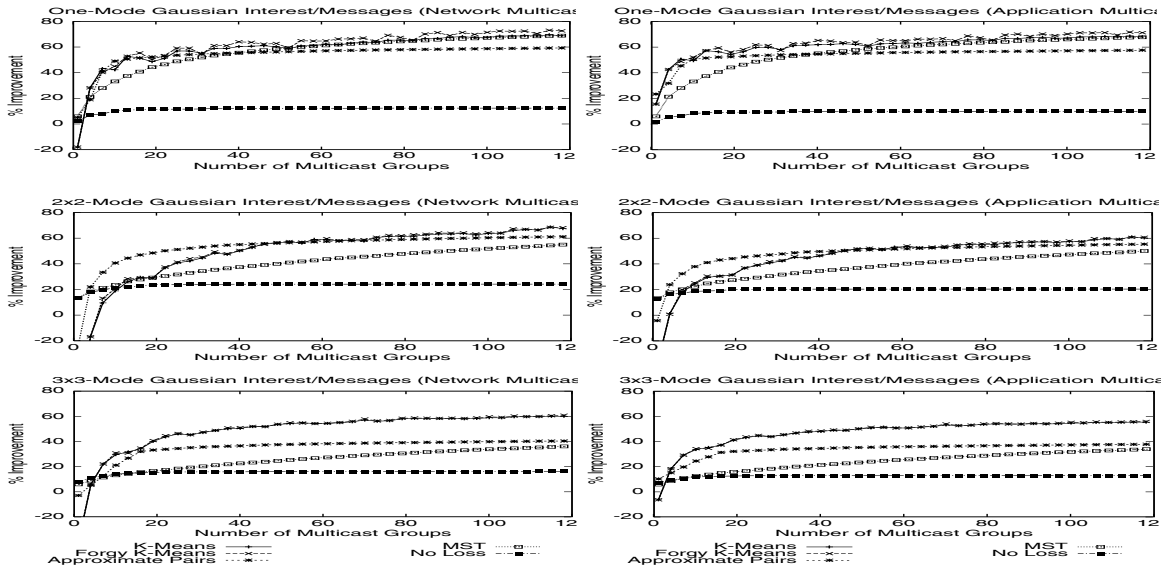

Figure 7. Algorithmic Comparison

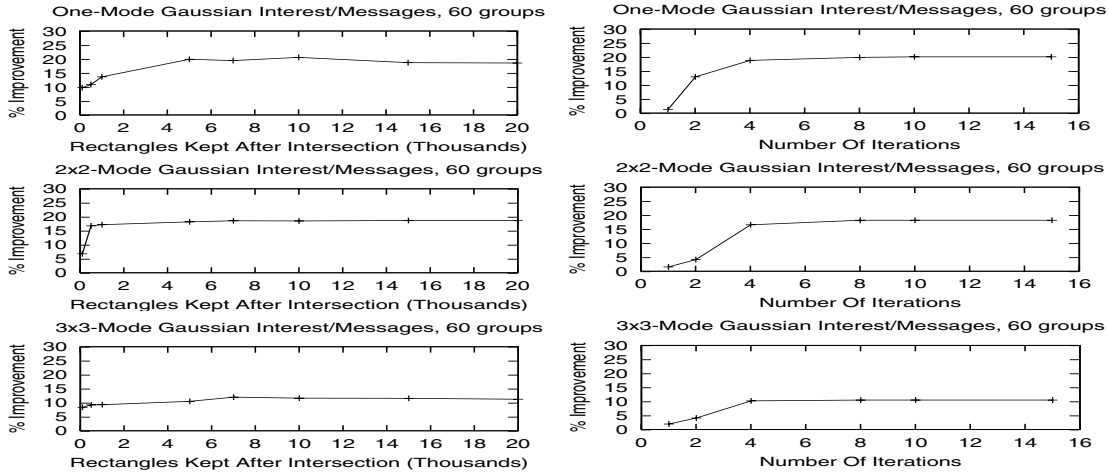

Figure 8. Effect of number of rectangles and iterations on no-loss algorithm

100% cost improvement corresponds to the cost of delivering each message to a multicast group specially formed only of clients interested in this particular message, which is the best possible, and in the worst case requires as many as $O(2^{N_S})$ multicast groups. The goal of clustering algorithms is to get as close to this performance bound as possible while using no more than K groups.

The absolute communication costs depend on different parameters. For the case of one-mode gaussian subscription distribution, the unicast is 7139, the broadcast is 8536, and the ideal solution of network supported multicast is 1763.

Figure 7 shows how the communication cost changes as more groups become available for different algorithms. We are interested in algorithms that demonstrate monotone improvement, since intuitively we expect the algorithms to do a better job with more groups. Each plot is shown for network-level multicast and application level multicast. While application-level multicast results in slightly higher costs, the trend remains the same, and the algorithms that perform better under network multicast maintain their leadership under application-level multicast.

Performance of k-means is almost the same as that of Forgy k-means. The approximate pairs curve closely follows the curve of the pairs algorithm, as one can see in figure 10, so the latter is not shown in order to make plots readable.

The algorithms were run with the following parameters. K-means and Forgy used 6000 rectangles and maximum of 100 iterations (usually the number of actual iterations was less than 20). The approximate pairs algorithm used only 2000 rectangles. The time complexity graph in figure 10 shows that in this case time complexity is almost the same as that of K-means. No-Loss algorithm was run with 5000 rectangles kept after intersection and 8 iterations. Figure 8 shows how the algorithm depends on these parameters. MST was run with 6000 rectangles.

Figure 9 shows that the trend of the algorithm performance does not depend greatly on the network topology. The left plot is taken from figure 7. The right plot is obtained from simulation on a different network with subscription assignment generated according to the same parameters, but with different random seeds. While the grid-based clustering algorithms achieve local optimal solutions at

Proceedings of the 22 nd International Conference on Distributed Computing Systems (ICDCS'02)
1063-6927/02 $17.00 © 2002 IEEE

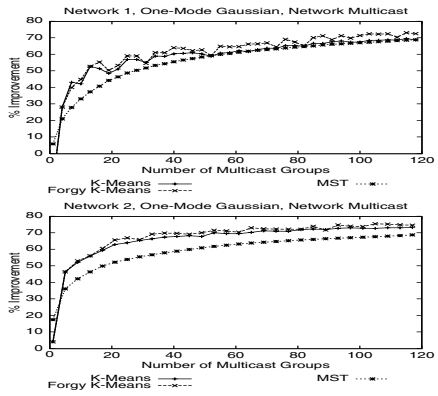

Figure 9. Algorithmic Comparisons

best, in practice the solutions usually are good enough, reaching 60% in this case.

Figure 11 combines the left and right plots of figure 10. Notice that the Forgy and K-Means performance plots go down when given more time. The quality of solution can degrade because the number of wasted messages becomes large. Data clustering algorithms usually make use of outlier removal techniques to avoid these problems. We leave the study of outlier removal effects for future work, simply noting here that the parameter that regulates the number of cells given to the clustering algorithm regulates its performance as well as the time it takes to produce a solution. This is shown in figure 10.

The results in figures 10 and 11 indicate that the Forgy clustering should be preferred over k-means, since it gives comparable or better results faster. Cell-based clustering works well when the dimensionality of the event space is not too high and the granularity of subscription interest is not too high. In this case these clustering methods should be preferred over the No-Loss algorithm. We leave the high-dimensional case for future study.

6 Conclusions and Discussions

We have considered the issue of efficient communication schemes based on multicast techniques for content-based pub-sub systems. We have proposed and adapted clustering algorithms to form multicast groups for these content-based pub-sub systems. These algorithms perform well in the context of highly heterogeneous subscriptions, and they also scale well. An efficiency of 60% to 80% with respect to the ideal solution can typically be achieved with less than 100 multicast groups.

Our experiments indicate that under several assumptions, which include a high degree of regionalism of interest and a similar distribution of messages, the Forgy algorithm should be preferred for most purposes. Iterative clustering algorithms (K-means and Forgy) seem to be better suited for subscription dynamics, although other algorithms can be adapted as well. Hierarchical clustering algorithms (MST and Pairs) have poorer performance than iterative clustering, but have the advantage of monotone improvement: when more multicast groups become available, the new groups are formed by subdividing the existing ones. Analysis in this paper of the effects of algorithmic parameters on performance should help guide practical implementations.

There are still many open issues to be addressed.

1). Proposed algorithms can be adapted to make use of non-rectangular subscription interest sets by rounding the sets to appropriate shapes. While the no-loss algorithm relies on the rectangular interest set assumption, it is not very important to the other (grid-based) algorithms. The same grid data structures can be created without requiring the sets to be rectangles.

2). In many real-world scenarios each client is connected to an ISP via a single last-mile link. One simple variant to extend the transit-stub network topology [20] involves assigning higher costs to the last-mile links, since these are usually the slowest and the most congested ones.

3). Evaluation of the algorithms with real-world data would be helpful. For example, stock trading data can be used to simulate a stream of events coming into the system. However, information about the real structure of subscriptions is harder to obtain.

4). In our experiments we did not simulate real network packets, implicitly assuming that there are no delays caused by congestion of network links. This is a reasonable assumption to make when the message size is small (1K or less). If the messages have large sizes, a different type of communication cost evaluation must be used.

5). In reality clustering groups need to be constantly updated, since subscribers change their preferences, join and leave the network. Katz *et al* show that iterative clustering algorithms are well suited for dynamic changes in subscription structure [19]. Although a different type of distance measurement is considered in that paper, the same iterative improvement strategy could be used to update the data structures of the k-means and Forgy cell clustering algorithms.

6). In our model we have assumed that the matching of messages to groups or individual subscribers is done once: the first "intelligent" node that receives the message decides how to route it. In an alternative approach, described in several Gryphon project papers [2, 14], each intermediate node knows about the preferences of its neighbors, and matches each event against its specific data structures to find those neighbors to which the event must be forwarded next. This approach may save communication and matching time. However in practice the dynamics of subscriptions require subscription changes to propagate quickly in the network, which makes this approach difficult to implement. Another related extension of the pub-sub model requires the system to store messages at intermediate nodes, allowing off-line clients to retrieve information of interest when they connect to the system [6].

References

[1] C. Aggarwal, J. Wolf, P. Yu and M. Epelman, "Using Unbalanced Trees for Indexing Multidimensional Objects". *Knowledge and Information Systems,*, Vol. 1, pp. 309-336, 1999.

[2] M. Aguilera, R. Strom, D. Sturman, M. Astley and T. Chandra. "Matching Events in a Content-Based Subscription System", *ACM Symposium on Principles of Distributed Computing*, Atlanta, USA, 1999.

[3] K. Almeroth, "The Evolution of Multicast: From the MBone to Inter-Domain Multicast to Internet2 Deployment", *IEEE Network*, January/February 2000.

[4] G. Banavar, T. Chandra, B. Mukherjee, J. Nagarajarao, R. Strom and D. Sturman, "An Efficient Multicast Protocol for Content-Based Publish-Subscribe Systems" *International Conference on Distributed Computing Systems*, 1999.

Proceedings of the 22 nd International Conference on Distributed Computing Systems (ICDCS'02)
1063-6927/02 $17.00 © 2002 **IEEE**

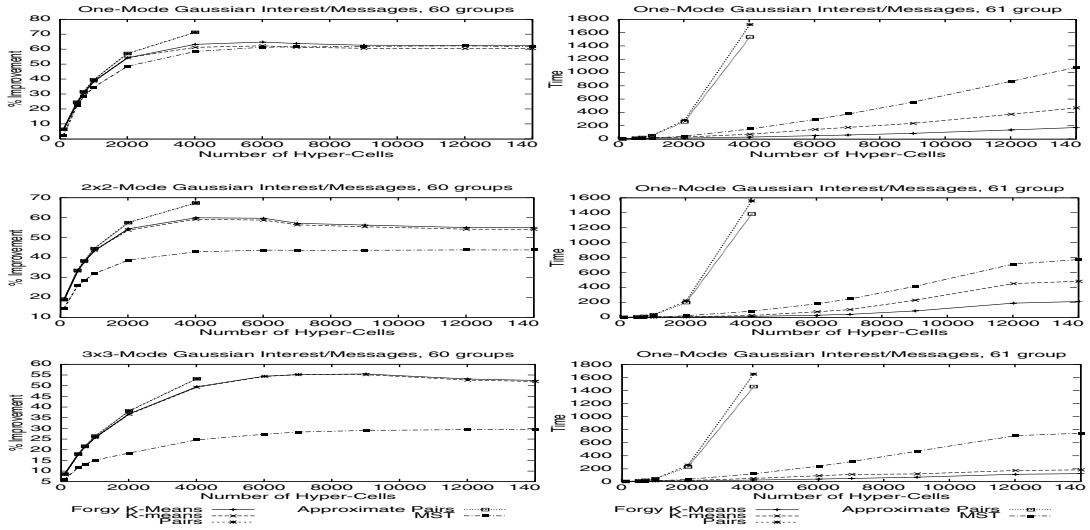

Figure 10. Effect of input data size

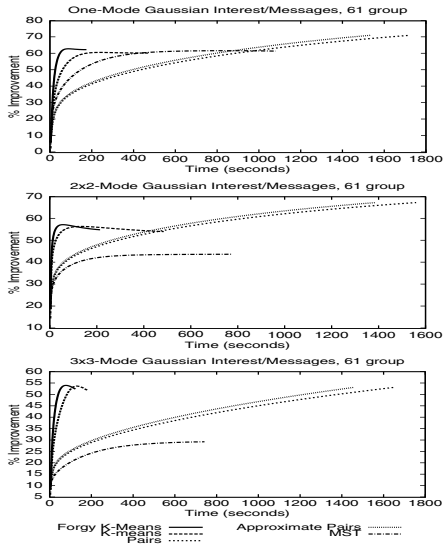

Figure 11. Solution Quality as Function of Time

[5] N. Beckman, H. Kriegel, R. Schneider and B. Seeger, "The R*-Tree: An Efficient and Robust Method for Points and Rectangles," *ACM SIGMOD Conference*, pp. 322-331, 1990.

[6] C. Binding, S. G. Hild, R. Hermann and A. Schade, "Intelligent Messaging for the Enterprise." IBM Research Report, RZ 3353 (# 93399), June 2001.

[7] F. Fabret, F. Llirbat, J. Pereira, and D. Shasha. "Efficient matching for content-based publish/subscribe systems." Technical report, INRIA, 2000. `http://wwwcaravel. inria.fr/ pereira/matching.ps.`

[8] R. E. Gruber, B. Krishnamurthy, and Panagos, "The Architecture of the Ready Event Notification Service", *IEEE International Conference on Distributed Computing Systems Middleware Workshop*, 1999.

[9] A. K. Jain and R. C. Dubes. 1988. *Algorithms for Clustering Data*. Engelwood Cliffs, New Jersey: Prentice Hall.

[10] I. Kamel and C. Faloutsos, "On Packing R-Trees," *International Conference on Information and Knowledge Management,* 1993.

[11] J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem.", *Proceedings of the AMS,* vol. 7, pp. 48-50, 1956.

[12] B. Levine, J. Crowcroft, C. Diot, J. Garcia-Luna-Aceves and J. Kurose, "Consideration of Receiver Interest for IP Multicast Delivery", *Infocom*, Tel Aviv, 2000.

[13] M. Oliveira, J. Crowcroft and C. Diot, "Router Level Filtering for Receiver Interest Delivery", *of ACM NGC*, 2000.

[14] L. Opyrchal, M. Astley, J. Auerbach, G. Banavar, R. Strom and D. Sturman, "Exploiting IP Multicast in Content-Based Publish-Subscribe Systems", *IFIP/ACM International Conference on Distributed Systems Platforms*, New York, 2000.

[15] D. Pendarakis, S. Shi, D. Verma and M. Waldvogel, "ALMI: An Application Level Multicast Infrastructure", Symposium on Internet Technologies, 2001.

[16] A. Riabov, Z. Liu, J. Wolf, P. Yu and L. Zhang, " On the Matching Problem in Content-Based Pub-Sub Systems", in preparation.

[17] S.M. Ross, "Introduction to Probability Models", Sixth Edition, Academic Press, San Diego, 1997.

[18] T. Sellis, N. Roussopoulos and C. Faloutsos, "The R+ tree: A Dynamic Index Structure for Multi-Dimensional Objects", *International Conference on VLDB,* 1987.

[19] T. Wong, R. Katz and S. McCanne, "An Evaluation of Preference Clustering in Large-Scale Multicast Applications", *IEEE Infocom*, Tel Aviv, 2000.

[20] E. Zegura, K. Calvert and S. Bhattacharjee, "How to Model an Internetwork", *IEEE Infocom*, San Francisco, 1996.

[21] New Era of Networks Inc. NEONet `http:// www.neonsoft.com/products/NEONet.html.`

Proceedings of the 22 nd International Conference on Distributed Computing Systems (ICDCS'02)
1063-6927/02 $17.00 © 2002 **IEEE**