Towards Scalable Content-based Publish-Subscribe Networks

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

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

***Abstract* — Architecture design for content-based publish-subscribe service networks is challenging for two reasons: first, communication in such systems is guided by the content of event publications and subscriptions, rather than by network addresses; second, while an event often matches subscriptions from multiple locations, the highly diversified matching patterns and hence communication needs imply that existing multicast techniques cannot be readily used for efficient event delivery.**

In this paper, we propose a new architectural approach called MEDYM – Match Early with DYnamic Multicast. Unlike existing approaches, MEDYM does not build static overlay networks for event delivery. Instead, an event is matched against subscriptions as early as possible, to identify destinations with matching subscriptions; then, a multicast tree is dynamically constructed to route the event to the destinations with high network efficiency.

We evaluate the MEDYM architecture using detailed simulations, and compare it with the two major existing design approaches: Content-based Forwarding and Channelization. Experimental results show that MEDYM significantly improves efficiency of event delivery, and is highly flexible and robust. We also analyze potential overheads introduced in MEDYM, and found them to be well acceptable and more than outweighed by the benefits of the approach. We expect the basic MEDYM architecture to scale to pub-sub networks of thousands of servers, which we believe is adequate for many interesting applications.

Keyword* — *System design, Simulations, Publish-subscribe, Event notification.

A. INTRODUCTION

Content-based publish-subscribe (pub-sub for short) is an important paradigm for asynchronous communication among entities in a distributed network. In such systems, users *subscribe* to content-based conditions, and will be notified when other users *publish* events to the system that satisfy their conditions. Content-based subscriptions are highly expressive, and can specify complex filtering criteria along multiple dimensions of event content. A content-based stock alert system, for example, may allow subscriptions on stock price movement events with "(*ticker*=IBM) AND (*price*>100 OR *volume*>8 million)". Such timely delivery of highly customized information is of great value to many distributed applications, such as distributed system monitoring [22], alerting and notification [30], personalized information dissemination [31] and multi-party games [2], and application integration [28].

In this paper, we study architecture design for large-scale content-based pub-sub service networks, which are expected to handle large numbers of content-based subscriptions and high volume of event publications from widely distributed users. As shown in Figure 1, in such a network, a set of pub-sub servers is distributed over the Internet; clients access the pub-sub service, either to publish events or to register subscriptions, through appropriate servers, such as those that are close to them or in the same administrative domains. Thus, pub-sub servers serve as publication/subscription proxies on behalf of clients, and we can view the pub-sub service as one of getting events from servers where they are published to the servers that subscribe – as proxies – to the events. We call these publication servers (*p-servers*) and subscription servers (*s-servers*). The same server may serve as a p-server and as an s-server. In this paper, we focus on the internal design of the service, and do not address the communication between pub-sub servers and their associated clients.

A content-based pub-sub service network presents a unique design challenge, because its communication paradigm is not directly supported by existing network protocol primitives. First, communication is based on the content of events and subscriptions, rather than network addresses. Publishers do not specify destination addresses for the events. Rather, events have to be *matched* with subscriptions to identify where they should be sent. Second, it is not clear how to best *route* the event, even after the destinations are known. While an event may match subscriptions from multiple s-servers, existing group communication techniques, such as IP multicast [12] or application-layer multicast [10], cannot be directly used. This is due to the high diversity of content-based subscriptions: different events may satisfy the interests of widely varying sets of s-servers. In the worst case, the number of such sets can be exponential in

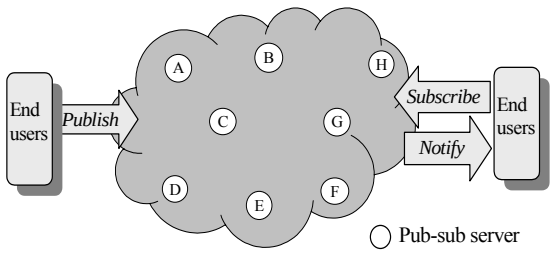


Figure 1. Example of a publish-subscribe service network.

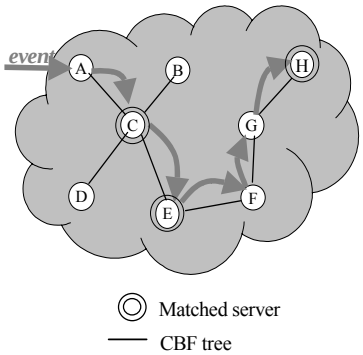


Figure 2. Event delivery in Content Based Forwarding (CBF) network.

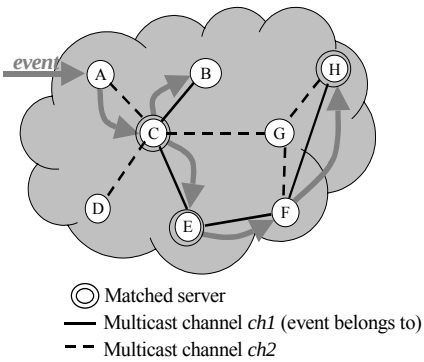


Figure 3. Event delivery in Channelization network.

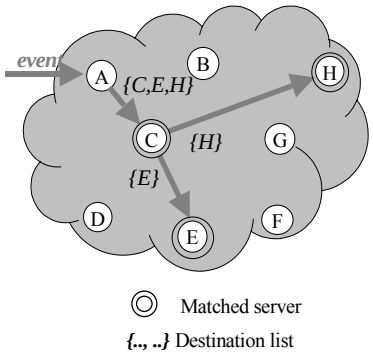


Figure 4. Event delivery in MEDYM network.

the network size (2^n where n is the number of servers), and it is impractical to build and manage a multicast group for each such set. Finally, the matching and routing problems are interrelated: routing decisions are based on matching results, while routing capabilities in turn affect where and how events are matched.

The contributions of the research described in this paper can be summarized as follows:

- We propose a new architecture for content-based pub-sub networks, called MEDYM (Match Early with DYnamic Multicast), which embodies a new way to address the matching and routing problems.
- We present design and efficient implementation techniques for MEDYM. We also closely analyze the new challenges it introduces, and show that they can be addressed with acceptable overhead.
- We propose and design *dynamic multicast* as a general routing method. While content-based pub-sub is the driving application to which it is very well suited, we expect this flexible multicast scheme to be broadly applicable for group communication scenarios where there is wide diversity and unpredictability in message destinations.
- We evaluate MEDYM and compare it with the major existing alternative approaches through detailed simulation. We find that compared to these approaches, MEDYM provides a better combination of efficient event delivery and low implementation and maintenance cost.

The rest of the paper is organized as follows: In the next section, we review two major existing design approaches for content-based pub-sub service networks and discuss their tradeoffs. In Section C, we present an overview of the MEDYM approach, including its major benefits and key challenges. In Section D, we explore the design space for dynamic multicast, and develop methods for efficient implementation. We describe our simulation setup in Section F and present detailed simulation results in Section G. In Section H, we conclude the paper and identify directions for future work.

B. Related Work

Existing content-based pub-sub network architecture design can be largely categorized into two classes, which we call the *Content-based Forwarding* (CBF) approach [1][4][6][7][11] and the *Channelization* approach [16][23][24][29].

1st. Content-based Forwarding (CBF)

As shown in Figure 2, in a CBF network, pub-sub servers organize *a priori* into one or more acylic overlay networks, which we call *CBF trees*. Subscriptions from s-servers are broadcast along a CBF tree and recorded in the *forwarding table* at every server, in the form of sum of subscriptions from each direction. Thus, subscriptions from the same direction can be *aggregated* for efficient storage, update, and matching. Events are routed along the CBF tree, matched with subscriptions in the forwarding tables at every step, and forwarded only towards directions with matching subscriptions.

As an optimization, to avoid sending subscriptions to servers that will not publish any relevant events, p-servers may broadcast *advertisements*, which describe the possible content of their future publications, on the CBF tree. Then, subscriptions are forwarded along the tree only toward p-servers that have made matching advertisements.

By content-based filtering at every step, CBF accurately routes events (and subscriptions) toward matching directions. However, its implementation and maintenance cost can be high.

First, performing content-based matching at every step is expensive. It includes parsing event content, extracting values from fields, and evaluating complex criteria, such as full-text search, range queries and/or Boolean expressions, against large number of subscriptions. Furthermore, many matching operations in the process of delivery may be redundant, when an event is repeatedly matched with the same subscriptions (at intermediate servers) before reaching the destination(s).

Second, the event traffic load on servers can be high and imbalanced. Since the CBF tree is built *a priori*, to reach matching destinations events may be routed through intermediate servers that are not interested in the events, resulting in unnecessary processing and bandwidth load. Servers and network links at the center of the network are especially susceptible and can become system bottlenecks.

Finally, the maintenance cost can be high. Whenever CBF tree topology changes, such as to adapt to server or underlying network environment changes, the forwarding tables associated with the tree also need to change. Because subscriptions have been aggregated and their source information unavailable, it is not known how to efficiently adjust the forwarding tables to reflect the changes. One way to reconstruct the forwarding tables is to broadcast subscriptions and advertisements again along the changed CBF tree, generating high network traffic and processing load.

In this paper, we use work in [6] and [7] as representatives for the CBF approach, as they are among the most prominent and complete works in this direction. Many other distributed pub-sub systems follow the CBF approach. In JEDI [11], a hierarchical event routing network was proposed, but was found in [6] to perform worse than the peer-to-peer topology as in Siena. In Gryphon [1], a link-matching algorithm was designed to partially match the event with an annotated network topology data structure to determine the directions in which to forward the event to. [8] designed detailed content-based forwarding algorithm for CBF. [4] proposed to build multiple small CBF trees, so that event routing in each tree can be implemented with lower cost.

2nd. Channelization

The central idea in the Channelization approach is to utilize existing multicast techniques, such as IP multicast [12] or application-level multicast [9], for event delivery. An example network is shown in Figure 3. Offline, the event space is partitioned into a limited number of disjoint *channels.* For each channel, a multicast group is built that spans all the servers that carry any client subscription that overlaps with (could match) any event in that channel. When publishing an event, a p-server first identifies the channel that the event belongs to, and then multicasts the event in the corresponding group. Event forwarding is by multicast routing, which is simpler and faster than content-based forwarding in CBF.

The main challenge for Channelization approach is that the available number of multicast channels is often much smaller than the 2^n possible event destination sets (see Section A). As a result, the same channel often has to accommodate events with substantially non-overlapping destination sets, and servers can easily receive irrelevant events in groups that they join for other events.

To reduce extraneous traffic and inappropriate receipt, clustering techniques are used to compute event space partitions, with the goal of maximizing the destination set overlap between events in the same channel. However, the effectiveness of clustering depends heavily on the distribution properties of events and subscriptions, and it is likely to be often difficult to accurately match the diversified user interests with a small number of groups. Furthermore, the distributions themselves can be difficult to estimate and react to changes in time.

In this paper, we will use the work in [23] as a representative for the Channelization approach.

The CBF and Channelization approaches thus balance very differently the tradeoff between network traffic reduction in event delivery and low computational and maintenance cost. CBF achieves higher routing accuracy and thus lower traffic than Channelization, but incurs significant computational and maintenance cost. However, since both use 'statically' determined overlay networks, both still generate unnecessary network traffic, and both also require potentially expensive network management and state maintenance when network topology changes. Our goal is to develop an approach with both lower computational cost than CBF and at least as good network traffic efficiency, while also easy to manage and maintain.

C. MEDYM OVERVIEW

In this section, we propose a new architecture for content-based pub-sub network called *MEDYM*: *Match Early with DYnamic Multicast*. Unlike existing approaches, MEDYM does not build static overlay networks. Its event delivery process is as shown in Figure 4. At a p-server, a published event is first matched against subscriptions from remote s-servers, to obtain a *destination list* of s-servers with matching subscriptions. Then, the event is routed to the destination servers through *dynamic multicast*: On receiving an event message with its associated destination list, based on its destination list, a server dynamically computes the next-hop servers to which to forward the message, as well as the new destination list for each of the next-hop servers. In this way, a transient *dynamic multicast tree* is built step by step. Along this tree, the event is routed to (and also through) only its matching s-servers.

Benefits. The match-early and dynamic multicast aspects of MEDYM both offer synergistic advantages. With match-early, an event is matched only once in the process of delivery, achieving low computation cost. While the match may be against a larger number of aggregated subscriptions than is done at each server along the path in CBF, the overall computational cost is

expected to be much smaller Once matched, the rest of event delivery is through simple address-based forwarding in dynamic multicast, with early accurate destination information that allows efficient routing decisions to be made.

Given the results of early matching, dynamic multicast routing conceptually provides substantial benefits. It routes each event through only the set of servers that actually subscribe to it. It is highly flexible, and allows routing decisions and paths to be optimized for individual event messages, which suits well the diversified and unpredictable communication needs of content-based pub-sub. Such fine-grained routing optimization is difficult to achieve in static, predetermined event delivery networks.

A final advantage of a MEDYM network is its maintainability, since it does not have to maintain 'static' overlay topologies and routing states that can be complicate when dealing with dynamic network changes.

Challenges. However, MEDYM also introduces new challenges that are not present in existing approaches.

First, every MEDYM server must be able to communicate directly (at the overlay layer) with all the other MEDYM servers in the system. This is not a requirement in static overlay networks, in which servers communicate with only a fixed set of neighbors. However, we believe it is reasonable given the scale and nature of a managed service network, and we will address efficient connection management in Section E. We also expect that in a content-intensive network, the amount of information kept about other servers information will be much smaller than the information kept for content-based subscriptions and will not add to too much storage or update overhead.

Second, dynamic multicast introduces new overheads such as real-time route computation and the traffic overhead of destination lists. As a new routing scheme, it is unclear *a priori* whether and how it can be implemented with high enough efficiency.

Finally, MEDYM servers need to also know about (sum of) subscriptions from all other servers for early matching, and server network location information for network-efficient dynamic multicast. This information is not required in the other approaches; e.g. CBF can potentially store aggregated information for each direction from a server. The overhead of this information is also an important question.

To address these challenges, we examine methods for efficient implementation of dynamic multicast in Section D, and mechanisms for state and network magement of MEDYM in Section E. The content-based matching used in match-early is an application-specific plug-in module in MEDYM, and is not a focus of this paper.

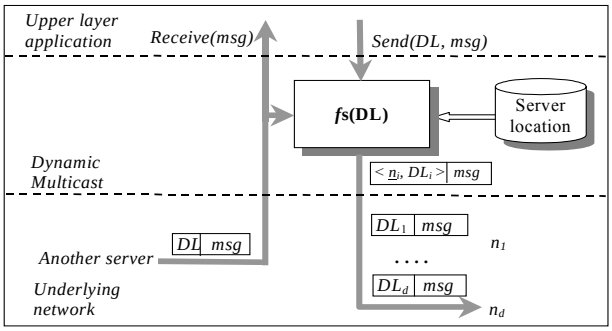


Figure 5. Dynamic multicast routing.

D. DYNAMIC MULTICAST

Dynamic multicast is designed as a general scheme for routing a message to a group of destinations. As shown in Figure 5, it serves a simple interface to the upper layer application: *Send*(*DestinationList, message*), and delivers received messages to application through a callback function *Receive*(*message*).

Dynamic multicast is a stateless protocol in which no session information needs to be established or managed. For a message received from either the upper layer application or another server, based on its associated destination list (*DL*), *dynamic multicast routing* runs as follows:

$$< n_i, DL_i > = f_s\,(DL) \qquad i = 1 \ldots d$$

The routing algorithm f_s computes a list of d $<n_i, DL_i>$ pairs, where n_i is the ith next-hop server, and DL_i is the new destination list for n_i. The algorithm's input and output must satisfy the following *routing invariants:*

(a) $\bigcup_{i=1}^{d} DL_i = DL - \{s\}$

(b) $DL_i \cap DL_j = \phi, \quad i \neq j$

(c) $n_i \in DL_i$

Correctness of dynamic multicast can be derived from the invariants: a message is delivered to all destination servers, as the routing process does not stop until with empty destination lists. The invariants also guarantee that dynamic multicast is efficient and fair: a message only traverses the destination servers, and each server at most once. This results in minimal total routing load, and the load is aligned with servers' self-interests: unlike in CBF or Channelization approach, MEDYM servers receive and route for messages that they themselves are interested in. Given the heterogeneous server interests in content-based pub-sub service networks, we believe this property provides an important incentive for servers to participate in the network. Dynamic multicast is highly flexible. Different routing algorithms f_s can be developed for different routing optimization goals and will have the above properties as long as the invariants hold.

In MEDYM, it is used to route not only events, but also subscription and control messages when needed. Next, we discuss its efficient implementation.

1st. High-level Routing Method

1) Source-based dynamic multicast

One way to implement dynamic multicast is source-based routing. The source server computes the whole multicast tree and encodes it in the destination list, for instance by sorting server IDs in depth-first order. A forwarding server simply reads the tree structure from the destination list and forwards the message as instructed. The drawback of this approach is that its routing quality depends on source's knowledge about the network, at the time of message sending. If this knowledge is incomplete or out-of-date, the multicast tree computed can be sub-optimal.

2) Distributed dynamic multicast

Distributed dynamic multicast is a scheme in which each server independently resolves its local part of the tree. In a widely distributed network, servers often have more accurate or up-to-date information about local network environments than for remote areas. In distributed dynamic multicast, such information can be utilized to improve routing decisions. Distributed dynamic multicast is also highly resilient to failures. When a server fails to deliver a message to next-hop server n_i, it re-runs $f_s(DL_i - \{n_i\})$ so that the message can bypass n_i and still be delivered to other servers in DL_i. Due to these advantages, in our work, we focus on distributed dynamic multicast.

3) Caching routing decisions

An interesting question is whether there is enough temporal locality among destination lists that dynamic multicast routing decisions can be effectively cached. We will examine caching in the context of real pub-sub workloads in future work, and do not assume it here.

2nd. Detailed Routing Algorithms

We have designed three routing algorithms for computing f_s with different optimization goals.

The first algorithm, *MST,* aims at minimizing the total communication cost in message routing. It uses Kruskal's algorithm [9] to compute the minimum spanning tree for destination servers, and then extracts $<n_i, DL_i>$ from the tree. It can be implemented in either source-based or distributed way. A potential problem with this algorithm is its high computation complexity of $O(D^2 \log D)$, where D is the number of destination servers. Because the routing algorithm is run for every incoming message in real time, it is important that it can be run fast enough to support high routing throughput.

The second algorithm, *SPMST* for *Short-Path-MST*, is a new algorithm we designed that computes an approximate minimum spanning tree in a faster way. The algorithm is as shown in Figure 6. Offline, an array called *ShadowBitVector* is maintained to help in quickly

```
computetShadowBitVectors_s() {      // offline
  foreach server s_i {
    foreach server s_j
      if (DistanceMatrix[i][j]<DistanceMatrix[s][i] &&
          DistanceMatrix[s][j]<DistanceMatrix[s][i])
          Set_jth_bit_in_ShadowBitVector[i]; }}

SPMST_s(DL) {                       // online
  Nexthops = DL;
  foreach server s_i in DL
    if (ShadowBitVector[i] & DLBitVector !=0)
      Nexthops_remove(s_i);
  if (|Nexthops|>maxNextHops))
    Nexthops = closest_nexthops(maxNextHops);
  foreach server s_j in (DL-Nexthops) {
    n_i = closest_nexthop_to(s_j);
    DL_i +={s_j}; }
  return(<n_i, DL_i>); }
```

Figure 6. SPMST routing algorithm.

```
BLSPMST_s(DL) {
  if (I_am_overloaded) {
    if (all_destinations_overloaded)
      n_0 = closest_destination;
    else
      n_0 = closest_not_overloaded_destination;
    DL_0 = DL-{s}-{n_0};
    return <n_0, DL_0>; }
  choose_nexthops_as_in_SPMST;
  if (all_nexthops_overloaded)
    assign_new_DL_as_in_SPMST;
  else
    foreach server s_j in (DL-Nexthops) {
      n_i = closest_non_overloaded_nexthop_to(s_j);
      DL_i +={s_j}; }
   return(<n_i, DL_i>); }
```

Figure 7. BLSPMST routing algorithm.

choosing next-hop servers. We say that server s_i is *shadowed* by server s_j, if s_i is closer to s_j than to current server s, and s is closer to s_j than to s_i. Under this condition, it is deemed as more efficient to let s_j forward the message to s_i than s directly sending to s_i. Therefore, s_i will not be chosen as next-hop server if s_j is also a destination server. Note that in this way, it is inclined to choose close by servers as next-hop servers, and result in multicast trees with lower diameters than the minimum spanning trees. After choosing the next-hop servers, the rest of the destinations are assigned to new destination lists of the next-hop servers closest to them. Using bit vector operations, the algorithm runs fast enough to support high event traffic volume.

The third algorithm, *BLSPMST* for *BaLanced-SPMST*, is shown in Figure 7. It is a variant of *SPMST* that tries to balance routing load across servers, loosely measured by server bandwidth usage in this work A MEDYM server is deemed as overloaded if its bandwidth consumed in routing the last K messages is above *OverloadThreshold* times the total message size. In our experiments, K is set to 1000 and *OverloadThreshold* is set to 3.

Table 1 presents the computation time of the three algorithms, written in Java and run with 2.0 GHz Pentium-III CPU and 512MB memory. Note that the

destination list sizes used in the table are much larger than the average destination list size (see below) in the process of routing one event. The results show that the algorithms, especially SPMST and BLSPMST, are efficient enough to support high routing throughput, confirming our expectation that address-based forwarding in dynamic multicast should be simple and fast.

3rd. Destination List Size

As discussed earlier, a potential overhead in dynamic multicast is that the increased network traffic for destination lists. In Figure 8 we present an analysis of the average destination list size in the process of routing one event, averaged over the edges in a dynamic multicast tree. Since destination list size is reduced at every step by a factor equal to the outgoing degree of the node, we expect the average list size to be about equal to the diameter of the tree. This is confirmed in Figure 9, which shows that the average destination list sizes in multicast trees computed using the three algorithms above are really about equal to the diameters of the trees, a grow slowly with total number of destinations (which are usually all the matched s-servers). We believer such overhead is quite acceptable, especially considering that messages in content-based pub-sub systems are likely to carry complex information, such as attribute-values pairs, full texts or XML documents.

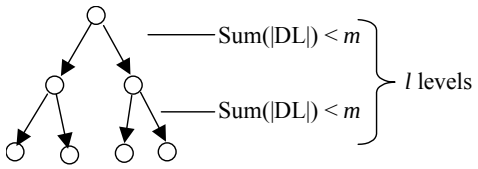
While the average destination list size is small, it can be imbalanced across pub-sub servers, as it is longer at servers close to the root of the multicast tree. We address this issue from two perspectives:

First, we do not attempt to reduce the destination list length at the message source; rather, we believe that a server ID in the destination list is a reasonable and fair "cost" that a server "pays" for using the dynamic multicast routing service. The more use the server makes of the service, i.e. the more destinations its messages are delivered to, the more (bandwidth) cost the server is expected to pay for. This is also an effective mechanism to prevent servers from spamming the network.

Second, destination list size is not the only reason for imbalanced server bandwidth consumption, which is also affected by factors such as the server degrees in the multicast tree. Therefore, our algorithm *BLSPMST* aims at balancing server bandwidth as a whole, not distinguishing bandwidth consumed for destination list or message payload.

Table 1. Computation time of dynamic multicast routing algorithms, with destination list size |DL|.

Routing algorithm	Computation time (ms)		
	|DL|=100	|DL|=500	|DL|=1,000
MST	1.8	9	34
SPMST	0.08	0.29	0.62
BLSPMST	0.08	0.32	0.69



In total: < *ml* destinations. (*m*-1) messages.
On average: *l* destinations/message

Figure 8. Analysis of average destination list size in dynamic multicast tree with *m* servers.

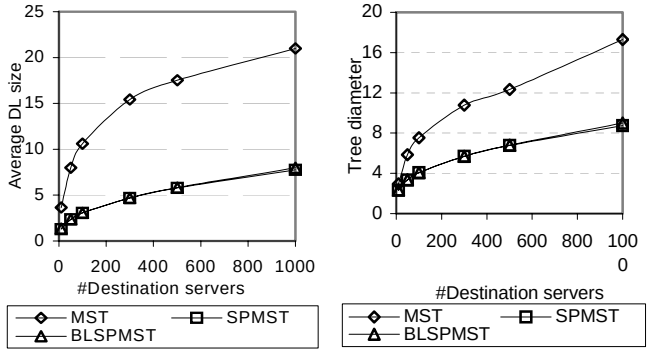


Figure 9. Average destination list size and multicast tree diameter, using different dynamic multicast routing algorithms.

E. MEDYM State and Network Management

In this section, we describe how MEDYM servers maintain state to support event delivery, as well as MEDYM network management.

1st. State Management

In MEDYM, each p-server maintains a *matching table* with an entry (server ID, sum_of_subscriptions) for every s-server in the system. And every MEDYM server maintains a *routing table*, which includes a list of (server ID, IP address, status) for all s-servers, and a *distance matrix M*, where $M_{i,j}$ represents communication cost between s-server *i* and *j*. Advertisement is an optional optimization, in which p-servers broadcast their advertisements to s-servers, using dynamic multicast.

Matching table maintenance

Matching tables in MEDYM differ from the CBF forwarding tables in three ways: first, in MEDYM, only p-servers maintain matching tables, and only subscriptions from s-servers that overlap with local publication interest (local advertisement, if used) need to be maintained. These are not true for CBF, because the CBF servers may need to forward (and thus match for) events that do not match their own subscriptions. On the other hand, MEDYM offers less subscription aggregation opportunity than CBF. In MEDYM matching tables, only subscriptions from the same server can be aggregated, while in CBF forwarding tables, subscriptions from all servers in the same direction can be aggregated. However, subscription aggregation is a very difficult

problem and there have been no efficient, systematic methods developed for it and no evaluations on how successful it can be. Therefore, although it is difficult to quantitatively compare the storage in the two approaches, we expect that the matching tables in MEDYM do not introduce much overhead compared to CBF, and may be even advantageous. Finally, unlike in CBF, matching tables are independent of network topology, and therefore do not need to adapt to network environment changes.

To make a new subscription or cancel a previous one, a server broadcasts the subscription using dynamic multicast. If advertisement is used, s-servers record the advertisements received, and only send the subscription to p-servers with overlapping advertisements, using dynamic multicast. Receiving the subscriptions, p-servers update their matching tables accordingly.

Routing table

Routing table is used to build network-efficient routing paths in dynamic multicast. In MEDYM, s-servers periodically broadcast *Refresh* messages, which include their ID, IP address, status (such as load condition) and network location information.

Server network location information is used in computing the distance matrix. It can be obtained in two ways: first, a server may actively probe other servers and report the probing results. This scheme measures real network latency but generates $O(n^3)$ total network traffic, where n is number of servers in the network. However, note that the probing and broadcasting can be rather infrequent: first, studies show that Internet paths are fairly stable: [32] reported that roughly 80% of Internet routes studied were stable for longer than a day, and [33] shows that in general delay appears steady on time scales of 10-30 minutes. Second and more importantly, inaccurate server distance information or inconsistent information on different servers does not affect dynamic multicast correctness; it may only result in sub-optimal routing network efficiency. As an alternative, MEDYM can benefit from state-of-the-art techniques [15][19][21][27] to approximately yet efficiently estimate server network locations. In this work, we simulated [27] for this purpose. In Section G, we will compare maintenance overhead as well as routing performance between using measurement and estimation.

2nd. Network Management

In MEDYM, network management is also relatively easy because no overlay network topology or session states need to be maintained.

Server join

A MEDYM network has one or more servers as Rendezvous Points (RP) that are in charge of network membership. A new server locates an RP, probably through DNS, and sends a *Join Request* message, asking for permission to join the network. If permitted, RP replies with a *Welcome* message, which includes an ID for the new server, a list of (server ID, IP address, advertisement) for all servers in the network, and a distance matrix *M*.

Receiving the *Welcome* message, the new server initializes its matching table and routing table. It then broadcasts a *Greeting* message with its (server ID, IP address, advertisement) to all servers in the network, using dynamic multicast. The new server also sends subscriptions to servers with overlapping advertisements.

Receiving a *Greeting* message, an existing server adds the information about the new server into its matching and routing table. If the new server's advertisement overlaps with local subscriptions, it sends *Subscribe* message to the new server with the subscriptions.

The new server then starts participating in information update and pub-sub communication in the network.

Server leave and failure

When a MEDYM server leaves, it tells an RP node, which broadcasts a *Departure* message, and all servers remove information about that server from their local tables. If a server abruptly goes down, its failure can be detected either by lack of *Refresh* messages, or in the dynamic multicast routing process, as described in Section D. Failure detection is reported to an RP. RP will try to contact the server, and if this fails, it broadcasts the Departure message. We note that handling failure is relatively easy in MEDYM, as unlike in overlay networks with static topologies, the failure will not lead to network partition or incorrect routing.

Connection Management

Reliability is an important feature in pub-sub communication. We use reliable network protocol such as TCP for inter-server communication. Unlike other approaches, in MEDYM, a server may need to directly communicate with any other server(s) at any time. Frequently opening and closing TCP connections is expensive and limits system throughput. We use persistent connection and pipelining techniques introduced in HTTP/1.1 protocol [14] for efficient connection management. If the total number of simultaneously open TCP connections reaches system's limit, reliable UDP or user-level TCP [13] can be used.

F. SIMULATION SETUP

To evaluate efficiency of event delivery in MEDYM, and compare it with existing approaches, we have developed a message-level, event-driven simulator.

1st. Architecture Approaches

We simulated two versions of the CBF approach: CBF_MST as in [6], which builds a single CBF tree as

the minimum spanning tree spanning all servers, and CBF_SPT as in [7], which builds CBF trees as source-specific shortest path trees, each rooted at a potential p-server. A challenge for CBF_SPT, not addressed directly in [7], is that a source-specific shortest-path tree in the overlay layer, constructed directly upon IP routing, is likely to degenerate into a star topology. In our simulation, we adopt a mesh-first approach for CBF_SPT, as in [10]. We first extract a 2-spanner [18] mesh from the underlying IP network, and then build SPT trees on top of the mesh.

We simulated Channelization approach as in [23]. We use Forgy K-means algorithm to cluster events into 50 event channels. This algorithm was found to generate the best clustering results among alternatives.

We simulated three versions of MEDYM, with the routing algorithms described in Section D. We call these MEDYM_MST, MEDYM_SPMST and MEDYM_BLSPMST.

2nd. Network Topology

We simulate a pub-sub network of 1000 servers. The underlying network topology is generated using the transit-stub model from GT-ITM Internet topology generator [3][1]. 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. Link latencies are randomly assigned as: 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, and the pub-sub servers are randomly attached to routers with 1ms latency.

3rd. Data Model

We assume that an event message has payload of 200 bytes and TCP/IP header of 44 bytes. Each MEDYM server has an ID of 16 bits. Since our focus is not on matching algorithms and our results are independent of them, for simplicity and without loss of generality, we use integers on [0, 100,000] as event and subscription values and perform only equality matching.

For comprehensiveness, we have experimented with various publication and subscription distributions, and found that CBF and MEDYM perform consistently while Channelization is sensitive to data distribution, but the difference is not large enough to change the comparative positions of the approaches. For space reasons, in this paper, we only present results for uniform distribution, in which publications and subscriptions are uniformly randomly generated and assigned to pub-sub servers. We expect it to provide the most basic and clear understanding of characteristics of the approaches. Results under other distributions are similar to the ones presented here and can be found in [5].

[1] We have also simulated with network topologies generated by Inet [17] and from Rocketfuel [25] and obtained similar results. They are omitted in this paper for space reasons.

G. Simulation Results

We evaluate event delivery efficiency along two dimensions: *server routing load*, which includes processing load and network bandwidth consumption at pub-sub servers, and *routing network efficiency*, which includes traffic load on underlying network links and the event delivery delay introduced.

We experiment with various levels of subscription selectivity, since different event delivery solutions can perform very differently for different selectivity situations, and it is desirable that an approach achieves consistent high efficiency under all circumstances. We define *matching ratio* to be the probability that an average event matches any subscription on a given pub-sub server. Low matching ratio means user subscriptions are highly selective, and vice versa.

1st. Server load

Event routing generates both processing load and network bandwidth consumption at pub-sub servers. Let us examine each.

1) Processing load

Because different approaches use event-forwarding methods, it is difficult to directly compare their computation cost. Instead, we use the number of events servers receive and route for as a metric to reflect the overall processing load. Figure 10 presents the average number of events a server receives in different approaches, normalized as a percentage of all events published. Note that this number can also be interpreted as the percentage of servers an average event is routed through, thus indicating the total routing load generated in the delivery process.

Figure 10 shows that servers in CBF and Channelization approaches receive more events than what they are interested in. In particular, a server that is interested in only 1% events receives 8% events in CBF_MST, 9% in CBF_SPT, and 29% in Channelization; a server interested in 10% events receives 29% events in CBF_MST, 34% in CBF_SPT, and 98% in Channelization. The curve for Channelization rises and saturates quickly, showing the ineffectiveness of clustering with less selective subscriptions: servers are likely to join more groups and receive many irrelevant events. The CBF approaches achieve higher routing accuracy than Channelization, but the proportion of irrelevant events received is still quite high when subscriptions are highly selective. This is

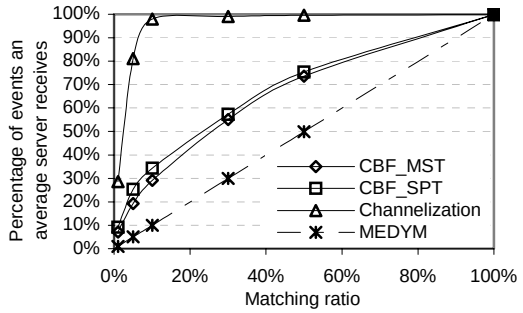


Figure 10. Average percentage of events servers receive.

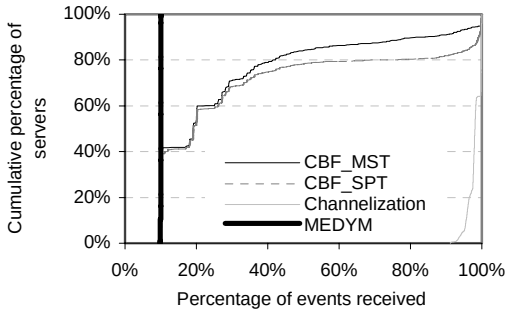


Figure 11. Cumulative distribution of percentage of events servers receive, with 10% matching ratio.

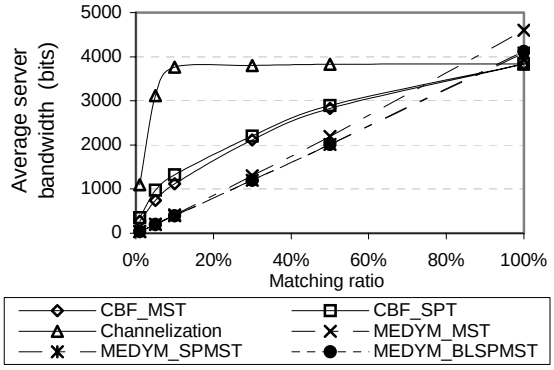


Figure 12. Average server bandwidth consumption.

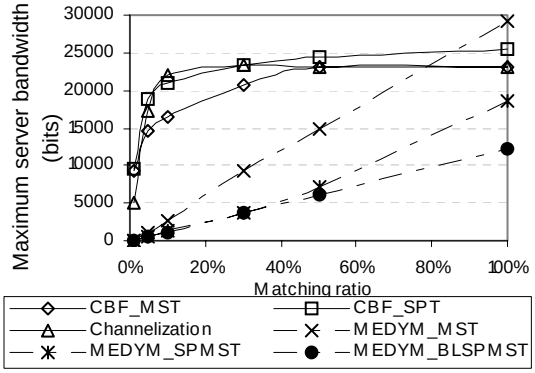


Figure 13. Maximum server bandwidth consumption.

because the number of intermediate servers an event traverses does not change with matching ratio, but with lower matching ratio, the probability that they happen to match the event forwarded is lower. In contrast, MEDYM servers always receive and route only the events that they are interested in. As this basic property of dynamic multicast is independent of specific routing algorithms, only one curve for MEDYM is shown.

Next, we look at how the processing load is distributed across pub-sub servers. Figure 11 shows the cumulative distribution of servers that receive certain number of events, when matching ratio is 10%. At the left end, it shows that every MEDYM server receives only the 10% of events that it is interested in. At the right end, it shows that almost all Channelization servers receive more than 90% of events each. In the middle, it shows load on CBF servers is highly imbalanced: about 40% servers receive only 10% events each, while 20% servers in CBF_MST and 10% in CBF_SPT receive more than 80% events each. This confirms our analysis in Section B that servers at the center of the CBF tree(s) are likely to route for many more events than their peripheral peers, most of which are irrelevant to their own interests. The heavy load on these servers can be bottleneck of system throughput and scalability.

2) Server bandwidth consumption

The average server bandwidth consumption in event routing is proportional to the average number of events each server receives, except for the destination list overhead in MEDYM schemes. Figure 12 shows that due to destination list overhead, MEDYM_MST consumes more bandwidth than CBF and Channelization when matching ratio is above 80%. It has highest destination list overhead among the MEDYM approaches, due to the long diameter of it multicast tree. MEDYM_SPMST and MEDYM_BLSPT has lower average destination list overhead, and outperform CBF and Channelization until matching ratio is above 90%. Interestingly, the points at which the destination list overhead hurts MEDYM efficiency relative to other systems is close to a point at which we may as well switch to broadcast.

Distribution of server bandwidth consumption can be different than that of number of events received. This is because bandwidth consumption also depends on server degree in the routing trees. Figure 13 shows maximum server bandwidth consumption with various matching ratios. Compared to Figure 12, three observations can be made. First, both CBF approaches perform much worse than for the average case, showing imbalanced network load across servers – servers at the network center not only receive more irrelevant events, but also have high degree in the CBF trees and forward more event copies. Second, despite its low routing accuracy, Channelization outperforms the CBF approaches under low matching ratios. This is because when clustering is effective, servers only join some of the multicast groups, and even

the most heavily loaded servers only receive events in their own groups. In contrast, all servers are always in all CBF trees. Finally, despite destination list overhead, MEDYM approaches significantly outperform the others, showing well balanced network load across MEDYM servers. This is for two reasons: the small number of events servers receive, as well as the high routing diversity in dynamic multicast. The advantage is especially significant with low matching ratios. With 1% matching ratio, the maximum bandwidth consumption in MEDYM_SPMST is only about 1% of that in CBF and 2% of that in Channelization. With 10% matching ratio, it is about 7% of that in CBF and Channelization. MEDYM_BLSPMST effectively reduces the maximum bandwidth consumption even further. Compared to these two algorithms, MEDYM_MST performs less well, due to its higher destination list overhead, and lower diversity between its non-source-specific multicast trees.

3) False positive routing in CBF_SPT

It was counterintuitive to us that with shortest path trees and per step filtering, CBF_SPT did not perform much better than CBF_MST. Interestingly, in our simulation, we found that CBF_SPT can have false positive event routing, i.e. sending event toward directions with no matching subscriptions, which is contradictory to the CBF design philosophy. An example is shown in Figure 14. For a similar reason as for IP multicast [12], we expect that accurate event routing in CBF_SPT (or other similar CBF approach with multiple routing trees) can only be achieved by also maintaining source information for subscriptions in the forwarding tables. The tradeoff between routing accuracy and state amount is an interesting topic that is beyond the scope of this paper.[2]

2nd. Network Efficiency

Next, we examine how event routing is carried out on the underlying network links.

1) Traffic load on network links

We compute total network resource usage as $\sum traffic_l * delay_l$ for all underlying network link l traversed in event delivery. $Traffic_l$ measures total amount of network traffic l carries, therefore taking into account destination list overhead in MEDYM. We define Normalized Resource Usage (NRU) as the ratio of the total network resource usage of an event delivery scheme over that of the *ideal multicast*. In *ideal multicast*, we assume there exists an IP multicast group for delivery of every event to and only to its matching servers, achieving high network efficiency.

Figure 15 compares the NRU of various approaches.

[2] We are in correspondence with the authors of the CBF_SPT approach and plan to investigate the tradeoff further together.

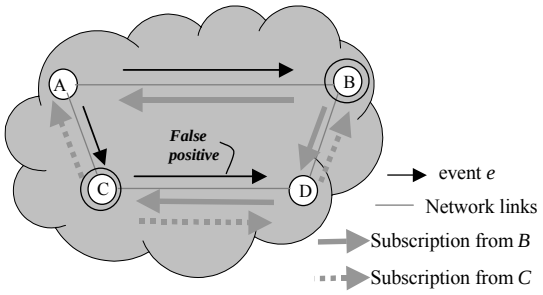


Figure 14. Example of false positive event routing in CBF_SPT. Server *B* and *C* broadcast their subscriptions along the shortest path trees rooted at each. Then, server *A* publishes event *e* that matches both subscriptions. Server *C* forwarding *e* to *D* is a false positive operation, because *C*-*D* is not on the shortest path from *A* to *B*, and *e* will not arrive reach *B* (or any matching server) in this direction.

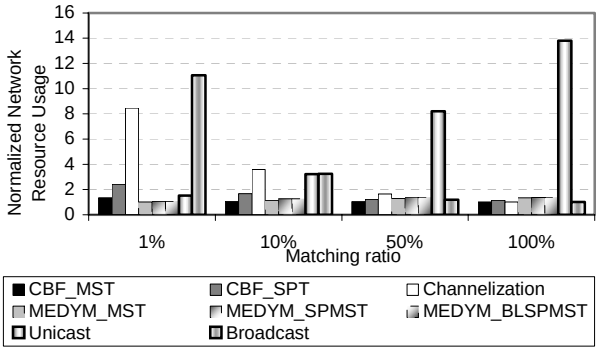


Figure 15. Normalized Network Resource Usage.

For reference, we also present results of two simple event delivery solutions: *unicast* and *broadcast*. Figure 15 shows that CBF_MST and MEDYM_MST achieve low network resource usage consistently. This is as expected, because their multicast trees are built with the goal of minimizing total communication cost. MEDYM_SPMST and MEDYM_BLSPMST, perform less well than MEDYM_MST, as they trade off network usage efficiency for computation efficiency and load balancing. However, when matching ratio is low, the advantage of MEDYM is apparent: at matching ratio of 1%, even with destination list overhead, all MEDYM approaches outperform CBF and Channelization. With low matching ratios, CBF_SPT performs even worse than unicast, clearly showing the effect of false positive routing. Results for Channelization are close to those of broadcast, showing that per-event content-based matching is indeed important to achieve high routing accuracy and efficiency. Under very high matching ratios, all approaches conform to broadcast, while the MEDYM schemes have higher resource usage due to destination list overhead.

Next, we look at how event routing traffic is distributed on underlying network links. We compute the stress of a network link by total amount of traffic it carries in the process of event delivery, and present the worst-case link stress in Figure 16. The results are

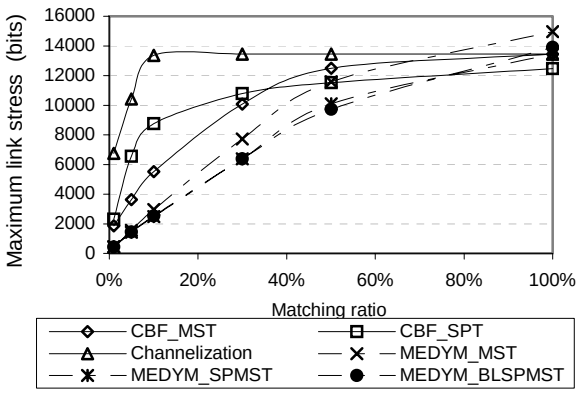


Figure 16. Worst-case link stress in event routing.

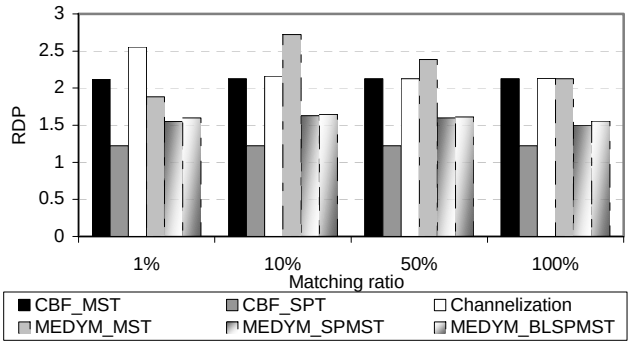


Figure 17. Relative Delay Penalty (RDP) in event routing.

similar to those of maximum bandwidth distribution, showing that MEDYM distributes event routing traffic across underlying network links in more balanced way that the other approaches.

2) Event delivery delay

In real-time pub-sub applications, it is desirable that events be sent to subscribers within short latency. Figure 17 presents the average Relative Delay Penalty (RDP) of event delivery in various approaches. RDP is computed as the ratio of event delivery delay in a pub-sub network to the delay of underlying IP routing between the p-server and s-server, averaged over all event routing paths. Using shortest-path trees, CBF_SPT always achieves lowest RDP, only slightly higher than 1. Note that its RDP should be 1 if the CBF trees were built directly on top of IP topology (rather than upon a 2-spanner mesh as in the simulation). MEDYM_SPMST and MEDYM_BLSPMST both achieve quite low RDP of about 1.5. Compared to MEDYM_MST, they benefit from the algorithm's biased selection of close destinations are next-hop servers. MEDYM_BLSPMST has slightly higher RDP because events are sometimes detoured for load balancing reasons.

3) Server location estimation

Finally, we compare the maintenance overhead and routing performance of event delivery in MEDYM using accurate server latency information versus using approximate server distance estimations. We simulated two scenarios: first, in a *measurement* network, each server probes one random server every 1 second, and broadcasts the probing results every 1000 seconds. We assume that in this way, the server distance information obtained is accurate. The bandwidth overhead generated is 210 Kbps. While such overhead is acceptable for servers in large-scale service networks, its $O(n^3)$ growth, where n is the number of servers in the network, is likely to prevent the network from scaling further.

We also simulate an *estimation* network as in [27], in which server locations are efficiently summarized by coordinates in a virtual d-dimensional Euclidean space, reducing total traffic overhead to $O(n^2)$. In our simulations, we randomly choose 50 servers as landmark nodes. Each server probes one landmark node every 20 seconds and computes its coordinates in a 20-dimensional Euclidean space. The coordinates are then broadcasted every 1000 seconds. The total bandwidth overhead is only 4.3 Kbps.

Table 2 presents the ratio of various metrics in the estimation network over those in the measurement network. Results show that the inefficiency introduced is largely reasonable, but higher with higher matching ratio. Overall, we expect server location estimation techniques offer promising opportunities to balance the tradeoff of routing performance and state maintenance cost in MEDYM. Because detailed results can be sensitive to real network environments and we plan to investigate this issue further through implementation.

3rd. Simulation results discussion

The simulation results in this section can be broadly summarized as follows:

Compared to existing approaches, event delivery in MEDYM generates lower routing load on pub-sub servers and underlying network links, and the routing load is distributed in a more balanced way. Advantage of MEDYM is most prominent with high subscription selectivity (low matching ratio), which is exactly the scenario when content-based pub-sub is most powerful and an efficient event delivery scheme is most needed. We also found that the destination list overhead in

Table 2. Network efficiency penalty for server distance estimation in MEDYM.

	Matching ratio	NRU	Average link stress	Max link stress	RDP
MEDYM_MST	1%	1.19	1.09	1.22	1.28
	10%	1.26	1.05	0.84	1.89
	100%	1.55	1.19	0.88	2.18
MEDYM_SPMST	1%	1.16	1.09	1.14	1.29
	10%	1.34	1.15	1.15	1.58
	100%	2.07	1.26	1.07	1.98
MEDYM_BLMST	1%	1.16	1.09	1.17	1.27
	10%	1.33	1.13	1.22	1.56
	100%	1.96	1.33	1.18	1.89

MEDYM is quite low and, except for unrepresentative situations of very low selectivity, is much more than outweighed by the advantages MEDYM provides. Among the three dynamic multicast routing algorithms, MEDYM_SPMST and MEDYM_BLSPMST perform very well under all metrics. MEDYM_MST results in higher destination list overhead and link stress, due to the "skinny" shape of the tree it constructs, and lower routing diversity it offers.

H. Conclusion and Future Work

In this paper, we presented design and evaluation of a new architecture approach for content-based pub-sub service networks, called MEDYM – Match Early with DYnamic Multicast. The high-level design of MEDYM follows the *End-to-end argument* in distributed system design [26]: it decouples the functionality of matching and routing in content-based pub-sub paradigm, placing expensive, application-specific event matching at end servers, and a relatively simple, generic address-based routing scheme in the network.

Simulation results show that MEDYM significantly improves event delivery efficiency, both in terms of server load and network efficiency, compared to existing design approaches. The overheads introduced in MEDYM are acceptable and more than outweighed by the benefits it brings. We also expect MEDYM network to be relatively easy to implement and maintain, compared to pub-sub networks with static event routing overlay networks.

As an event routing scheme designed for MEDYM, dynamic multicast is a highly flexible, robust and lightweight (stateless) multicast routing scheme. We believe its design and efficient implementation methods presented in this paper can be valuable for contexts other than pub-sub as well.

Overall, we believe that MEDYM presents a promising architectural approach for content-based pub-sub service network. We have implemented a prototype of MEDYM on the PlanetLab testbed [20]. Preliminary results have validated our simulation findings and can be found in [5]. We plan to deploy a publicly available pub-sub service network using MEDYM approach. This work has three goals: to provide a useful pub-sub service, to collect real pub-sub workload, which is in great need in this research area, and to further analyze and understand MEDYM performance in real networks.

The current MEDYM architecture and the techniques it uses are expected to scale to pub-sub service networks of thousands of servers. As each server is expected to serve a large number of end users, this scale is adequate for most known interesting pub-sub applications at realistic scales for the foreseeable future. Meanwhile, we are also exploring opportunities of further scaling MEDYM using hierarchical structure and content-partition techniques.

References

[1] 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 Symp on Principles of Distributed* Computing, 1999.

[2] A. *Bharambe, M. Agrawal, S. Seshan.* "Mercury: Supporting Scalable Multi-Attribute Range Queries". In *Proc. ACM SIGCOMM,* 2004

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

[4] F. Cao, J. P. Singh, "Efficient event routing in content-based publish-subscribe service network". In Proc. *IEEE INFOCOM* 2004.

[5] F. Cao, J. P. Singh, "MEDYM: Architectural design for content-based publish-subscribe service networks". Tech. Rep. *Princeton U.* 2004.

[6] A. Carzaniga, D. Rosenblum, and A. Wolf, "Design and evaluation of a wide-area event notification service," In *ACM TOCS.*, 2001.

[7] A. Carzaniga, A.L. Wolf, "A routing scheme for content-based networking". In *Proc. of IEEE INFOCOM 2003*.

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

[9] T. H. Cormen, C. E. Leiserson, R. L. Rivest, "Introduction to algorithms", MIT Press, 1990.

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

[11] 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.

[12] S. E. Deering, "Multicast Routing in Internetworks and Extended Lans". In *Proc. ACM SIGCOMM*, 25(1), Jan 1995

[13] D. ELY, S. SAVAGE et al. "Alpine: A user-level infrastructure for network protocol development." In *Proc. 3rd USITS,* 2001.

[14] R. Fielding, J. Gettys, et al. "Hypertext Transfer Protocol -- HTTP/1.1". RFC 2616, 1999.

[15] P. Francis, S. Jamin, et al, "IDMaps: a global internet host distance estimation service". In Proc. *IEEE/ACM Trans. Netw.*, 2001

[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] C. Jin, Q. Chen, and S. Jamin, "Inet: Internet Topology Generator," Technical Report, EECS Department, University of Michigan, 2000.

[18] G. Kortsarz, D. Peleg, "Generating Low-Degree 2-Spanners", In SIAM Journal on Computing, Volume 27, Number 5, pp. 1438-1456.

[19] A. Nakao, L. Peterson, "A routing underlay for overlay networks", In *Proc. ACM SIGCOMM* 2003.

[20] PlanetLab Testbed: http://planet-lab.org

[21] T. S. E. Ng and H. Zhang. "Predicting Internet Network Distance with Coordinates-Based Approaches." In Proc. *IEEE INFOCOM* 2002.

[22] R. Renesse, K. P. Birman, W. Vogels, "Astrolabe: A robust and scalable technology for distributed system monitoring, management, and data mining". In *ACM Trans. Comput. Syst*. 2003

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

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

[25] N. Spring, R. Mahajan, et al, "Measuring ISP Topologies with Rocketfuel". In SIGCOMM 2002.

[26] J. Saltzer, D. Reed, and D. Clark. "End-to-end arguments in system design". In *ACM Trans. Computer System*, 2(4), pp. 277--88, 1984.

[27] L. Tang, M. Crovella. "Virtual Landmark for the Internet", in Proc. *ACM SIGCOMM Internet Measurement Conference*, 2003

[28] TIBCO, http://www.tibco.com

[29] T. Wong, R. Katz, and S. McCanne. "An evaluation of preference clustering in large scale multicast applications," In Proc. IEEE INFOCOM 2000.

[30] Yahoo! Alert service. http://mobile.yahoo.com/wireless/alert

[31] T. Yan and H. Garcia-Molina. "SIFT---A tool for wide-area information dissemination". In *Proc USENIX TECH CONF*. 1995

[32] Y. Zhang, V. Paxson and S. Shenker, "The stationarity of internet path properties: Routing , loss, and throughput", Tech.Rep., ACIRI, 2000

[33] Y. Zhang, N. Duffield, et al, "On the constancy of internet path properties", In *Proc. Internet Measurement Workshop* 2001