Efficient Event Routing in Content-based Publish-Subscribe Service Networks

Fengyun Cao
Computer Science Department
Princeton University
Princeton, NJ 08540, USA
fcao@cs.princeton.edu

Jaswinder Pal Singh
Computer Science Department
Princeton University
Princeton, NJ 08540, USA
jps@cs.princeton.edu

***Abstract*—Efficient event delivery in a content-based publish/subscribe system has been a challenging problem. Existing group communication solutions, such as IP multicast or application-level multicast techniques, are not readily applicable due to the highly heterogeneous communication pattern in such systems. We first explore the design space of event routing strategies for content-based publish/subscribe systems. Two major existing approaches are studied: *filter-based* approach, which performs content-based filtering on intermediate routing servers to dynamically guide routing decisions, and *multicast-based* approach, which delivers events through a few high-quality multicast groups that are pre-constructed to approximately match user interests. These approaches have different trade-offs in the routing quality achieved and the implementation cost and system load generated. We then present a new routing scheme called Kyra that carefully balance these trade-offs. Kyra combines the advantages of content-based filtering and event-space partitioning in the existing approaches to achieve better overall routing efficiency. We use detailed simulations to evaluate Kyra and compare it with existing approaches. The results demonstrate the effectiveness of Kyra in achieving high network efficiency, reducing implementation cost and balancing system load across the publish-subscribe service network.**

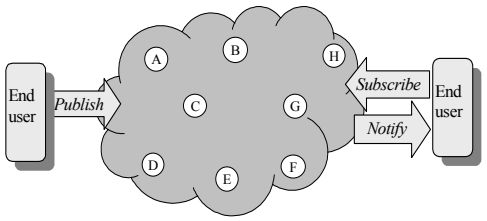
***Keywords*—System design, simulations, publish-subscribe, event notification**

I. INTRODUCTION

Publish-subscribe (pub-sub for short) is an important paradigm for asynchronous communication between entities in a distributed network. In the pub-sub paradigm, subscribers specify their interests in certain event conditions, and will be notified afterwards of any event fired by a publisher that matches their registered interests. Such timely notification of customized information is of great value for many distributed applications, such as enterprise activity monitoring and consumer event notification systems [5][7][12], mobile alerting systems [1][35], etc.

Pub-sub systems can be characterized into three broad types based on the expressiveness of the subscriptions they support. In *topic-based* and *subject-based* schemes, events are classified and labeled by publisher as belonging to one of a predefined set of subjects. This type of pub-sub system is able to leverage existing group-based multicast techniques for event delivery, by assigning each subject to a multicast group. *Content-based* pub-sub is a more general and powerful paradigm, in which subscribers have the added flexibility of choosing filtering criteria along multiple dimensions, using thresholds and conditions on the contents of the message, rather than being restricted to (or even requiring) pre-defined subject fields. Content-based pub-sub applications present a unique challenge not only for efficient matching of events to subscriptions but also for efficient event delivery. In particular, content-based subscriptions can be highly diverse, and different events may satisfy the interests of widely varying groups of subscribers. As a result, mapping events into exact multicast groups may require the number of groups exponential in the number of subscribers (i.e. 2^n where n is the number of subscribers) in the worst-case scenario. Thus, existing group-based multicast techniques cannot readily be applied to such systems.

In this paper, we study the event delivery problem in the context of a content-based pub-sub service network. The general architecture of a pub-sub service network is shown in Figure 1: a set of pub-sub servers are distributed over the Internet; clients access the pub-sub service, either to publish events or to register subscriptions, through appropriate servers, such as the ones that are close to them or in the same administrative domains. Thus, pub-sub servers serve as publication proxies as well as subscription proxies on behalf o clients, and we can view the problem as one of getting published events to the pub-sub servers that subscribe – as proxies – to the events. Communication between pub-sub servers with their associated clients is a separate matter and is not discussed in this paper. We focus on the following questions:



Server	Subscriptions
A	{1,5}
B	{7,8}
C	{1,2}
D	{0,6}
E	{3,5}
F	{5,7}
G	{4,6}
H	{2,9}

Figure 1. Example of a pub-sub service network with eight pub-sub servers. The subscriptions submitted to the servers are listed in the table on the right. Events are represented by integer values between 0 and 9.

- What should the interconnection topology of the pub-sub servers look like?
- How should events be correctly and efficiently routed through the network to the interested subscribers?

We use the following metrics to evaluate the efficiency of an event routing scheme: the storage, management, and computation costs at the pub-sub servers, and the network resource utilization for event transmission.

Existing event routing solutions can be largely categorized into two classes: the *filter-based* approach [12][7][22][29] and the *multicast-based* approach [22][16][25][34]. In the filter-based approach, routing decisions are made via successive content-based filtering at all nodes from source to destination: every pub-sub server along the way matches the event with remote subscriptions from other servers, and then forwards it only toward directions that lead to matching subscriptions. This approach can achieve high network efficiency, but at the cost of expensive subscription information management and high processing load at pub-sub servers.

In the multicast-based approach, a limited number of multicast groups are computed before event transmission begins. For each event, the routing decision is made only once at the publisher, mapping the event into the single appropriate group. The event is then multicast to that group, assuming IP multicast [13] or application-level multicast [9][10] support. Because only a limited number of multicast groups can be built, servers with different interests may be clustered into same group, and events may be sent to uninterested servers as well. The network efficiency of this approach is often highly sensitive to the data types and the distributions of events and subscriptions in the application.

In this paper, we propose a new event routing scheme called Kyra. The goal of Kyra is to reduce the implementation cost of the filter-based approach while still maintaining comparable network efficiency. The main idea is to construct multiple smaller routing networks, so that filter-based routing is implemented in each one with lower cost. Server load is reduced because each Kyra server is guaranteed to only participate in a small number of routing networks. This is achieved through strategically "moving" subscriptions between servers to improve content locality. Therefore, the effectiveness of Kyra is independent of data characteristics of pub-sub applications. Detailed simulation results show that Kyra significantly reduces the storage, processing and network traffic loads on pub-sub servers, while achieving network efficiency close to that of the filter-based approach. Kyra also balances routing load across the pub-sub service network.

The remainder of the paper is organized as follows. We study the two major existing approaches in Section II and present Kyra system design in Section III. We describe our performance evaluation methodology in Section IV, and present detailed simulation-based evaluation of Kyra and other routing schemes in Section V. Section VI discusses related work and Section VII concludes the paper.

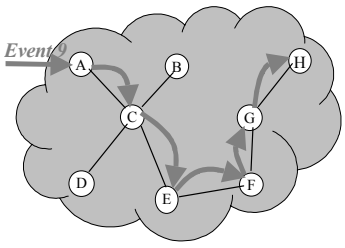
II. OVERVIEW OF EXISTING SOLUTIONS

In this Section, we briefly review two major state-of-the-art event routing approaches and discuss their trade-offs. The analysis explains our observations and leads to the design of Kyra.

A. Filter-based event routing

We use the implementation of Siena system [7] as a representative for the filter-based event routing approach. The architecture is as shown in Figure 2. Pub-sub servers are organized into an acyclic (tree) peer-to-peer topology[1,2]. First, all subscriptions are broadcast over the entire network along the tree topology[3]. Each server then records the subscriptions received from each direction in its routing table. When an event is received, it is matched against subscriptions in the routing table and forwarded toward only the directions with matching subscriptions.

Since events are only routed in the directions to which they are relevant, filter-based event routing achieves network efficiency in an elegant way. However, the implementation and management cost can be high. First, the cost of flooding and replicating all subscriptions at all pub-sub servers grows super-linearly against total number of subscriptions in the system. Although summarization techniques such as *merging* and *covering* have been proposed to alleviate this problem, it is an open question as to how efficiently and effectively they can perform, especially with multi-dimensional data types. Even with the simple, one-dimensional example shown in Figure 2, the routing tables still contain a lot of information, much of which is duplicated over many servers. The second problem is that event routing can result in high processing and network traffic load at pub-sub servers that are not interested



Server	Routing table	
	Neighbor	Subscriptions
A	C	{0-9}
B	C	{0-7,9}
C	A	{1,5}
	B	{7,8}
	D	{0,6}
	E	{2-7,9}
D	C	{1-9}
E	C	{0-2,5-8}
	F	{2,4-7,9}
F	E	{0-3,5-8}
	G	{2,4,6,9}
G	F	{0-3,5-8}
	H	{2,9}
H	G	{0-8}

Figure 2. Example of filter-based event routing.

[1] [8] proposed that Siena can work with a cyclic network topology by first extracting a routing tree rooted at the origin of the message. However, the actual routing scheme is the same as with acyclic graph and is not further discussed in their papers. Therefore, we only consider acyclic topology for Siena in this paper.

[2] Another acyclic topology, i.e. hierarchical topology, was shown to perform worse than the peer-to-peer topology and therefore is not considered in this paper.

[3] Siena also proposed an alternative strategy of using *advertisements (by publishers)* to contain the transmission of subscriptions. Since this is an additional and nonstandard burden on a pub-sub service, we postpone discussion of it until Section IV.

in the event themselves. For example, in Figure 2, when a client publishes event *9* at server *A*, the message is matched four times at server *C*, *E*, *F*, and *G* before reaching destination *H*. Finally, routing load on the pub-sub servers is imbalanced: generally, the closer a server is to the center of the tree, the more events it receives and forwards. A server at the edge of the network only receives events of its interest and never routes for others.

B. Multicast-based event routing

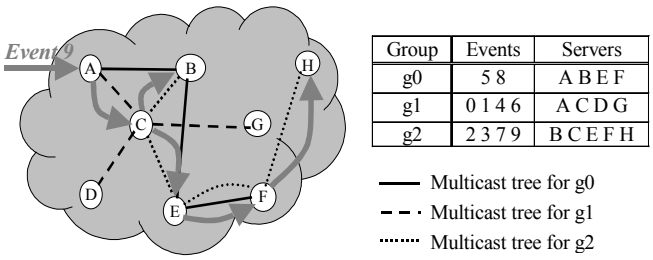
We use the approach in [25] as a representative for the multicast-based event routing approach. The process is illustrated in Figure 3. First, the event space is partitioned into a limited number of multicast groups. For each group, a multicast tree is built that spans all servers with subscription for any event in that group. When an event is published, it is mapped into a group and multicast on the corresponding tree to all group members.

Three major differences are seen in comparing Figure 3 to Figure 2. First, there are three routing trees and each tree only spans a subset of servers. As a result, the routing path can be shorter: event *9* no longer traverses server *G* to reach server *H*. Second, the routing table is simpler. It maps events to multicast groups, and the routing table is the same for every server. Finally, without fine-grained filtering, events can be sent to servers that are neither interested in the event nor needed to route it to its interested destinations. In Figure 3, event *9* is forwarded to server *B*, resulting in extraneous network traffic.

To reduce network wastage, the multicast-based approach uses intelligent clustering algorithms to partition multicast groups, with the goal of maximizing the commonality between member interests within each group. However, the effectiveness of clustering heavily depends on the locality property of events and subscriptions in the application. If the application data distribution does not lend itself to clustering opportunities, it is expected to be difficult to form only a few groups to match every server's interest with high accuracy. For example, when events and user interests are uniformly distributed, each of the 2^n possible multicast groups would be needed with roughly equal probability.

C. Discussion

The discussion above implies that filter-based event routing should achieve better network efficiency than the multicast-based approach. Its fine-grained filtering functionality naturally fits the highly diversified communication pattern in content-based pub-sub systems. However, the problems of subscription management, high processing load imbalance can be substantial impediments to the scalability of this scheme.



Group	Events	Servers
g0	5 8	A B E F
g1	0 1 4 6	A C D G
g2	2 3 7 9	B C E F H

Figure 3. Example of multicast-based event routing. Forgy's K-Means algorithm is used to cluster the events into three multicast groups.

We observe that partitions and topologies can be constructed to confine the information flooding and event routing to smaller scopes. The idea is to build multiple, smaller routing networks, and to guarantee that certain events are only routed through certain networks and a pub-sub server only joins a small subset of networks. In this way, events traverse fewer pub-sub servers, reducing processing and network load; also, pub-sub servers only need to maintain a subset of routing information, pertaining the events that may be routed on the networks in which it participates. Furthermore, dividing the routing load between multiple networks provides opportunities for better resilience and load balancing.

To meet the requirement above, the content space (or "event space") of the pub-sub system must be partitioned between the routing networks. The partitioning is critical to the effectiveness of the approach, because it determines the size and membership of the routing networks. A bad partitioning may result in all servers joining every network.

One candidate partitioning method is the content space clustering used in the multicast-based routing scheme discussed above. However, in this paper, we hope to develop a general event routing scheme whose success does not depend so much on specific pub-sub application characteristics. Therefore, instead of simply exploiting the clustering opportunity offered by the subscriptions and event patterns as they happen to be associated with servers, we explore the opportunity of actively creating content locality for the routing networks, by moving subscriptions and events around in constrained ways.

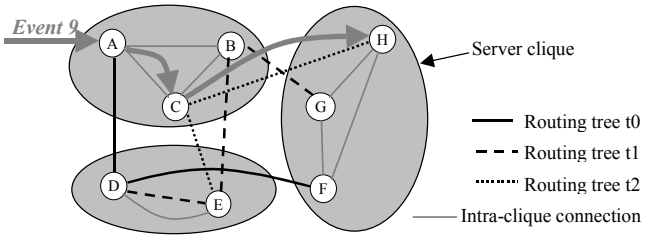
In the next section, we present the design of Kyra system developed based on these ideas.

III. KYRA DESIGN

The architecture of Kyra system consists of multiple event routing networks, with the following properties:

- Filtering-based event routing within each routing network generates low processing and network traffic load.
- Each pub-sub server manages only a small amount of routing information for the networks in which it participates.
- The event routing load is more evenly balanced across all pub-sub servers.

Kyra is designed with a two-level interconnection topology, as shown in Figure 4. At the bottom level, Kyra servers are organized into *server cliques* based on their network proximity. Servers in the same clique know about each other and communicate through unicast. At the second



Server	Server zone	Proxy subscriptions
A	0-3	{1,2}
B	4-6	-
C	7-9	{7,8}
D	0-4	{0,3}
E	5-9	{5,6}
F	0-3	{2}
G	4-6	{4-6}
H	7-9	{7,9}

Tree	Tree zone	Servers
t0	0-3	A D F
t1	4-6	B D E G
t2	7-9	C E H

Server	Tree	Routing table	
		Neighbor	Subscriptions
A	t0	D	{0,2,3}
B	t1	E	{5,6}
		G	{4-6}
C	t2	E	-
		H	{7,9}
D	t0	A	{1,2}
		F	{2}
	t1	E	{4-6}
E	t1	B	{4-6}
		D	-
	t2	C	{7-9}
F	t0	D	{0-3}
G	t1	G	{5,6}
H	t2	C	{7,8}

Figure 4. Example of Kyra network, with three server cliques and three routing trees.

level, multiple *routing trees* are built, each for routing a subset of events.

Corresponding to the two-level topology, the content space in the pub-sub system is partitioned at two levels: locally, it is partitioned between servers in the same clique. Each server is assigned a non-overlapping *zone* in the space, and becomes the *proxy server* for all subscriptions in the same clique that overlap with this zone, which are in turn called this server's *proxy subscriptions*. The *original servers* that receive subscriptions from end clients will forward the subscriptions to the appropriate proxy servers. We call this process *subscription movement*. Globally, the content space is partitioned between the routing trees. Each routing tree is assigned a non-overlapping content zone and used to route all events falling into its zone. The global partition is the same across all Kyra servers, while the local partitions are only visible inside each clique. Kyra servers join all the routing trees whose zone overlap with that of their own, and route on behalf of their proxy subscriptions. Each routing tree then becomes an independent filter-based routing network as described in Section 2. When an event is published, it is first forwarded to the server in the same clique whose content zone covers it, and then routed on the tree with covering zone.

In Figure 4, the pub-sub servers are organized into three server cliques, and three routing trees are built. The content zone of the servers and the routing trees are listed in the tables on the left. Each server maintains a routing table for each routing tree it joins, as shown on the right. When event *9* is published, it is first forwarded to server *C*, and then routed on tree *t2* to arrive at server *H*.

Three observations can be made from Figure 4. First, the routing tables are more concise than those in Figure 2, as each server only needs to know about a subset of subscriptions in the system. Second, routing trees in Figure 4 span fewer servers than those in Figure 3, due to the increased content locality on each server obtained from subscription movement. Finally, the routing path of event *9* traverses fewer immediate servers than in Figure 2 and Figure 3, resulting in less network traffic and processing load.

In the rest of this section, we present the design of Kyra in more detail.

A. Interconnection topology

In this paper, we use network latency to measure the distance between servers. We use the Hierarchical Agglomerate Clustering (HAC) algorithm [21] to cluster "close" servers into server cliques. The distance between two cliques is defined as the furthest distance between any pair of servers in the two cliques. The algorithm is presented in Figure 5. Two parameters are specified: the maximum distance between servers in the same clique, and the maximum number of servers in one clique. The output of the algorithm is a set of server cliques that satisfy both conditions.

For small-scale server cliques, the intra-clique topology is indeed a "clique": each server knows the address and content zone of all other servers in the clique; if a clique has too many servers, the Distributed Hash Table (DHT) techniques [24][27][31] can be used as an elegant solution for scalable subscription and event routing inside clique. Specifically, when there are k servers in the clique, a server only needs to know about O(log$k$) other servers and a message can be routed between any two servers in the clique within O(log$k$) steps. The content space partition in the clique can be directly used for dividing the index value space in DHT. For simplicity, we only experiment with the full-mesh topology within cliques in this paper.

In Kyra, routing trees are built as minimum spanning trees (MST) across all servers whose content zones overlap with that of the tree. The number of routing trees built, T, is related to server clique size as shown in Figure 6: if a clique has more than T servers, multiple servers have to join the same tree. As a result, subscription information for this tree is replicated on all these servers, reducing the effectiveness of local content space partitioning. On the other hand, increasing T to larger

```
Cluster_servercliques(maxDistance, maxNumServers) {
    foreach i in [1, …, n]      // n is the number of servers
        clique c_i ← server s_i;
    proximitymatrix_{i,j} = distance(s_i, s_j);
    while (number_of_cliques > 1) {
        foreach (c_i, c_j) with increasing proximitymatrix_{i,j} {
            if (proximitymatrix_{i,j} > maxDistance)
                return cliques;
            if (size(c_i) + size(c_j) ≤ maxNumServers) {
                merge(c_i, c_j);
                update_proximitymatrix;
                break;
            }}}
    return cliques;
}
```

Figure 5. Server clique clustering algorithm.

than the clique size cannot improve the effect of global space partitioning, because multiple trees will span the same set of servers. Therefore, in practice, we expect $T \sim \max\{k_i\}$ to be a reasonable configuration, where k_i is the number of servers in clique i.

B. Content space partition

The partitioning methodology in Kyra is simple: to partition the content space into non-overlapping continuous zones with balanced load.

We choose to partition the space into continuous zones for several reasons: first, such zones can be concisely described by their boundaries. This leads to low storage and communication cost to store the partition results and synchronize between servers. It is also easy to determine the membership of an event. Second, many pub-sub systems support subscriptions in the format of range queries, such as "price<5" or "5,000<volume<10,000". Compared to discrete partitions (such as by clustering individual event values), continuous partitions reduce the number of partitions with which such range subscriptions overlap. This is desirable because a subscription has to be replicated on all the servers and routing trees whose zones overlap with it. For the same reason, when the number of routing trees is different from the number of servers in the cliques, continuous partitions reduce the number of trees a server needs to join. Finally, continuous partitions make building more structured and scalable topology, such as DHT systems, possible.

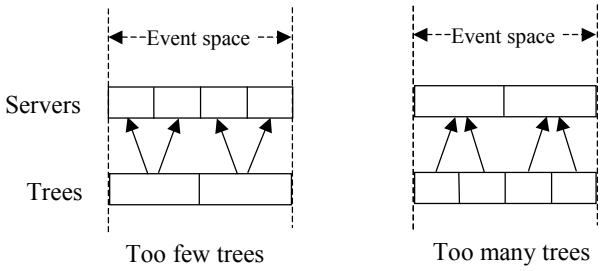


Figure 6. Relationship between number of routing trees and number of servers in a clique.

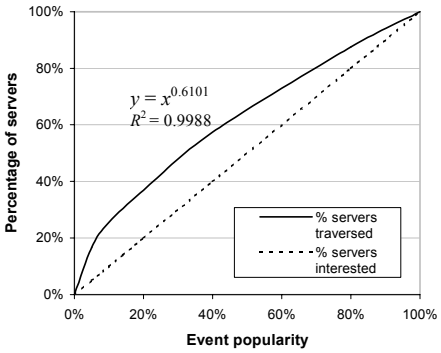


Figure 7. Percentage of servers an event traverses in a tree topology.

We define the *popularity* of an event to be the percentage of subscriptions interested in it, the *volume* of an event to be the frequency with which it is published, and the *weight* of an event to be the normalized resource consumption for processing the event. The load of a content zone is then computed as

$$workload_{zone} = \sum_{e \in zone} (popularity_e^{\alpha} \cdot volume_e \cdot weight_e)$$

The reason for using $popularity_e^{\alpha}$ rather than $popularity_e$ is the observation that when routed in a tree topology, an event is routed through more servers than the ones that are interested in it, and the routing load on all the servers traversed should be counted. In Figure 7, the horizontal axis shows the popularity of an event, and the solid curve plots the percentage of servers on the tree that the event is actually routed through. The curve is regressed to the power function presented, with R-square value of 0.9988. For reference, the dotted line shows the percentage of servers from the tree that actually interested in the event, which is in fact a 45-degree line. Figure 7 is based on experimental results with minimum spanning trees of randomly distributed servers, and the regression function is used to derive the α value of 0.6101 in our experiments.

The problem of partitioning a multi-dimensional space into continuous zones with balanced load has been well studied in many areas, such as parallel and distributed computing and database management [19][20][32]. Partitioning can be challenging since the nature of the event and subscription distributions can change with time, and the necessary information may have to be gathered and recomputed periodically. However, reasonably good partitioning results may be achieved based on coarse-grained load estimation and experience. In addition, we expect that in many pub-sub applications, partitioning along only a subset of dimensions, such as one or two of event attributes, will be sufficient to achieve the goals. Thus, we expect the partitioning process to scale well with both routing load and dimension of the content space. A specific partitioning algorithm dependents on application data types and properties, and is beyond the scope of this paper. Instead, we assume that such an algorithm is available and focus on the effectiveness of the overall routing scheme.

C. Subscription and publication

In Kyra, a subscription is submitted to a server close to the subscriber. Then, it is forwarded from the original server to one or more proxy servers, based on the content zones with which it overlaps. The subscription management process is shown in Figure 8.

Note that on the routing trees, events are routed for proxy subscriptions at each server, rather than its original subscriptions. Because the proxy subscriptions are wholly contained within the server's content zone, the content locality of proxy subscriptions on the server is expected to be higher than that of the original subscriptions.

Filter-based event routing is performed on each routing tree. At the same time, a received event is matched with the server's proxy subscriptions. Upon successful matches, the

```
receive_original_subscription(sub, client) {
    store_original_subscription(sub, client);
    Z = all_overlap_zones(local_partition, sub);
    foreach z in Z {
        server s = server_for_zone(z);
        subscription newsub = intersection(z, sub);
            send newsub to s;
    }}

receive_proxy_subscription(sub, from_server) {
     store_proxy_subscription(sub, from_server);
     Z = all_overlap_zones(global_partition, sub);
     foreach z in Z {
         tree t = tree_for_zone(z);
         subscription newsub = intersection(z, sub);
        advertise_subscription(t, newsub);
     }}
```

Figure 8. Subscription management in Kyra.

```
route_event(event, from_server) {
    t = tree_for_event(e);
    foreach neighbor n on tree t {
        if ((n != from_server) &&
            match(subscriptions_from(t, n), event))
            send event to n;
    }
    foreach server s in local_clique {
        if ((s != from_server) &&
            match(subscriptions_from(s), event)) {
            mark event as final notification;
            send event to s;
        }}}
```

Figure 9. Event routing in Kyra.

event is sent to the original servers of the matching subscriptions. The original server will notify the subscriber about the event, so that the process of subscription movement is transparent to end-users. The event routing process is shown in Figure 9.

In this paper, we assume centralized topology construction and content space partitioning algorithms. This provides simplicity and reduced communication overhead. We leave distributed algorithms as a topic for future work.

IV. EXPERIMENTAL METHODOLOGY

We now evaluate the performance of Kyra and other routing schemes with detailed simulations.

A. Event routing schemes

To understand how Kyra compares with existing approaches, we have also simulated a basic filter-based routing scheme (FBR) and a basic multicast-based routing scheme (MBR).

FBR is based on the Siena implementation described in [7], with the peer-to-peer minimum spanning tree topology. Some optimization techniques, such as use of advertisements (which are an additional burden on the system) and subscription summarization (whose success is application-dependent), are not included in FBR. These optimizations are applicable to Kyra as well, since it uses filter-based routing within each routing tree. In fact, we expect them to be more effective in Kyra, because of their lower implementation cost and the increased subscription locality in Kyra. By not including these optimizations, we can better compare the basic approaches.

MBR is based on the multicast-based routing scheme described in [25]. The Forgy's K-Means algorithm [21] is used for data clustering, as it was found to perform best among the clustering algorithms in [25]. An optimization technique is proposed in a companion paper [26] to dynamically switch to unicast if the event popularity is below a threshold. We do not include this optimization in MBR, so that we can clearly identify the effectiveness of the multicast-based approach.

We believe that FBR and MBR as we implement them represent the major properties of the two routing approaches, and the comparison provides us an opportunity to understand the trade-off of various routing schemes. To our knowledge, there has not been comprehensive comparison and evaluation of different event routing schemes for content-based pub-sub network.

Performance of three other basic routing schemes, unicast, broadcast and *ideal multicast*, are also presented as reference baselines. In ideal multicast, each event is sent to matching servers through IP multicast, assuming multicast trees exist for all possible matching subscription server sets.

B. Data model

A major challenge in pub-sub system evaluation is the lack of real-world workloads. For comprehensiveness, we experimented with four different distributions for events and subscriptions. These distributions are either prevalent in other information delivery applications [4] and/or have been used in the pub-sub literature [25][34][33]:

- Uniform distribution, in which both popularity and volume of events are uniformly randomly distributed.
- Zipf-uniform distribution, in which event popularity follows Zipf distribution [4], i.e. the number of subscriptions matching the ith most popular event is proportional to $i^{-\alpha}$, (with α here set to 1). The volume of events is uniformly randomly distributed.
- Multimodal distribution [25], in which both popularity and volume of events follow the same multivariate Gaussian distribution. In this case, more popular events are also published more often. In our experiments, five distribution peaks are randomly chosen in the content space, and the standard deviations are set to 1/4 of the average distance between peaks.
- Regional distribution [34], in which the probability that a subscription from server s_i matches an event from server s_j is set to:

$$p_{match}(s_i, s_j) = \frac{c}{distance\ (s_i, s_j)^{\gamma}}$$

where c is a normalizing factor. This distribution simulates the scenario that users are more interested in events close to them, such as local activities. In our experiments, γ is set to 1.

In all distributions, event weights are uniformly randomly assigned.

We define *average user interest rate* to be the probability that subscriptions on a server match a randomly chosen event. Three level of user interest rates, 1%, 10% and 50% are chosen to represent applications with user interests of high, medium and low selectivity.

Since our focus is not on partitioning algorithms themselves, we simplify partitioning by experimenting with a one-dimensional content space of integer values. We believe that the evaluations presented in this paper are not sensitive to the dimensionality of content space, and the results are of general importance.

C. Performance measures

We evaluate the performance of event routing schemes along the following dimensions:

- Storage and management cost, measured by the amount of routing information each pub-sub server maintains.
- Processing load. In FBR and Kyra, this is measured by the total number intermediate servers that perform content-based matching to route one event.
- Network performance, which includes:
  - Node stress: for every fixed number (1000 in our experiments) of randomly chosen events handled by the system, the number of messages that are received and sent by the average pub-sub server.
  - *Link stress*: for every fixed number (1000 in our experiments) of randomly chosen events handled by the system, the number of messages that are carried by the average underlying network link.
  - *Normalized resource usage (NRU)*. As in [10], we define *network resource usage* as the summation of underlying network link costs consumed in routing an event. Link latency is used as the cost measure. Since the ideal multicast scheme achieves the lower bound of network resource usage, normalized resource usage is defined as the ratio of network resource usage of an event routing scheme relative to this lower bound.

For MBR, only its network performance is studied. Its storage and processing cost depends on pub-sub data type and is not evaluated in this paper.

V. SIMULATION RESULTS

We developed a message-level, event-based simulator for evaluation. Our network topology is generated by GT-ITM [6] random graph generator using the transit-stub model. There are 20 transit domains with an average of 5 routers in each. Each transit router has an average of 3 stub domains attached, and each stub domain has an average of 8 routers. The link latencies are randomly chosen between 50-100ms for intra-transit domain links, 10-40ms for transit-stub links, and 1-5ms for intra-stub domain links. Altogether there are 2500 routers and 8938 links. 500 pub-sub servers are randomly attached to the routers by LAN links with 1ms latency. Events and subscriptions from the distributions described above are randomly assigned to the servers. IP multicast routing is simulated using a shortest path tree formed by the merger of the unicast routes from the source to each destination.

A. Kyra performance analysis

In this section, we analyze the performance of Kyra with varying configurations of server clique size and number of routing trees built. Since FBR can be seen as a special case of Kyra, with single-server cliques and one routing tree, our presentation discusses the results for Kyra relative to this case, allowing us to very naturally compare Kyra with FBR. Results for MBR and other routing schemes will be discussed in Section V. B. Due to space constraint, we present detailed results for only the Zipf-uniform data distribution here, leaving others to Section V. B.

1) Storage and management cost

Figure 10 shows the amount of routing information that a Kyra server maintains. The horizontal axis shows the clique size configuration, in terms of maximum intra-clique distance. The corresponding average and maximum numbers of servers in each clique are given in Table I. The vertical axis shows, using a log scale, the fraction of the total subscription information that the average server maintains. The four curves represent the cases of 1, 10, 20 and 50 routing trees. Figure 10 clearly demonstrates the effectiveness of Kyra in reducing the information load on each server. For example, with cliques of 200ms intra-clique distance and 20 routing trees, a Kyra server only knows about 1/10 of total subscriptions. Another observation is that both the server clique size and the number of routing trees have to be greater than 1 to effectively reduce the per-server information size. This confirms the importance of two-level content space partitioning and subscription movement: Without local content space partitioning and subscription movement, every server has to join all the routing trees; with only one routing tree, each server has to know about all subscriptions to correctly route for other nodes on the tree. Finally, Figure 10 shows that the server clique size and number of routing trees interleave in a fashion that validates

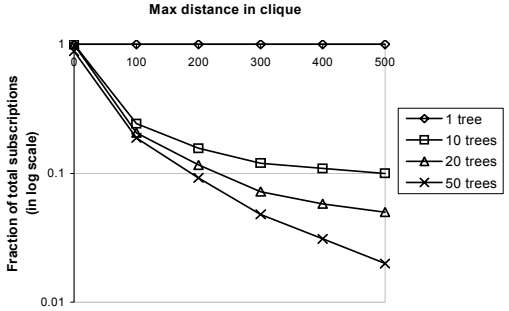


Figure 10. Amount of subscription information at each Kyra server.

TABLE I. KYRA SERVER CLIQUE SIZE.

Max intra-clique latency	0	100	200	300	400	500
Avg #servers/clique	1	6	13	32	125	500
Max #servers/clique	1	21	29	64	266	500

the setting of $T \sim \max\{k_i\}$ in Section 3. For example, when there are at most 21 servers per clique (max intra-clique latency is 100ms), the use of 50 trees results in almost no improvement over the case of 20 trees.

2)Processing load

In filter-based event routing, an event is repeatedly matched with remote subscriptions at intermediate pub-sub servers. Figure 11 plots the number of servers on which matching is performed in routing one event in Kyra. The three charts present the results with different user interest rates. All the curves converge at the two ends: the left end represents the case of FBR; the right end represents the extreme case of all servers organized into one clique. In this case, each event is matched once at the publishing server and sent directly to all matching servers.

Figure 11 shows that increasing clique size and increasing number of trees both effectively reduce the processing load in event routing. Differently from Figure 10, the top curves show that even with only one routing tree, increasing clique size leads to smaller matching load. This is because an event is matched only once in each clique. The saving is even more significant with high user interest rates in the figure. Higher user interest has the same effect of large clique size in this regard, because more users in the clique are interested and there is a larger space for improvement.

3)Network performance

a)Node stress

Figure 12 presents the average node stress of a Kyra server. The trend in each curve is similar to that in Figure 11: with larger server cliques and more routing trees, fewer intermediate servers are traversed on a routing path and the average node stress is reduced. However, the improvement diminishes with increasing user interest rates. The reason can be seen from : in the FBR approach, the fraction of uninterested servers an event traverses decreases as more users are interested in the event.

b)Link stress

From Figure 13, we can see that different configurations of Kyra can affect network link stress in three ways: first, with larger clique size, an event traverses fewer network links on the routing trees. This effect dominates when user interest level is as low as 1% and with large clique size. Second, the intra-clique unicast can result in high stress on links close to the unicast source. This effect is stronger with higher user interests, because more servers in the clique must be notified. Finally, multiple routing trees improve average link stress by distributing the network traffic over more network links. However, the magnitude of improvement is not as significant as we expected. We found that this is because of the low path diversity in the GT-ITM topology graph we used. For example, each stub domain is connected to a transit server through a single link. Building more routing trees cannot relieve the high stress on these links. We found that by setting 10% domains as multi-homed can reduce average link stress of Kyra by 10%. To gain a more comprehensive understanding of routing load on underlying network links, we plan to deploy experiments on larger network scale and take link bandwidth capacity into consideration.

c)Normalized Resource Usage

Figure 14 presents the NRU of Kyra. Larger server cliques almost always result in higher resource usage, mainly due to the network inefficiency of the intra-clique unicast. The inefficiency is severe with high user interest rates, in which case unicast communication comprises a high fraction of the total network traffic. The number of routing trees does not have much effect on NRU.

4)Kyra performance summary

We have evaluated Kyra using various metrics and the results are summarized in Table II. Briefly, with large server cliques and multiple routing trees, Kyra effectively reduces the storage, processing and network traffic load on each pub-sub server, compared to FBR. The intra-clique unicast communication results in increased network link stress and network resource usage. The inefficiency is more significant with larger server cliques and higher user interests, and independent of the number of routing trees. In general, this trade-off must be balanced by choosing configurations based on the characteristics of the pub-sub application.

Table III illustrates a set of concrete configurations that we use for Kyra in further experiments, chosen such that the NRU of Kyra is always smaller than 1.3 times that of FBR.

B. Comparison of Routing Approaches

In this section, we compare the network performance of various event routing schemes using four different pub-sub data distributions. We use 50 trees for MBR.

TABLE II. KYRA PERFORMANCE SUMMARY

	Storage and proc. load	Average node stress	Average link stress	NRU
Increasing clique size	↓	↓	↑ (w/ low interests) ↓ (w/ high interests)	↑
Increasing #trees	↓	↓	↓	–

TABLE III. KYRA CONFIGURATION AND PERFORMANCE COMPARISON WITH FBR.

Average interest level		1%	10%	50%
Kyra config.	Clique size	500	100	50
	#routing trees	50	20	10
Kyra/ FBR	Strorage	2%	20%	30%
	Processing load	6%	46%	35%
	Avg. node stress	30%	78%	92%
	Avg. link stress	62%	98%	116%
	NRU	126%	116%	111%

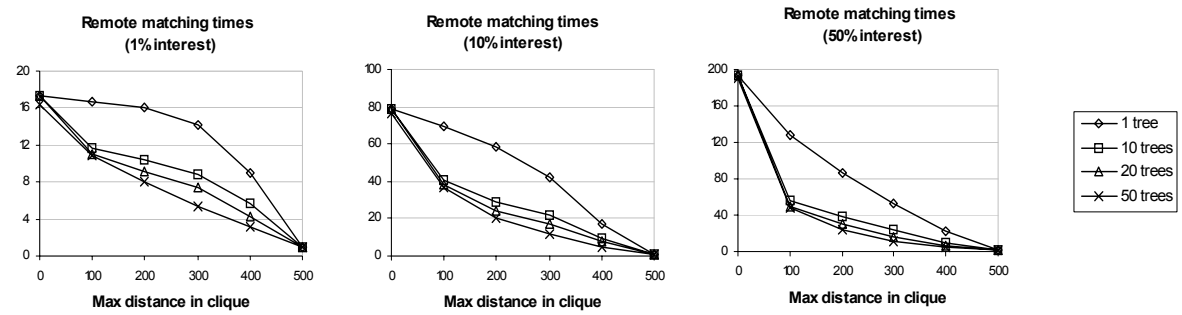


Figure 11. Routing processing load in Kyra

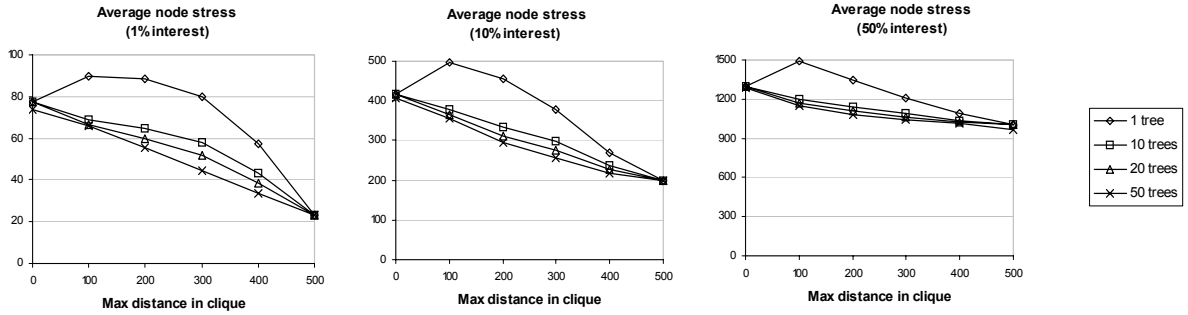


Figure 12. Average node stress in Kyra.

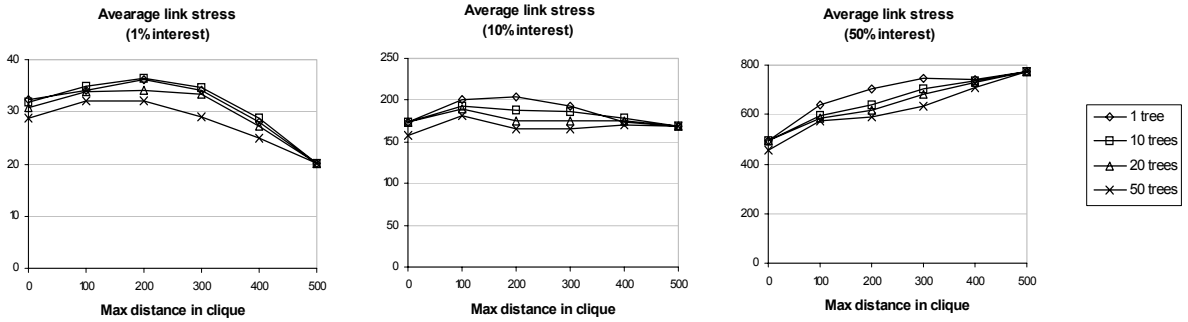


Figure 13. Average link stress in Kyra.

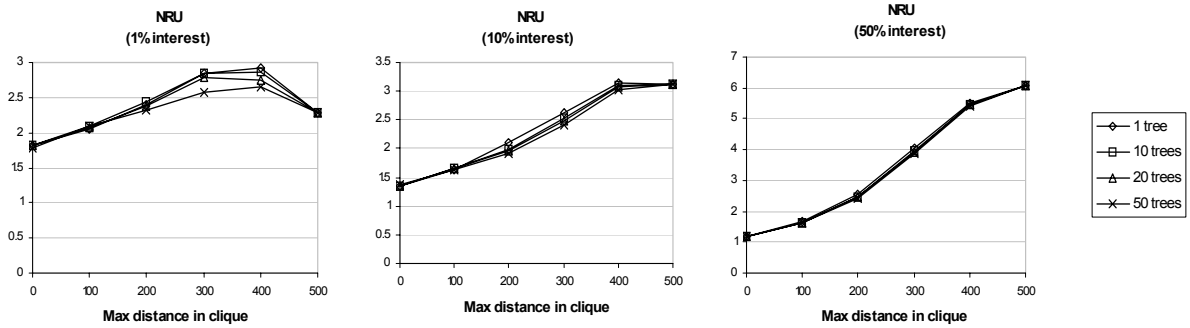


Figure 14. Normalized Resource Usage (NRU) in Kyra.

C. Comparison of Routing Approaches

Figure 15 compares the NRU of the FBR, Kyra, MBR, unicast and broadcast schemes. By definition, ideal multicast achieves NRU of 1. Overall, the results show that FBR and Kyra perform quite well under all circumstances. When user interests are highly selective, performance of Kyra is close to that of unicast and even better than FBR in some cases. In comparison, MBR is penalized for sending events to uninterested users. It performs worst with the Zipf-uniform distribution: the network waste mainly comes from multicasting the many "cold events" with few interested subscribers to the whole multicast group. The best distribution for MBR is the multimodal one, in which cold events are also published less often. In particular, when average user interest rate is 10% with the multimodal distribution, MBR achieves NRU 70% better than unicast, which confirms the results found in [25] under the same data distribution. With the regional distribution, MBR is penalized for sending events to uninterested users that are far away. When the average user interest rate is high enough, all the three routing schemes perform close to broadcast.

Table IV presents node stress and link stress of the three routing schemes. Due to space constraints, only the results for the zipf-uniform distribution and multi-modal distribution are presented. Under all circumstances, Kyra achieves the smallest average and maximum server node stress, and the savings are significant: for subscriptions with 1% selectivity, an average Kyra server experiences 1/4 network traffic load compared to an FBR server and only 1/25 of that of an MBR server. Even for the case of 50% interests, when the average node stress results are close, Kyra is more effective in distributing the network traffic across all servers and reducing the maximum node stress. In fact, Kyra always achieves the smallest average link stress except for the case of 50% interests, when FBR outperforms Kyra slightly. In this case here too, Kyra effectively minimizes the maximum link stress compared to FBR.

D. Load balance

Load balance is an important factor in pub-sub networks, as any overloaded server or network link may degrade the total system performance and limit system scalability. Table III shows that there is still a large gap between the average and maximum node stress and link stress in Kyra, which we should address. In this paper, we mainly focus on balancing node stress on pub-sub servers. Because link stress is affected by network resource dimensioning and provisioning strategies, it is left as future work.

To build a more load-balanced Kyra, we developed a modified version of Kruskal's MST algorithm [11] for building routing trees: at each step of adding an overlay connection into the routing tree, we first find the M shortest connections that do not add loops into the tree; these connections are then ranked by the maximum degree of their two end nodes. The connection with the lowest maximum degree is added into the tree. We call M the *balance factor*.

When M=1, the algorithm is Kruskal's algorithm; when M equals to the total number of valid connections, the algorithm aims at pure load balancing. Figure 16 shows the cumulative distribution of node stress in FBR, MBR, basic Kyra and balanced Kyra with balance factor of 100. The horizontal axis represents a given value of node stress, and the vertical axis

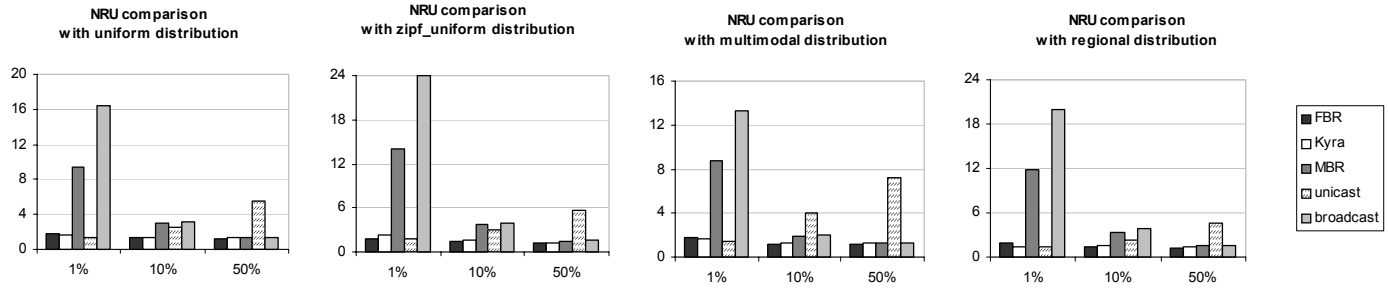


Figure 15. NRU comparison

TABLE IV. BANDWIDTH AND LINK STRESS COMPARISON, WITH ZIPF-UNIFORM DISTRIBUTION.

Average user interest rate		1%			10%			50%		
Event routing scheme		FBR	MBR	Kyra	FBR	MBR	Kyra	FBR	MBR	Kyra
Zipf-uniform distribution	Avg. node stress	77	559	23	416	1791	326	1298	1781	1199
	Max node stress	1828	2154	557	4949	9248	1626	8390	9208	3820
	Avg. link stress	32	286	20	173	759	171	491	755	548
	Max link stress	654	2510	560	2475	7338	1743	5211	5872	4606
Multi-modal distribution	Avg. node stress	132	693	31	873	1736	565	1742	1939	1574
	Max node stress	2758	2793	72	7318	10046	2136	10406	10723	4253
	Avg. link stress	61	359	27	370	777	297	668	938	746
	Max link stress	1512	4716	189	4967	7648	2412	7695	7933	5070

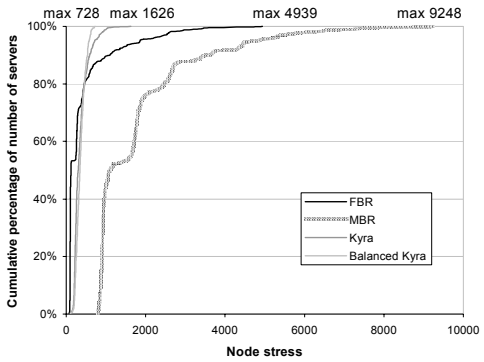


Figure 16. Cumulative distribution of node stress.

represents the percentage of servers with stress less than this value. It is evident from the graph that FBR and the two versions of Kyra perform much better than MBR: the 80-th percentile is about 500 for these schemes and 2500 for MBR. FBR has a heavy tail that ends at 4939, while the maximum stress is 1626 in Kyra and only 728 in balanced Kyra.

VI. Related Work

Much research exists on the distributed pub-sub systems. The architecture designs include Siena [7][8], Gryphon [2][22], JEDI [12], Rebeca [15], Elvin [28], Ready [17], and Herald [5]. Most of these systems adopt the filter-based routing approach. In JEDI, a hierarchical interconnection topology is proposed in which a server is only informed of subscriptions from servers in its sub-tree. Events are always forwarded up the hierarchy regardless of the interests in other parts of the network. [7] shows that the performance of such hierarchical scheme is inferior to the peer-to-peer topology we discussed in this paper. In Gryphon, a link-matching algorithm is designed to partially match an event at each step in filter-based routing, in order to determine the directions in which to send the event. The Elvin system proposes a technique called "quenching", in which publishers are aware of all subscriptions so that they only publish the events that have at least some matching subscribers. [29] has a different problem statement. It assumes that only a small number of content filters are available, and studies the problem of strategically placing the filters on a multicast tree topology, to minimize total network traffic or event delivery delay.

The problem of delivering an event message to a group of interested users through a service network is similar to the traditional multicast problem (except that in traditional multicast the addresses of the destination nodes are known while in pub-sub systems they have to be determined via content-based matching). IP multicast has been proposed since [13] but has not been fully deployed due to its inherent scalability problems. Recently, much effort has been put to move the multicast functionality to the application layer, at end hosts [10][9]. Application-layer multicast is expected to scale better than IP multicast, mainly because an end host only needs to perform complex processing for a small number of groups that it participates. This philosophy of confining the expensive routing functionality to only a subset of participants is similar to our idea of constructing multiple small routing networks in Kyra.

How to efficiently match an event against a large number of subscriptions is another important problem in pub-sub system design. The matching problem has been studied for various data types and event schemes [2][3][18][30]. In this paper, we have assumed that a suitable matching algorithm is available, and have focused on the problem of routing events (based on matching results, as appropriate).

A review of the various properties of pub-sub systems can be found in [14].

VII. Conclusion and Future Work

We have designed and evaluated Kyra, an event routing scheme for content-based publish-subscribe service networks. Our findings can be broadly summarized as follows:

- The two-level hierarchical topology and the content space partitioning techniques in Kyra effectively partition a pub-sub network into multiple smaller routing networks. Event routing within each routing network generates significantly lower storage, processing and network traffic load, compared to routing in the global network.
- The reduced scope of filter-based routing in Kyra can lead to inefficient network resource usage in unicast communication in server cliques. However, detailed simulations showed that the penalty is small compared to the magnitude of the savings in implementation cost and routing load. This is because unicast communication is only used between servers that are close to one another in the network, i.e. in the same clique. The trade-off between storage, processing cost and node/link stress on the one hand and the network resource usage measure on the other can be managed by configuring Kyra's main parameters (clique size and number of routing trees) based on application-specific preferences.
- Unlike other systems, Kyra is effective in balancing routing load over all pub-sub servers.

Because of its efficiency and balance along various resource usage criteria, we expect Kyra to gracefully scale to large pub-sub systems.

There are many areas of future work. In addition to those already mentioned in the paper, an interesting direction is to investigate the possibility of combining our subscription movement technique with multicast-based routing (rather than filter-based routing, which we have used within the routing trees here). We expect that the increased locality of subscription distributions would improve the quality of the multicast groups formed; however, the overall network resource usage may still be less efficient than filter-based routing.

A key (and complementary) direction of our current work is in replacing filter-based routing with an approach that decouples the matching and routing steps in a content-based pub-sub service network. The idea is to first match event with

global subscriptions at publisher, and obtains a list of destination servers interested in the event. This destination list is then attached in the message header as the event is forwarded; a pub-sub server receiving the event will dynamically figure out the next hops for the event based on the destination list. Our preliminary analysis and experimental results show that this *match-early* approach offers high routing efficiency and flexibility: because routing decision is made for each individual event on the fly, the approach naturally fits the highly diversified communication pattern in pub-sub systems, and can easily adjust for network conditions and application preferences.

REFERENCES

[1] http://mobile.yahoo.com/wireless/alert

[2] M. K. Aguilera, R. E. Strom, D. C. Sturman, M.Astley, and T. D. Chandra, "Matching events in a content-based subscription system," In *Eighteenth ACM Symposium on Principles of Distributed Computing,* 1999.

[3] M. Altinel and M. Franklin, "Efficient Filtering of XML Documents for Selective Dissemination of Information," In *VLDB Journal*, pp. 53-64, 2000.

[4] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and zipf-like distributions: Evidence and implications," In *Proc. of IEEE INFOCOM*, 1999.

[5] L. F. Cabrera, M. B. Jones and M. Theimer, "Herald: Achieving a Global Event Notification Service," In *Proc. of the Eighth Workshop on Hot Topics in Operating Systems (HotOS-VIII)*, May 2001.

[6] K. Calvert, E. Zegura, and S. Bhattacharjee. "How to Model an Internet-work". In *Proceedings of IEEE Infocom*, 1996.

[7] A. Carzaniga, "Architectures for an Event Notification Service Scalable to Wide-area Networks". PhD Thesis. Politecnico di Milano. December, 1998.

[8] A. Carzaniga, D. Rosenblum, and A. Wolf, "Design and evaluation of a wide-area event notification service," In *ACM Transactions on Computer Systems*, 2001.

[9] M. Castro, P. Druschel, A.M. Kermarrec and A. Rowstron, "SCRIBE: A large-scale and decentralized application-level multicast infrastructure," *IEEE Journal on Selected Areas in communications* (JSAC), 2002.

[10] Y. H. Chu, S. G. Rao and H. Zhang, "A case for end system multicast," in *ACM SIGMETRICS*, 2000.

[11] T. Corman, C. Leiserson, R. Rivest and C. Stein, "Introduction to Algorithms", MIT Press, 2001.

[12] G. Cugola, E. Di Nitto, A. Fuggetta, "The JEDI Event-based Infrastructure and its Application to the Development of the OPSS WFMS", in *Proc. Of IEEE Transactions on Software Engineering*, 2001

[13] S. Deering, "Multicast routing in internetworks and extended lans," in *Proceedings of the ACM SIGCOMM*, pp. 55-64, Stanford, CA, August 1988.

[14] P. Eugster, P. Felber, R. Guerraoui, A. Kermarrec. "The Many Faces of Publish/Subscribe," *Microsoft Research Technical Report EPFL, DSC ID*, 2000.

[15] L. Fiege, G. Mühl, F. Gärtner. "A Modular Approach to Building Event-Based Systems", In ACM Symposium on Applied Computing, 2002

[16] Z. Ge, M. Adler, J. Kurose, D. Towsley and Steve Zabele, "Channelization problem in large scale data dissemination," Technical report, University of Massachusetts at Amherst, 2001.

[17] R. Gruber, B. Krishnamurthy, and E. Panagos. "The architecture of the READY event notification service". In *Proc. of the19th Middleware Workshop*, 1999

[18] E. N. Hanson, C. Carnes, L. Huang, M. Konyala, "Filtering Algorithms and Implementations for Very Fast Publish/Subscribe Systems," In *Proc. of ACM SIGMOD*, pages 115-126, 2001.

[19] R. v. Hanxleden and L. R. Scott, "Load balancing on message passing architectures", In *Journal of Parallel and Distributed Computing*, 13 (1991).

[20] B. Hendrickson and R. Leland. "Multidimensional spectral load balancing." Technical Report SAND93-0074, Sandia National Laboratories, January 1993

[21] A.K. Jain, M. N. Murty, and P.J. Flynn, "Data clustering: a review." In *ACM Computing Surveys* 31, 3 (1999), 264--323.

[22] L. Opyrchal, M. Astley, Joshua S. Auerbach, G. Banavar, R. E. Strom, and D. C. Sturman, "Exploiting IP Multicast in Content-Based Publish-Subscribe Systems," In *Proc. of Middleware 2000*.

[23] J. Pereira, F. Fabret, F. Llirbat and D. Shasha, "Efficient matching for web-based publish/subscribe systems," *In Proc. of the Fifth IFCIS,* 2000.

[24] S. Ratnasamy, P. Francis, M. Handley, R. Karp, S. Shenker, "A scalable content-addressable network," In *Proc. ACM SIGCOMM*, pp. 161-172, 2001.

[25] A. Riabov, Z. Liu, J. Wolf, P. Yu and L. Zhang, "Clustering Algorithms for content-based publication-subscription systems," In *ICDCS* 2002.

[26] A. Riabov, Z. Liu, J. Wolf, P. Yu and L. Zhang, "New Algorithms for content-based publication-subscription systems", In ICDCS 2003.

[27] A. Rowstron and P. Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pp. 329-350, November, 2001.

[28] B. Segall, D. Arnold, J. Boot, M. Henderson and T. Phelps, "Content Based Routing with Elvin4," In Proc. of *AUUG2K*, 2000.

[29] R. Shah, R. Jain, F. Anjum, "Efficient Dissemination of Personalized Information Using Content-Based Multicast," In *IEEE Infocom*, 2002.

[30] C. Snoeren, K. Conley, and D. K. Gifford. "Mesh based content routing using XML," In *SOSP,* 2001.

[31] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. "Chord: A scalable peer-to-peer lookup service for internet applications," In *Proc. of ACM SIGCOMM,* 2001.

[32] G. Vanecek, "Brep-index: a multidimensional space partitioning tree." In *International J. of Computation Geometry Application*, 1(3), 243--261, 1991.

[33] Y. Wang, L. Qiu, D. Achlioptas, G. Das, P. Larson, and H. Wang, "Subscription Partitioning and Routing in Content-based Publish/Subscribe Networks," In *16th International Symposium on DIStributed Computing*, 2002.

[34] T. Wong, R. Katz, and S. McCanne. "An evaluation of preference clustering in largescale multicast applications," In *Proc. IEEE INFOCOM*, March, 2000.

[35] H. Yongqiang, G. M. Hector, "Publish/Subscribe in a Mobile Environment," *MobiDE* 2001.