HYPER: A Hybrid Approach to Efficient Content-based Publish/Subscribe

Rongmei Zhang and Y. Charlie Hu
Purdue University
West Lafayette, IN 47907
{rongmei, ychu}@purdue.edu

Abstract

Publish/Subscribe (pub/sub) is an important paradigm for distributed content delivery. Traditionally, there have been two approaches to supporting pub/sub service: subject-based and content-based. Content-based pub/sub allows fine-grained expressiveness, and thus is a more attractive solution for content dissemination. However, the performance of a content-based pub/sub network is bounded by the expensive matching cost of content messages. In this paper, we propose a hybrid approach capable of minimizing both the matching and forwarding overhead within the pub/sub network and the delay experienced by clients receiving the content. The hybrid approach aims to eliminate redundant matching and forwarding inside the pub/sub network. In particular, it identifies a number of virtual groups *by exploring common subscription interests among clients, and messages for each virtual group are only matched once at the group entry point. In addition, for each virtual group, the content delivery tree embedded in the underlying pub/sub network can benefit from shortcutting forwarding-only paths. Simulations have shown that the hybrid approach is highly effective in improving the service efficiency and quality of a content-based pub/sub system.*

1 Introduction

Publish/Subscribe (pub/sub) systems [11] provide selective content distribution services. Clients specify content of interests through "subscriptions"; any content matching the subscriptions is delivered to the client when available. Pub/sub is an important building block of various distributed applications such as event notification systems and resource discovery services.

There have been two basic approaches to pub/sub services. In *subject-based* pub/sub systems, content is labeled with pre-defined "subjects", to which clients subscribe. Content for each subject is usually disseminated by multicast. The other approach allows fine-grained content distribution. Both subscriptions and content are specified with respect to *attributes*; content delivery is performed based on matching the attributes. These systems are called *content-based* pub/sub systems. A classic example is a stock trade system, where content can be described by three attributes: $(issue, price, volume)$ and a subscription can be specified as a disjunction of *predicates*, e.g., $(issue = Google, price < 100, volume > 1000)$.

A content-based pub/sub system usually consists of a network of pub/sub servers that manage client subscriptions and forward content to interested clients. Clients attach to these servers in order to send or receive content messages. For the efficiency of subscription propagation and content delivery, the pub/sub network usually has a tree topology [4, 7].

Content-based pub/sub systems fall into two categories. The first category groups similar content into clusters, and each cluster is usually implemented by a multicast group, as in subject-based pub/sub systems. Previous work have studied various content clustering schemes (e.g., [13]) and multicast channel assignment algorithms (e.g., [1]). Clients subscribe to all clusters that overlap, possibly partially, with their interests, and thus may receive unwanted content. Therefore, the fine-grained expressiveness of content-based pub/sub systems has to be sacrificed to certain degrees in this "clustering" approach. The second category achieves precise content delivery: content is only delivered to the clients whose subscriptions match the attribute descriptions of the content [7, 4]. This "exact-match" approach retains the desirable features of expressiveness and flexibility in content-based pub/sub systems, at the expense of potentially higher state maintenance and processing cost.

Previous work on content-based pub/sub has focused on the pub/sub system architecture and content matching. Designing efficient content matching algorithms is a key challenge in content-based pub/sub systems [2, 8], especially for the "exact-match" approach. Since content messages are not given explicit destination addresses, the pub/sub network is responsible for determining the forwarding paths for each message. At each step, this process amounts to evaluating the predicates associated with the message, matching them against the subscription table, and deciding the next-step servers to forward the message. If the content space is defined by a large number of complex predicates, message matching can be a significant source of cost at the pub/sub servers. The situation is exaggerated when there are large volumes of published content so that the frequency of matching at each pub/sub server is extremely high, or when the content is very popular so that matching has to be performed by a large number of pub/sub servers. In [8], a matching algorithm aimed at fast message forwarding is proposed; it has been shown to achieve matching time of from a few milliseconds to tens of milliseconds for a content space with millions of attribute constraints. The results imply that matching time is significant compared with the network delay of message forwarding, which usually is also on the magnitude of milliseconds. Therefore, from the perspective of the pub/sub system, content matching can be a defining factor for the throughput of the service network. Meanwhile, content matching can also be an important contributor to the delay experienced by clients receiving the content.

In both Gryphon [4] and Siena [7], matching is conducted at each step of forwarding the content message from the publisher towards the subscribers, thus possibly reaching all the servers in the pub/sub network. More recently, Kyra [6] proposes to create multiple smaller pub/sub networks based on content clustering so that each network is in charge of a subset of the content space. In Kyra, each instance of content distribution only involves the one pub/sub network associated with the content; however, it may still result in matching at all the servers in the corresponding cluster. In all of these systems, expensive matching operations may be invoked unnecessarily many times, when a large volume of messages are published with the same attribute descriptions, e.g., online price monitoring of a particular issue of stock.

In this paper, we propose a hybrid pub/sub scheme that minimizes both the amount of matchings inside the pub/sub network and the delay to receive subscribed content at clients. In particular, the pub/sub network dynamically identifies a number of *virtual groups* based on common subscriptions. For each of such groups, content messages only need to be matched once at the entry point of the group, and subsequently forwarded to the other group members without any more matchings. Similar to subject-based pub/sub systems, each group uses a delivery tree for message dissemination. However, the groups are defined on-demand according to client subscriptions other than being pre-defined. Moreover, these virtual groups co-exist with the original content-based pub/sub system, and their correspondent delivery trees are embedded inside the default pub/sub network. The virtual groups collectively cover a subset (possibly all) of the content space, and the content outside their definition scope is matched and forwarded by the default pub/sub network as usual.

In addition, for each virtual group, the delivery tree can be optimized to further reduce the forwarding overhead in the pub/sub system. The rationale is similar to that of tunneling in IP multicast [19]. If a particular path in the delivery tree passes through one or more servers without branching off, this path can be shortcut to bypass those intermediate servers. In this way, message forwarding can be achieved by traversing fewer servers. The benefit of shortcutting is also two-fold: First, the efficiency of content delivery is improved for both the pub/sub servers and the underlying network. Secondly, clients can receive the content with reduced delay.

The rest of the paper is organized as follows. We first give an overview of the hybrid pub/sub architecture, and then discuss the techniques of on-demand grouping and tunneling respectively, followed by results from extensive evaluations of the hybrid approach. Finally we draw conclusions.

2 Design Overview

In this section, we describe the design rationale of the hybrid pub/sub architecture.

2.1 System Model

For clarity of presentation, we model the content space Ω as a multi-dimensional space, with each dimension representing an attribute. A content message is uniquely described as a point in such a space, while a subscription is a defined as a rectangle. A published content message matches a subscription if it is within its defining rectangle. This data model is consistent with the assumptions made by existing pub/sub systems in the literature [13, 14]. In practice, the content space can be refined by designating a name, type and value range to each dimension. In the examples we will show in this section, the content space is a 1-D line and each subscription is described as a range.

A content-based network is an overlay of pub/sub servers. We assume that the pub/sub network is organized as a single-source tree N, as show in Fig. 1. The tree root R is the source of content. Clients (subscribers) are attached to the tree leaves $L_0, \ldots, Ln$, which serve as their pub/sub proxies. Each tree node maintains a subscription table, which records the subscriptions from each downstream tree node or clients. Subscription tables are created as client subscriptions are aggregated and propagated from the leaves up the tree. This process is similar to the subscription advertisement in Siena [7]. Here we focus on the pub/sub tree N and do not consider data dissemination from the leaf servers to the clients, i.e., the actual subscribers. When a content message is published, it is forwarded down the tree N as being matching against the subscription tables at each step. In the example by Fig. 1, if a message is described by the value of 3, it will be forwarded first to nodes K_0 and K_1, and then to leaves L_0 and L_2. We do not assume any specific matching algorithm in this paper.

2.2 Content-based Pub/Sub with Virtual Groups

In the hybrid pub/sub system, a virtual group G_i is identified as a sub-region of the content space shared by the same subset of subscribers, i.e., the same leaf nodes in our model.

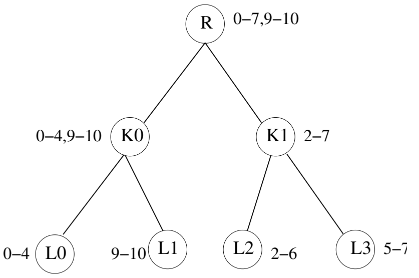

Figure 1. Pub/sub tree

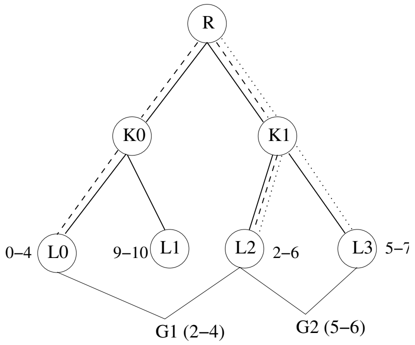

Figure 2. Virtual groups in the pub/sub tree

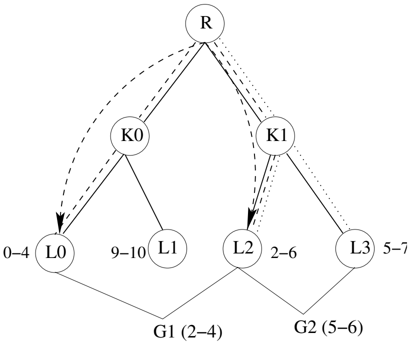

Figure 3. Virtual groups with shortcuts

For the pub/sub tree in Fig. 1, there exist two virtual groups G_1 and G_2, as highlighted by dash lines. Each leaf node L_j can belong to multiple of such virtual groups. However, the definition of each group in the content space is unique and non-overlapping with other groups. We discuss how to find these virtual groups in the next section.

Each virtual group G_i induces a sub-tree T_i embedded in the original pub/sub tree N. The sub-tree consists of the leaf servers $L_{i_0}, \ldots, L_{i_n}$ in the virtual group and all the intermediate tree nodes on the paths from these leaves to the tree root. For virtual group G_1 in Fig. 2, its corresponding sub-tree includes the root R, the two internal nodes K_0 and K_1 and leaves L_0 and L_1. As a content message is published at the root R, it is first matched against the virtual groups. If the message belongs to one of these groups, i.e., G_i, it is forwarded down the associated sub-tree T_i without being semantically matched at any other node. Otherwise, the message is delivered by the default pub/sub tree N by matching and forwarding at each step. It is easy to see that for an individual message, the number of matching operations saved by using the virtual group G_i is determined by size of the corresponding sub-tree T_i. More precisely, the saving per message is equal to the number of internal nodes F_i in the sub-tree. Therefore, generally speaking, the more popular the virtual group G_i, i.e., the more leaf servers that the group spans, the more beneficial to use it for content delivery. In addition, the value of F_i, and thus the matching saving, is also determined by the specific topology of an individual sub-tree T_i.

In a real-world situation, the density of published messages in the content space can vary, A second factor in determining the potential benefit of creating a virtual group G_i is the amount of messages C_i covered by the group. The more messages to be delivered by the virtual group, the more savings in terms of matching operations. Overall, the benefit of a virtual group in terms of saved matchings can be formulated as: $F_i * C_i$.

2.3 Efficient Forwarding in Virtual Groups

The benefit index of a virtual group can be improved by optimizing the sub-tree structure. The delivery tree T_i for each virtual group G_i is obtained from the original pub/sub tree N. As a result, there may exist redundant forwarding hops in the induced tree T_i as messages are passed through a sequence of server nodes without being replicated. In this case, the sequence of simply-forwarding hops can be replaced by a single hop from the entry node to the exit node where the forwarding path branches off. The virtual group G_1 in Fig. 2 has two shortcuts, as shown in Fig. 3. This type of shortcutting can further reduce the cost of the pub/sub system by eliminating unnecessary messaging between the pub/sub servers. The benefit of a shortcut is determined by the number of forwarding hops being bypassed by the shortcut.

Shortcuts may exist in any virtual group tree. However, it is likely to find more shortcuts in sparse virtual groups shared by only a few leaf servers. Therefore, although the benefit from saved matching operations may be lower for sparser groups, their overall benefit index can be improved by applying shortcutting. Similar to the achievable matching saving by a virtual group G_i, the benefit of shortcutting is also tied with the specific topology of the sub-tree T_i.

An extreme situation of shortcutting is a virtual group with a single leaf, and thus a shortcut can be established from the root to the leaf, bypassing all internal nodes. In this case, the shortcut creates a unicast path.

2.4 Summary

The hybrid pub/sub architecture combines the strengths of both subject-based and content-based systems. It is built based on a content-based pub/sub network and retains the expressiveness and flexibility of fine-grained content delivery. Meanwhile, virtual groups are extracted to exploit shared subscription interests among clients. Virtual groups are used as expressways for disseminating content of common inter-

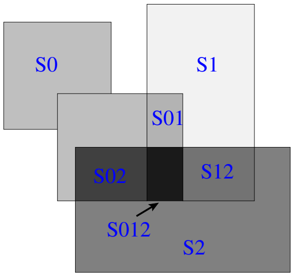

Figure 4. Subscription rectangles at leaf servers

ests. These expressways can be further expedited by adding shortcuts between branching points in the delivery tree.

Virtual groups enhanced with shortcutting is potentially beneficial for various group sizes. A virtual group can overlap with the underlying content-based pub/sub tree; in this case, it serves as an express broadcast channel. On the other hand, a virtual group can consist of a single shortcut from the root to a leaf server and is equivalent of unicast. Nevertheless, virtual groups are more likely to include subsets of the pub/sub network and act as optimized multicast channels.

So far we have assumed a single-sourced tree for the content-based pub/sub system. However, the same approach is applicable in other pub/sub network topologies. For example, in a content-based network like Siena [7], there exists a delivery tree for each publishing server; each of these trees can benefit from exploring virtual groups and shortcutting, as we have discussed in this section.

3 Implementing Virtual Groups

This section discusses the techniques related to creating virtual groups in a content-based pub/sub network.

3.1 Virtual Group Identification

A virtual group G_i is defined by the subset of the multidimensional content space shared by the bottom-level group members: $L_{i_0}, \ldots, L_{i_n}$. Since subscriptions are described by rectangles, the aggregated subscriptions S_j at a leaf pub/sub server L_j can be described as a disjunction of individual rectangles. Fig. 4 shows an example of three leaf servers, with their respective subscription rectangles highlighted by different shades. Each leaf server can have many, possibly overlapping rectangles. In this simple example, the aggregated subscription S_0 at leaf server L_0 consists of two overlapping rectangles. Identifying virtual groups amounts to finding the intersections of the aggregated subscriptions from different leaf servers. There exist four distinct intersections between the three leaf servers in Fig. 4, each labeled with the intersecting servers.

It is straightforward to produce the intersection of two rectangles. However, finding all possible virtual groups requires testing and reporting intersections from any combination of group members, i.e., 2^n combinations in total. Therefore, a brute-force solution is not practical.

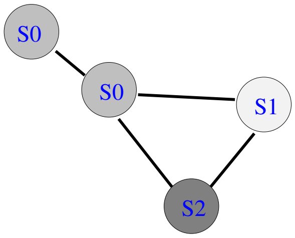

Figure 5. Colored intersection graph

Theorem: The virtual group identification problem is NP-complete.

Proof: Given the n set of rectangles, one from each leaf server, an intersection graph V can be generated by denoting each rectangle with a node and connecting any two intersecting rectangles with an edge. Furthermore, we assume that each set is characterized with a distinct color code (shown as different shadings in our example), and each node in the intersection graph has the same color as the set associated with the respective rectangle. Fig. 5 depicts the colored intersection graph corresponding to the subscription rectangles in Fig. 4. In this way, an intersection between k rectangles is translated into a k-size clique in the intersection graph V. The number of distinct colors ($\leq k$) in each clique represents the number of intersecting sets. There exist five cliques in total in Fig. 5. However, only four have at least two different colors (shades), while the two overlapping subscription rectangles at leaf server L_0 corresponds to the fifth clique. The problem of virtual group identification is equivalent of producing any intersection between k sets, where $k \in [2, n]$, and thus is reduced to finding all cliques with at least two different colors in the intersection graph. A special case of this problem is when each set contains only one rectangle. If we could solve this special case, we would be able to answer the following question, e.g., by scanning all the identified cliques: whether there exists a clique of size k in the graph V. However, the latter is a well-known NP-complete problem called the **clique problem** [9]. Therefore, the virtual group identification problem is NP-complete. □

3.1.1 Grid-based Grouping

In this paper, we adopt a grid-based scheme for virtual group identification. A similar approach was used to cluster similar content in [13]. First the content space Ω is partitioned into a regular grid. Each leaf server L_j marks the grid cells based on its local aggregation of subscriptions, i.e., whether the local aggregated subscription covers a cell a_k.

Next, a subscription chart is generated by putting together all the marked grids, with each cell a_k annotated with all the leaf servers interested in it. Finally, virtual groups can be picked out by grouping all the grid cells that feature the same set of subscribing leaves. If both cell a_1 plus cell a_2, and no other cells, are subscribed by the same set of leaves, a_1 and a_2 make a virtual group. In this way, the total number of potential virtual groups is bounded by the number of cells in the grid, i.e., in the worst case, there can exist as many virtual group candidates as the grid cells. In practice, only the most beneficial candidates are selected to form virtual groups.

A virtual group can be characterized by both its defining grid cells and the associated leaf servers. As discussed previously, the potential benefit of a virtual group is determined by both the group size and the content volume delivered by the group. In addition to the distribution of subscriptions, information on message density can also be collected based on the same grid, i.e., each cell can be marked with the amount of relevant content messages. In this way, the weight of a virtual group can be simply calculated as the sum of the message density of each associated cell. In our model, message density can be recorded by the root R, where all content messages are originated.

The partition of the grid is mandated by the specific pub/sub application and its content space. In the stock exchange example, the dimension corresponding to the price or the volume can be divided into unit lengths based on the numerical precision of the application. Since there can be a large number of cells in the grid for some applications, the grid can be pre-processed using the message density information [13]: only those cells with message density higher than a certain threshold are selected for consideration in virtual group identification. In applications with highly skewed publication density, this process can eliminate a potentially large number of cells with zero or very low density. A second potential solution is to use coarser partitioning if clients can tolerate unwanted deliveries to certain extent, e.g., by deploying local content filters. In this case, each leaf server can mark the grid cells based on whether the local aggregated subscription partially overlaps with a cell (instead of containing the cell).

3.2 Virtual Group Setup

This subsection proposes a distributed protocol for virtual group construction based on the algorithm described above.

We assume that there exists a protocol coordinator in the pub/sub network, and it can be the tree root R or any predesignated entity. First, each leaf server updates the coordinator with its local subscriptions, i.e., as a marked grid. After aggregating all subscriptions, the coordinator identifies all candidate virtual groups, each with a benefit value as formulated previously.

Creating a virtual group is also associated with a cost, in terms of state maintenance and messaging overhead. The net benefit of a virtual group should take this cost into account. In other words, the potential benefit should offset the cost of setting up a virtual group. For example, the coordinator can choose those candidate groups with benefit values larger than a certain threshold.

```
Step 1: for each leaf server $L_i$
    Update subscriptions to coordinator
Step 2: for the coordinator
    Identify virtual group candidates
Step 3: for each virtual group candidate $G_j$
    for each leaf server in $G_j$
        Send Group message upward subtree $T_j$
    for each non-leaf server on subtree $T_j$
        Update forwarding table for $G_j$
Step 4: Repeat step 1-3 for each virtual group candidate
```

Figure 6. Distributed algorithm for virtual group setup

```
Step 1: for root $R$
    Send Shortcut Probe message down subtree $T_i$
Step 2: for each non-root server on sub-tree $T_i$
    if branching point or leaf server
        Check Shortcut Probe message for shortcut
        if shortcut exists
            Send back Shortcut Reply message
        if not leaf server
            Send new Shortcut Probe message down
    else
        Append itself to Shortcut Probe message
        Forward Shortcut Probe message on
Step 3: Repeat step 1-2 for each virtual group
```

Figure 7. Distributed algorithm for shortcut setup

After finalizing the selection of virtual groups, the coordinator initializes the process of group creation by informing the leaf servers of the groups that they belong to. For each virtual group being notified of, a leaf server sends a *Group* message up the pub/sub tree. At receiving a *Group* message, an internal tree node sets up the forwarding state in the corresponding forwarding table that it maintains for the virtual group. The setup process is complete when the tree root R receives the *Group* message for each virtual group. This protocol is briefly summarized in Fig. 6.

This protocol can be invoked repeatedly to update virtual groups according to subscription changes. The coordinator is responsible for initiating protocol executions and for maintaining correct content delivery throughout group reconfigurations. For example, the tree root R can be instructed to temporarily buffer all content messages during the setup process.

4 Optimizing Virtual Groups

The *Group* messages described in the previous section create a content delivery (sub)tree for each virtual group. This tree is induced by the group membership and embedded inside the default pub/sub tree. The tree structure can be improved by adding shortcuts to bypass forwarding-only tree nodes. This section presents a distributed protocol for materializing such shortcuts. The protocol is applied to each existing virtual group in the pub/sub system.

First, the root R sends *Shortcut Probe* messages down the virtual group tree. Each probe message represents a probing thread. After receiving a probe message, an internal node checks the corresponding forwarding table. If there are more than one downstream nodes, i.e., if the current node is a branching point on the tree, the probing thread associated with the probe message is ended. However, new probe messages are sent down the sub-tree rooted at the current node, creating new probing threads. On the other hand, if the receiving node is forwarding only, it appends itself to the probe message and forwards it on to the next hop. In this way, a probe message records its own traveling path. When a probing thread is ended at a branching point, or when it reaches the a leaf node, the path recorded by the probe message is checked for possible shortcuts, i.e., if the path consists of at least two hops. If a shortcut is identified, a *Shortcut Reply* message is sent back to the origin of the probe message to set up the shortcut. The distributed algorithm is formulated in Fig. 7.

During content delivery, a tree node (except the leaves) first checks if there exists a shortcut on each of the downstream paths and uses the shortcut for message forwarding.

5 Performance Evaluation

This section presents results from extensive simulations to evaluate the hybrid content-based pub/sub system.

5.1 Methodology

The simulations were conducted using a regular 4-ary tree of 4 levels, i.e., there are 64 leaves and 85 nodes in total. For the purpose of constructing virtual groups, the content space is partitioned into 100 unit cells. We experimented with combinations of different distributions for both the popularity and the message density of the content space. These distributions are widely used in the pub/sub literature [13, 6]. In particular, we study three scenarios:

- **uniform-uniform**, or uni-uni for brevity. Both the popularity and publishing density follow a random uniform distribution. More precisely, the number of leaf servers interested in each cell is randomly selected from the range $[1, 64]$. The amount of content messages published in each cell is randomly distributed in the range $[1, 100]$.
- **zipf-uniform**, or zipf-uni for short. The popularity of content messages follow a zipf distribution: the ith cell has $64 * i^{-1}$ subscribing leaf servers. The message density follows the same random uniform distribution as described above.
- **zipf-zipf**, The popularity of the content space follows the same zipf distribution as the previous scenario. The message density also has a similar zipf distribution: the ith cell has $100 * i^{-1}$ messages.

We measure the performance of the proposed hybrid pub/sub scheme using the following metrics:

- **Delay**: The average delay to receive subscribed content at the leaf servers. The delay to deliver a message is determined by two factors: the network delay of forwarding the message between the pub/sub servers, and the processing delay of matching the message against the subscription tables at relevant pub/sub servers. We assume that each overlay hop of forwarding takes one unit time, as well as each message matching operation.
- **Messaging cost at internal nodes**: The amount of content messages received at an internal server node, whether matching is performed or not. This metric is normalized by the total number of published messages.
- **Matching cost at internal nodes**: The number of matching operations performed at an internal nodes. It is also normalized by the total message count.
- **Overhead**: We measure the messaging overhead of the hybrid approach, i.e., the number of messages spent to maintain the virtual groups and to set up shortcuts.

5.2 Results

We simulated the hybrid pub/sub approach with and without applying shortcutting, and compare the results with a pure content-based pub/sub system. During the process of virtual group initialization, we also investigate three configurations: the first materializes the top 10% virtual group candidates identified using the protocol presented previously, which are ranked based on their benefit values. The second and the third select the top 20% and all 100% from the candidate virtual groups, respectively. Since the protocol coordinator does not have enough information to determine the benefit of shortcutting at the time of virtual group initialization, only the benefit from saved matchings is used at this stage. Building a virtual group improves the pub/sub system efficiency, but at the cost of state and messaging cost. The purpose of simulating these three options is to quantify this trade-off between performance and overhead.

5.2.1 Delay

Table 1 summarizes the content delivery delay of the hybrid approach, in terms of the average delay experienced by the leaf servers. Since the pub/sub tree has 4 levels, it takes 3 hops to deliver a message from the root to a leaf. Thus, the total delay to receive a message is 6 for the default content-based system as matching is performed at each hop. For the hybrid scheme, the delay is reduced to 4 if all virtual groups are used (100%), because only one matching is required at the root to determine which group that the message belongs to. When the virtual groups go through a selection process based on their benefit values (10% and 20%), the delay at the leaf servers is between 4.078 and 5.377.

For the same configuration, the "zipf-zipf" distribution experiences the lowest content delivery delay. With only 10% of virutal groups, the hybrid approach can reduce the delay

Table 1. Average delay at leaf servers.

	hybrid	hybrid+shortcuts
uni-uni(10%)	5.377	5.375
uni-uni(20%)	4.950	4.943
uni-uni(100%)	4.000	3.885
zipf-uni(10%)	4.748	4.582
zipf-uni(20%)	4.568	4.277
zipf-uni(100%)	4.000	3.203
zipf-zipf(10%)	4.137	4.037
zipf-zipf(20%)	4.078	3.942
zipf-zipf(100%)	4.000	3.796

to very close to the optimal value, i.e., the delay with 100% virtual groups. On the other hand, the uni-uni distribution has the highest content delivery delay. This suggests that the hybrid approach is more powerful for application with skewed subscribing popularity and publishing density distributions, where more popular content is also published more often. In this case, a small number of virtual groups can substantially improve the performance of the pub/sub system.

When shortcutting is applied to the hybrid approach, the delay can be further minimized, e.g., to below 4 with all virtual groups incarnated. Generally speaking, the further reduction in delivery delay is more notable with more virtual groups used. This can be explained by the fact that given a fixed number of virtual groups to be materialized, larger groups are selected first, while smaller (sparser) groups present better chances for shortcutting.

Fig. 8,11,14,17,20,23 depict the more detailed distribution of the delay at the leaf servers under various settings. Overall, the improvement in content delivery delay is mainly the result of applying virtual groups, while shortcutting also contributes, to various extent depending on the size and membership distribution of the constructed virtual groups.

5.2.2 Messaging Cost

Table 2 shows the average percentage of messages received by an internal server. Without shortcutting, each internal server receives the same amount of messages, whether there exist virtual groups or not. Shortcuts bypass the non-branching internal servers in a virtual group, and thus eliminate unnecessary messaging by up to 73%, when all virtual groups are in use ("zipf-uni"). Since shortcuts are more likely to exist in sparser groups, the benefit of shortcutting is not as notable when only large groups are used (10% and 20%). Nevertheless, it can reduce the messaging cost by up to 16-28% ("zipf-uni").

Fig. 9,12,15,18,21,24 describe the distribution of messaging cost at internal servers. Out of all three scenarios, the "zipf-uni" and "zipf-zipf" combinations benefit more from shortcutting. These configurations produce more smaller virtual groups, compared with random uniform group sizes in the "uni-uni" distribution.

Table 2. Average messages (%) received at internal servers.

	w/o shortcuts	with shortcuts
uni-uni(10%)	0.828	0.824
uni-uni(20%)	0.828	0.816
uni-uni(100%)	0.828	0.650
zipf-uni(10%)	0.238	0.200
zipf-uni(20%)	0.238	0.171
zipf-uni(100%)	0.238	0.065
zipf-zipf(10%)	0.540	0.436
zipf-zipf(20%)	0.540	0.403
zipf-zipf(100%)	0.540	0.342

Table 3. Average matches (%) performed at internal servers.

	content-based	hybrid
uni-uni(10%)	0.828	0.651
uni-uni(20%)	0.828	0.507
uni-uni(100%)	0.828	0.000
zipf-uni(10%)	0.238	0.150
zipf-uni(20%)	0.238	0.116
zipf-uni(100%)	0.238	0.000
zipf-zipf(10%)	0.540	0.110
zipf-zipf(20%)	0.540	0.067
zipf-zipf(100%)	0.540	0.000

5.2.3 Matching Cost

Table 3 shows the average percentage of messages that are matched at an internal server. The number of matching operations is determined by the number of virtual groups and is not affected by shortcutting. If each candidate virtual group is taken advantage of, the matching cost at the internal nodes is zero, since no matching needs to be conducted except at the root. In the other scenario with limited number of virtual groups, the matching saving is as high as more than 80%. This suggests that a substantial amount of matching cost can be saved even with selective virtual grouping. In all three scenarios, the "zipf-zipf" combination sees the most significant improvement in matching cost. Only 10% virtual groups can reduce the matchings by about 80%. This result is consistent with the result of content delivery delay shown previously.

Fig. 10,13,16,19,22,25 show the distribution of matching cost at internal servers. Although the hybrid approach is more valuable for situations with highly skewed popularity and density distributions in the content space, it is universally beneficial, as can be seen in various simulation settings.

5.2.4 Protocol Overhead

Implementing virtual groups incurs control messaging overhead, as shown in Table 4 (column 2). However, this overhead is moderate when only 10% or 20% virtual group are actually constructed. In addition, it can be amortized over time by the messages delivered by the virtual groups. Optimizing virtual groups with shortcuts incurs additional control over-

Table 4. Total messaging overhead and saving.

	w/o shortcuts	with shortcuts	
	overhead	overhead	msg. saving
uni-uni(10%)	1298	2051	368
uni-uni(20%)	2441	3873	1133
uni-uni(100%)	7463	12352	17875
zipf-uni(10%)	478	826	4309
zipf-uni(20%)	626	1116	7575
zipf-uni(100%)	1170	1928	19688
zipf-zipf(10%)	519	897	1195
zipf-zipf(20%)	670	1192	1578
zipf-zipf(100%)	1170	1928	2279

head (Table 4, column 3). However, shortcuts reduce the content messages being forwarded by the pub/sub servers, and the resulted messaging savings (Table 4, column 4) can potentially offset the messaging overhead of constructing shortcuts and virtual groups, as in the "zipf-uni" and "zipf-zipf" configurations.

A second type of overhead comes from state maintenance for virtual groups. In the hybrid pub/sub system, each group member maintains a forwarding table for the virtual group. Since the pub/sub network usually consists of well provisioned application-layer servers the state maintenance overhead does not pose resource constraint problems. Content delivery in the pub/sub network does not rely on IP multicast, and therefore the number of virtual groups are not confined by the available IP multicast channels, as assumed in previous work [14, 1]. Moreover, in practice, it is not necessary to build many virtual groups to achieve the desired performance; only a few of the highest ranking groups can bring significant improvement.

5.3 Summary

In summary, the simulations have demonstrated that:

1. The hybrid approach is beneficial in various scenarios with different popularity and density distributions in the content space.
2. Virtual groups, even only a few, can significantly reduce the matching cost at the pub/sub servers, and thereby improve the delay of content delivery. This technique is more beneficial for larger virtual groups.
3. Shortcutting is effective in eliminating redundant messaging inside virtual groups, resulting in improved efficiency in the pub/sub network and reduced content delivery delay. This technique is more beneficial for smaller, sparser virtual groups.

6 Related Work

This section discusses previous work related to content-based pub/sub systems. A survey on the general pub/sub topic can be found in [11].

In [13], various content clustering schemes were studied, including a grid-based algorithm. However, the grid was used to group *similar* content, while in this paper, grouping is conducted only for those cells with the same subscribers. In [14], matching and distribution issues were discussed based on the clustering algorithms presented in [13]. Since clustering can cause content to be delivered to un-interested clients, matching is first performed to build a list of interested clients for distribution. In [20], the k-mean method was evaluated in a preference clustering framework. Since the "clustering" approach usually assumes a fixed number of available IP multicast channels, the problem of mapping clusters to these multicast channels was studied in [1]. In [15], a set of schemes for using a small number of IP multicast groups in a content-based pub/sub network were also investigated.

In the "exact-match" category, Gryphon [4] organizes the pub/sub network into a single-source tree and proposes a link matching algorithm to forward content towards directions of matching subscriptions. In [18], a similar pub/sub topology was assumed, but this work considers a different problem of filter placement, i.e., how to find the optimal positions for a fixed number of matching points in a pub/sub tree. Siena [7] builds a symmetric spanning tree, and each pub/sub server can be a potential publisher or subscriber. Kyra [6] proposes to create multiple pub/sub networks based on content clustering, with each network responsible for a subset of the content space, and thus requires a pre-processing stage of clustering. In [8], a fast matching algorithm was proposed and it can achieve matching time of milliseconds in a highly complex content space. INS/Twine [3] uses a peer-to-peer based pub/sub architecture for resource discovery. Herald [5] aims to build a distributed event notification system at the global scale. Other work on pub/sub can be found in [16, 10, 12, 17].

7 Conclusion

In this paper, we have presented a new hybrid approach to content-based pub/sub systems. The hybrid approach leverages shared interests in a distributed content delivery environment by identifying virtual groups. These virtual groups serve as expressways for the content-based pub/sub network. Messages are only matched once at the entry point and then forwarded to the rest of the virtual group, similar to in subject-based pub/sub systems. In addition, these expressways can be made faster by adding shortcuts to the virtual group tree. We have also proposed distributed protocols for virtual group management and shortcut construction. Simulations under various settings have shown that the hybrid approach is highly effective in reducing the content delivery delay and in improving the pub/sub service efficiency.

Acknowledgment

This work was supported in part by by NSF CAREER award grant ACI-0238379 and NSF grant CCR-0313026.

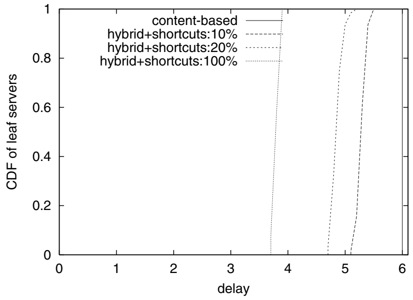

Figure 8. Average delay at leaf servers (uni-uni)

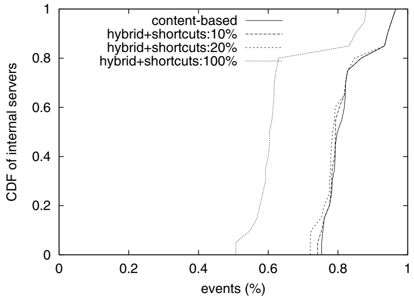

Figure 9. Average messages received at internal servers (uni-uni)

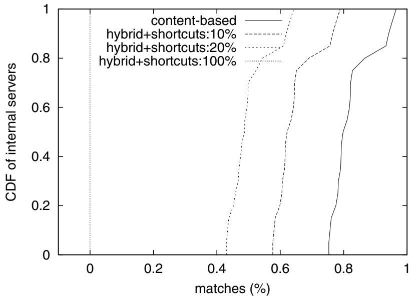

Figure 10. Average matches performed at internal servers (uni-uni)

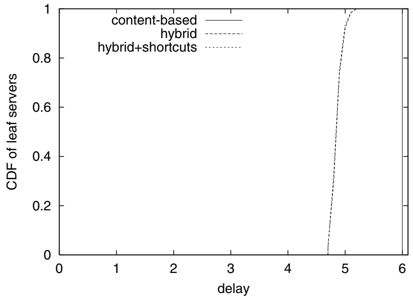

Figure 11. Average delay at leaf servers (uni-uni:20%)

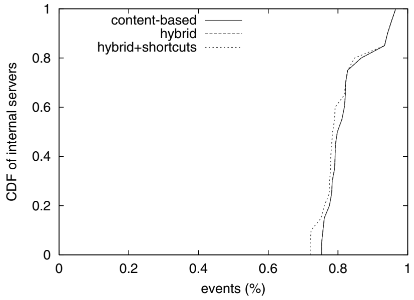

Figure 12. Average messages received at internal servers(uni-uni:20%)

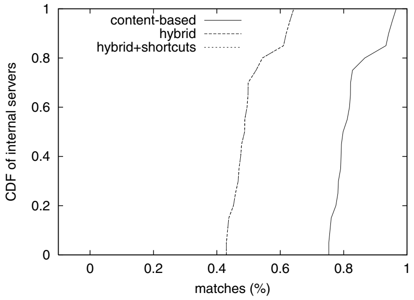

Figure 13. Average matches performed at internal servers (uni-uni:20%)

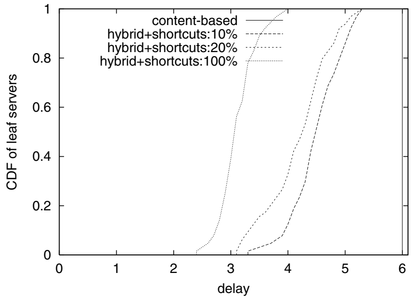

Figure 14. Average delay at leaf servers (zipf-uni)

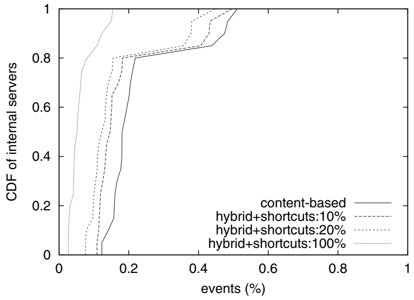

Figure 15. Average messages received at internal servers (zipf-uni)

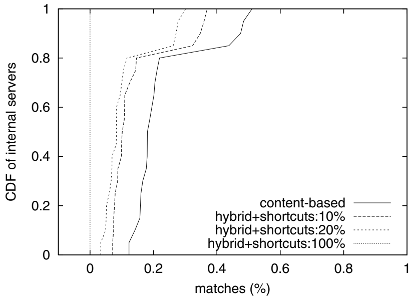

Figure 16. Average matches performed at internal servers (zipf-uni)

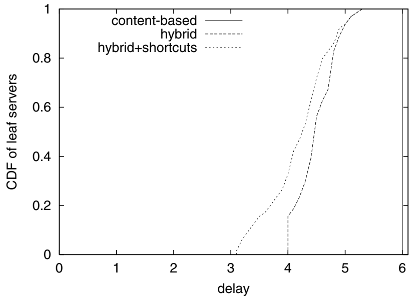

Figure 17. Average delay at leaf servers (zipf-uni:20%)

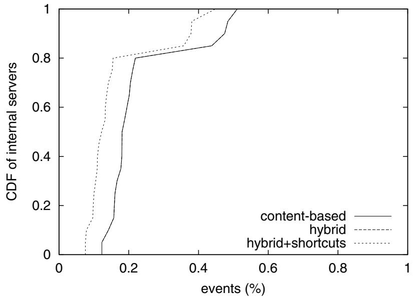

Figure 18. Average messages received at internal servers(zipf-uni:20%)

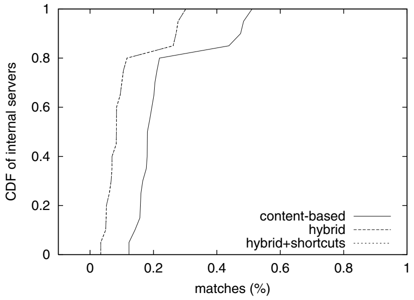

Figure 19. Average matches performed at internal servers (zipf-uni:20%)

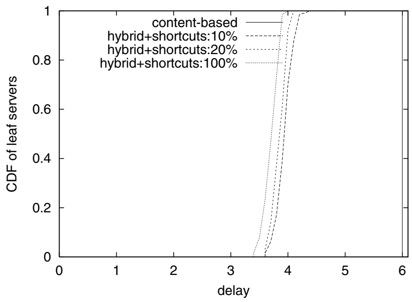

Figure 20. Average delay at leaf servers (zipf-zipf)

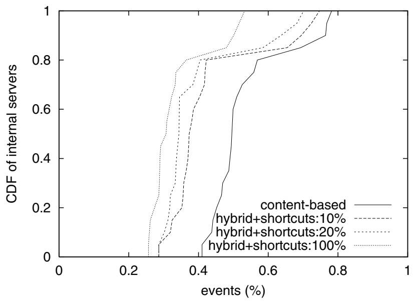

Figure 21. Average messages received at internal servers (zipf-zipf)

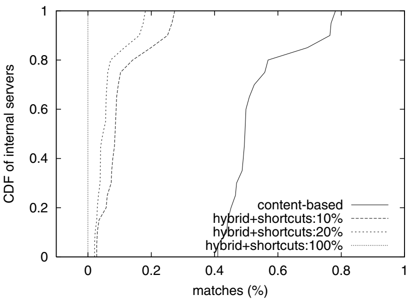

Figure 22. Average matches performed at internal servers (zipf-zipf)

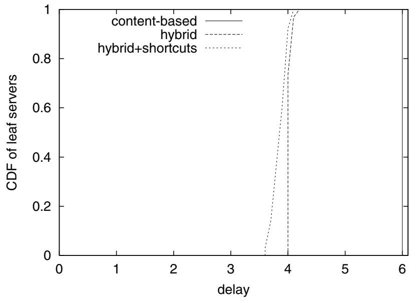

Figure 23. Average delay at leaf servers (zipf-zipf:20%)

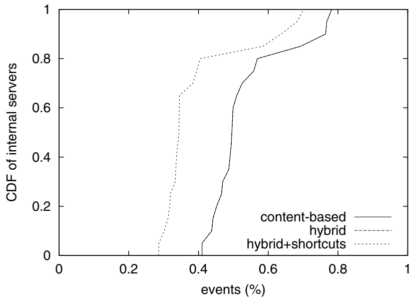

Figure 24. Average messages received at internal servers(zipf-zipf:20%)

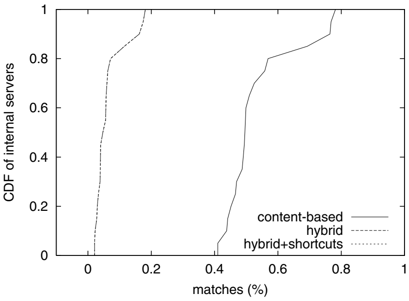

Figure 25. Average matches performed at internal servers (zipf-zipf:20%)

References

[1] M. Adler, Z. Ge, J. Kurose, D. Towsley, and S. Zabele. Channelization Problem In Large Scale Data Dissemination. In *Proceedings of IEEE ICNP*, November 2001.

[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 *Proceedings of ACM PODC*, May 1999.

[3] M. Balazinska, H. Balakrishnan, and D. Karger. INS/Twine: A Scalable Peer-to-Peer Architecture for Intentional Resource Discovery. In *Proceedings of Pervasive*, August 2002.

[4] G. Banavar, T. Chandra, B. Mukherjee, J. Nagarajarao, R. E. Strom, and D. C. Sturman. An Efficient Multicast Protocol for Content-Based Publish-Subscribe Systems. In *Proceedings of IEEE ICDCS*, June 1999.

[5] L. F. Cabrera, M. B. Jones, and M. Theimer. Herald: Achieving a Global Event Notification Service. In *Proceedings of HotOS*, May 2001.

[6] F. Cao and J. P. Singh. Efficient Event Routing in Content-Based Publish-Subscribe Service Networks. In *Proceedings of IEEE INFOCOM*, March 2004.

[7] A. Carzaniga, M. J. Rutherford, and A. L. Wolf. A Routing Scheme for Content-Based Networking. In *Proceedings of IEEE INFOCOM*, March 2004.

[8] A. Carzaniga and A. L. Wolf. Forwarding in a Content-Based Network. In *Proceedings of ACM SIGCOMM*, August 2003.

[9] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, Cambridge, MA, 1990.

[10] G. Cugola, E. D. Nitto, and A. Fuggetta. The JEDI Event-Based Infrastructure and Its Application to the Development of the OPSS WFMS. *IEEE Transactions on Software Engineering*, 2001.

[11] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec. The Many Faces of Publish/Subscribe. *ACM Computing Surveys*, 35(2):114–131, June 2003.

[12] L. Fiege, G. Muhl, and F. C. Gartner. A Modular Approach to Build Structured Event-Based Systems. In *Proceedings of ACM SAC*, 2002.

[13] Z. Liu, J. L. Wolf, P. S. Yu, and L. Zhang. Clustering Algorithms for Content-Based Publication-Subscription Systems. In *Proceedings of IEEE ICDCS*, July 2002.

[14] Z. Liu, J. L. Wolf, P. S. Yu, and L. Zhang. New Algorithms for Content-Based Publication-Subscription Systems. In *Proceedings of IEEE ICDCS*, May 2003.

[15] L. Opyrchal, M. Astley, J. Auerbach, G. Banavar, R. Strom, and D. Sturman. Exploiting IP Multicast in Content-Based Publish-Subscribe Systems. In *Proceedings of ACM/IFIP/USENIX Middleware*, April 2000.

[16] E. P. Robert E. Gruber, Balachander Krishnamurthy. The Architecture of the READY Event Notification Service. In *Proceedings of IEEE ICDCS*, May/June 1999.

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

[18] R. Shah, R. Jain, and F. Anjum. Efficient Dissemination of Personalized Information Using Content-Based Multicast. In *Proceedings of IEEE INFOCOM*, June 2002.

[19] J. Tian and G. Neufeld. Forwarding State Reduction for Sparse Mode Multicast Communication. In *Proceedings of IEEE INFOCOM*, March/April 1998.

[20] T. Wong, R. Katz, and S. McCanne. An Evaluation of Preference Clustering in Large-Scale Multicast Applications. In *Proceedings of IEEE INFOCOM*, March 2000.