

Signaling Protocols for Distributed Multicast-Based Publish/Subscribe Applications

Ping Ji, Zihui Ge, Jim Kurose and Don Towsley
 Computer Science Department,
 University of Massachusetts at Amherst,
 {jjiping,gezihui,kurose,towsley}@cs.umass.edu

Abstract—The publish/subscribe paradigm provides content-oriented data dissemination, where communication channels between content providers and content consumers are set up on the basis of interest matches between content provided by the publishers and content requested by the the subscribers. In this paper, we present a distributed active matchmaker architecture for publish/subscribe paradigm, where “active matchmakers”, consisting of end hosts and interior active routers or application level relay nodes, intercept and perform matches between publication announcements and subscription requests, and trigger establishing content based data forwarding controls. We explore two common signaling approaches, the broadcast subscription approach and the broadcast publication approach, and introduce another link-marking approach of accomplishing the functionality of active matchmakers. We develop a simple and fully distributed active matchmaker signaling protocol (AMSP) that dynamically adapts to applications’ publishing and subscribing behaviors and adjusts link-marking configurations so as to optimize the control overhead of conducting “matchmaking” among publishers and subscribers. We compare the performance of the three signaling approaches using simulations. Our simulation results show that AMSP significantly reduces system control message overhead in comparison to the existing broadcast publication or broadcast subscription approaches under various network and application configurations.

I. INTRODUCTION

The publish/subscribe (*pub/sub*) paradigm provides content-oriented data dissemination, where communication channels between content providers and content consumers are set up on the basis of interest matches between content being provided and requested. In such a paradigm, publishers (i.e., content providers) announce the data specifications, such as data topics or other content attributes, before data flow actually begins. Sim-

ilarly, subscribers (i.e., content consumers) reveal their interests so as to be able to receive data in which they have declared an interest. An *active matchmaker* system consisting of a set of distributed “mediators”, can then sit along the data dissemination paths between the publishers and subscribers, intercepting both publication announcements and subscription requests so as to facilitate communication channel setup and scoped content delivery (for example, by setting up data filters [15], [11]). Such an architecture is very useful in building large scale publish/subscribe applications such as distributed simulations [15] and large scale event notification systems [3], [4].

Due to their inherent reliance on group communication, most pub/sub systems [15], [12], [2], [13] are built on top of IP multicast or overlay multicast [8]. The active matchmaker system can therefore reside in end hosts and active routers or intermediate relay nodes, matching publication and subscription messages and triggering the setup of communication channels. To accomplish the functionality of content matches, two different types of mechanisms have been explored and applied in various pub/sub applications. In the first approach (which we will refer to as the “broadcast subscription” approach), subscription requests from content consumers are broadcast to all the publishers and registered at all the nodes along the paths. Data from the publishers are then forwarded toward interested receiver(s) only when they match some of the subscription requests. In the second approach (which we will refer to as the “broadcast publication” approach), publication announcements from content publishers are broadcast to all subscribers. The subscribers then forward their subscription requests only toward the publisher(s) who can provide the content of their interest. These matched subscription messages set up the data channels for content-based data dissemination. Recent work [6], however, suggests a third approach, in which publications and subscriptions meet in the middle of the network, and matches are detected where they meet. In that paper, the authors for-

malized an optimization problem for determining the optimal “meeting points” for the publication and subscription messages in order to minimize the amount of state stored in the network. Polynomial time algorithms were presented to solve the problem for both shared-tree based multicast and per-source-tree based multicast.

In this paper, we further investigate this third approach in which “matchmaking” between publication announcements and subscription requests are conducted not only at the edge but also in the middle of the network. We introduce the idea of a distributed active matchmaker system and describe its functionalities and its components. Furthermore, we present a simple link-marking scheme that forms the foundation of such an active matchmaker system. By assigning each link in the multicast network an attribute of *p-link* or *s-link*, and allowing only publication messages go through p-links and subscription messages go through s-links, the message overhead required for matching publications and subscriptions is significantly reduced. With guidance from the theoretical results from [6], we develop a distributed algorithm to accomplish optimized link-markings. Given that determining the optimal link-markings for per-source multicast trees can be complicated and requires global information, our scheme is fully distributed and works properly regardless of the underlying multicast structure, at the cost of achieving sub-optimal configurations in some cases. Moreover, we design an active matchmaker signaling protocol (AMSP), which integrates our link-marking algorithm. Through a simple signaling process, AMSP dynamically adapts to the publishing and subscribing behaviors of applications and adjusts link-marking configurations accordingly so as to optimize the control overhead of conducting matches among publication announcements and subscription requests. To evaluate the performance of our design, we implement the active matchmaker signaling protocol as well as the broadcast publication and the broadcast subscription approach in a simulation. Our simulation results demonstrate that the active matchmaker signaling protocol significantly reduces the control message overhead in pub/sub infrastructures in comparison to the existing broadcast publication/subscription approaches under various network and application configurations.

The remainder of this paper is organized as follows. In Section II, we present an active matchmaker network architecture for pub/sub applications, and describe the functionality of the active matchmaker component and its interactions with other components in the architecture. In Section III, we provide an overview of the three approaches, broadcast publication, broadcast subscription and link-marking approach in designing a matchmaker

signaling protocol. We present in detail our design of the active matchmaker signaling protocol in Section IV. We compare the performance of the three signaling approaches using simulations. Our simulation settings and simulation results are presented in Section V. Finally, in Section VI, we conclude our work and describe related future research.

II. A PUBLISH/SUBSCRIBE NETWORK ARCHITECTURE

As mentioned in the Introduction, most pub/sub systems are built on top of IP multicast or overlay multicast. Therefore, the end hosts, where the applications are running, together with the active routers or intermediate relay nodes along the multicast data dissemination paths can form a pub/sub infrastructure, where the participating *active nodes* cooperate with each other to exchange publication/subscription information, and to disseminate application data on the basis of their matched interests. We illustrate such a pub/sub network infrastructure in Figure 1.

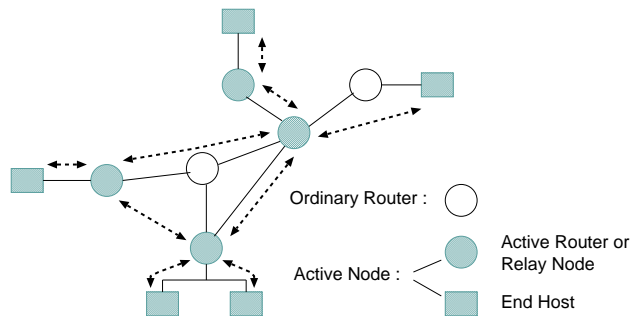


Fig. 1. A publish/subscribe network infrastructure

In Figure 1, the white circles represent ordinary routers, and the gray circles and gray squares represent the active nodes which can be active interior nodes (gray circles) or end hosts (gray squares). We note that the active interior nodes may be active routers in an active network [15] or relay nodes in an application overlay network [8]. In either case we will refer to these interior nodes as active routers, reflecting their functionality. In addition, in Figure 1, the solid lines indicate the multicast data dissemination paths, and the dashed lines indicate the signaling channels between the active nodes.

In a pub/sub network as shown in Figure 1, publishers and subscribers are *applications* sitting on the end hosts, where publishers announce the content specifications, such as data topics or some other content attributes, and subscribers reveal their subscription requests through the signaling channels. Together with the end hosts, active routers thereafter perform content matching based on

publication/subscription messages, and correspondingly set up content-oriented data forwarding channels. Data contents from publishers are then disseminated through the data forwarding channels toward the interested subscribers.

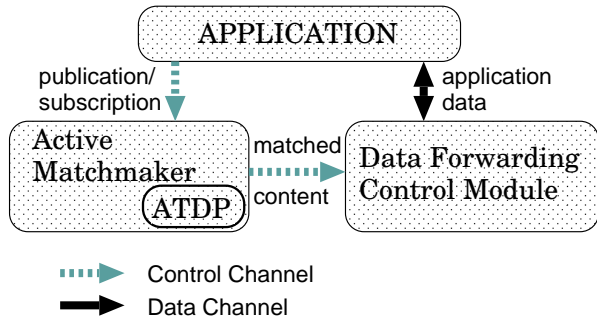


Fig. 2. The functionality components of an active node in pub/sub networks

Without loss of generality, we can further divide each active node in a pub/sub network into three components, according to their functionality: the application module, the active matchmaker module and the data forwarding control module, as illustrated by Figure 2. The **application** module is responsible for generating the publication announcements for publishers or subscription requests for subscribers and propagating these messages to the active matchmaker module through signaling channels. It also provides and/or consumes data contents that are sent through the data forwarding channels.

The **active matchmaker** component is responsible for distributing publication announcements and subscription requests and conducting content matches between them. Once a match is detected, it propagates such information to all active nodes along the multicast path from the related publishers to the subscribers. These nodes then set up their data forwarding channel accordingly. Since the design of an active matchmaker system and its signaling protocol is the primary focus of this paper, we will further discuss different approaches of realizing such an active matchmaker system in Section III.

Finally, the **data forwarding control** module utilizes underlying multicast infrastructures (IP multicast or application level multicast) to disseminate application data. The end hosts running the same application join one multicast group, and the underlying multicast topology can be either a single shared tree (e.g., CBT [1]) or per-source (publisher) trees (e.g., PIM-SM version 2 [5]). With information provided by the active matchmaker system, other than simply forwarding data, more controls are applied in the data forwarding module to further scope the content delivery to only interested receivers. This is usually

accomplished through content based routing [13] or data filtering [11], [15].

Thus, in a pub/sub network as shown in Figure 1, the active matchmakers sitting at each active node cooperate with each other and form a signaling overlay, through which publication announcements and subscription requests, as well as other control signaling messages, are propagated. Since the functionality of the active matchmaker system rely heavily on signaling processes, some signaling aspects such as detection of neighboring active nodes, reliable message transmission, should all be considered. An active topology discovery protocol (ATDP) [14] is designed to facilitate such signaling process in a multicast network. ATDP uses light-weighted source path messages (SPM) to track changes in the underlying multicast topology, and dynamically organizes active nodes into the signaling overlay where, for each per-source based multicast tree, reliable bidirectional connections between each active node and its parents and children are maintained. As a building block, ATDP can be easily incorporated into the active matchmaker system.

Having set aside concerns about the dynamic changing routing topology and the reliability of signaling transmission, in this paper, we concentrate on the mechanisms of how an active matchmaker utilizes signaling messages to accomplish interest matches among the publication announcements and subscription requests, and triggers setting up the data forwarding channels in pub/sub infrastructures. In the next section, we discuss different approaches of accomplishing interest matches.

III. THREE PUBLICATION/SUBSCRIPTION MATCHMAKING APPROACHES

As we illustrated in previous section, an active matchmaker component is responsible for matching interests between publication announcements and subscription requests, and setting up data forwarding channels among active nodes in a pub/sub infrastructure. In the following subsections, we explore three different approaches that can be used to accomplish these functionality.

A. Broadcast Subscription and Broadcast Publication Approaches

Previous studies [4], [9], [10] take for granted that either publication announcements or subscription requests should be broadcast to set up the data channels in a pub/sub infrastructure. In the existing **broadcast subscription** signaling approach, the subscription requests from all subscribers are propagated to all publishers throughout the multicast tree. For the purpose of data dissemination, the descriptors of interested contents that are

contained in these subscription messages are then stored at the data forwarding control modules at each active node regardless of whether there exists a publisher that can satisfy the requests. Unnecessary workload is thus brought into the data channel. By introducing the *matchmaker* functionality in broadcast subscription approach, content matches are executed at the publishers' end to match the publication announcements signaled from the application modules (publishers) and the subscription messages. The matched content is sent by publishers toward downstream active nodes and subscribers within the multicast tree. Active nodes along the multicast data dissemination paths can thus only store the matched content descriptors in the data forwarding control module, consequently reduce the resource consumption.

In another existing approach, **broadcast publication** approach, publication announcements from content producers are broadcast to all subscribers. The content matches are performed at the subscribers' end between the publication messages received and the subscription requests from the applications (subscribers). If matched interests are detected, signaling messages indicating the matched contents are generated and sent to the related publishers, meanwhile, being stored at the active nodes to set up the data forwarding controls along the way.

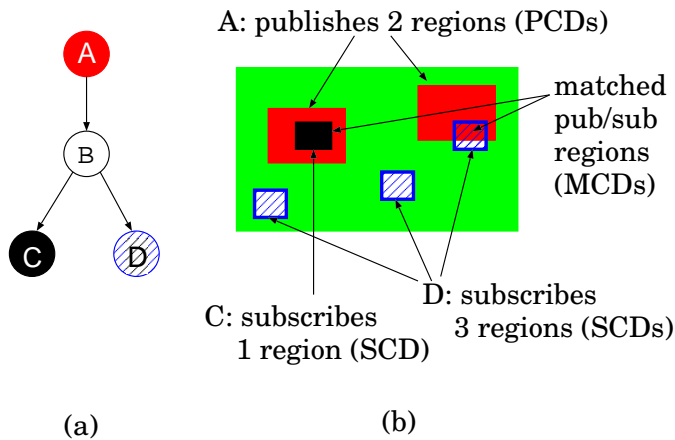


Fig. 3. An example of content descriptors (CDs) as geometric regions in a two-dimensional virtual world

An example may be helpful to better understand these two approaches. Consider a distributed simulation of a 2-dimensional virtual world [15]. A simulation unit (e.g., an entity in the simulation) located at a point in the world may only be interested in the events that occur within a certain distance from it. Thus the geometric description of this area is included in its subscription requests. Similarly a unit's publication on some geometric area indicates that its taking actions (i.e. pending publishing data) are regarding to this area. Figure 3 shows an example illus-

trating such a scenario. Figure 3(a) is a simple multicast topology with one publisher (*A*) and two subscribers (*C* and *D*). Figure 3(b) illustrates the 2-dimensional virtual world showing the regions within which *A* is generating data and the regions that *C* and *D* are subscribing to. Note that one publisher can announce multiple regions, and one subscriber can also subscribe to multiple regions. This is due to the fact that multiple simulation units can be simulated on the same host. In this paper, we refer to the descriptive information on predefined variables/categories (e.g., the geometric regions) that determines the boundaries of published/subscribed contents as a piece of **content descriptor (CD)**. Furthermore, we use **PCD** to indicate the publication content descriptor and **SCD** to indicate the subscription content descriptor.

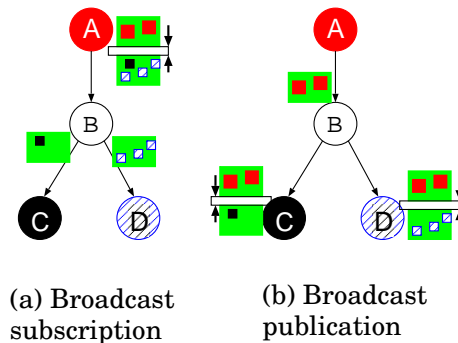


Fig. 4. Examples of broadcast publications and broadcast subscriptions approaches

We now use Figure 4 to illustrate the two broadcast approaches for the distributed simulation application described in Figure 3. The small rectangles in Figure 4 represent the CDs that *A* publishes or *C* and *D* subscribe to. Figure 4(a) indicates the broadcast subscription approach, in which content matchings are conducted at publisher *A*. Figure 4(b) indicates the broadcast publication approach, in which content matchings are performed by subscribers *C* and *D*. As a CD describes only certain attributes (e.g., the boundaries of a region in the virtual world simulation) of the published/subscribed contents, the size of each CD is approximately constant within one system. Consequently, the number of CDs being propagated and stored throughout the system is a good measure of the control overhead of a matchmaker system. In fact, we notice that, in the example that we just described, the broadcast subscription approach introduces more CDs to be propagated and maintained in the multicast tree than the broadcast publication approach.

B. An Active Matchmaker Approach Using Link Marking Scheme

A recent study [6] recommends that the publication announcements and the subscription requests should be allowed to meet in the middle of the network. Perform matches at their “meeting points” can reduce the amount of content state being maintained and propagated in the system. To accomplish this scheme, a link-marking method is used, where each directed link within the multicast tree is marked as either publish link (*p-link*) or subscribe link (*s-link*). All publish messages are required to be propagated over *p-links*, and all subscribe messages are required to be propagated over *s-links*. For each publish/subscribe message propagated over a link, the publication announcement/subscription request contained in the message is stored at the receiving node (active router or end host). Having ensured this, content matching between publication announcements and subscription requests can be performed not only at the end hosts but also at the active routers, where *p-links* meet *s-links*, and broadcast publication/subscription is avoid.

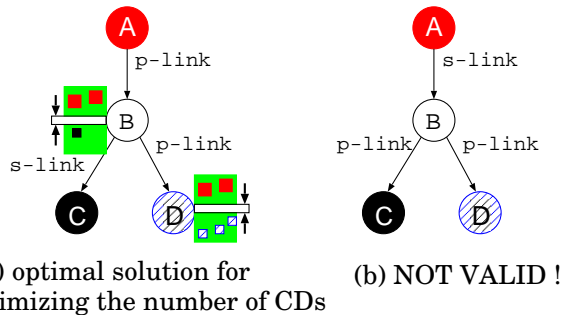


Fig. 5. Examples of using link-marking schemes

Figure 5(a) shows a link-marking configuration, for the scenario described in Figure 3, where links *AB* and *BD* are marked as *p-links* and *BC* is marked as an *s-link*. We observe that, this link-marking configuration results in a reduction of the total number of CDs. In fact, in terms of minimizing the content state, the link-marking configuration shown in Figure 5(a) is the optimal solution.

We observe that not all link-marking configurations are *valid*. To ensure that the system works correctly, the publish messages should be able to meet subscribe messages at some node along the data dissemination path from any publisher to any subscriber. This node can thereafter conduct matches between pub/sub messages and trigger the establishment of the data dissemination path for the matched contents. In terms of valid link-markings, this requires that, there exist a switch-over node whose upstream links are all marked as *p-link* and whose downstream links are all marked as *s-link*, along any data dissemination path

between a publisher and a subscriber. Figure 5(b) shows an example, where the link-marking configuration is invalid.

Using the link-marking algorithm, in this paper, we design an active matchmaker signaling approach, where each active matchmaker in a pub/sub network is responsible for marking related links as *p-links* or *s-links*, propagating the PCDs and SCDs correspondingly, and performing content matching as needed. Furthermore, the “active matchmakers”, those nodes to which publication announcements and subscription requests merge, signal the matched content descriptors (MCDs), which indicate an overlap between related PCDs and SCDs, upstream toward publishers and downstream toward subscribers.

We would like to base our design on the theoretical results of [6], in which, the authors formalized the optimization problem to determine the optimal link marking configuration to minimize the total content state being propagated and maintained in the system, and proposed polynomial time algorithms to solve the problem for shared-tree based multicast structure and per-source-tree based multicast structure separately. However, the *Min-Cost Matchmaker* algorithm, proposed in [6], for per-source based multicast topology requires more than local information to conduct the *optimal* link-markings. Deploying this algorithm into a distributed signaling protocol for per-source based multicast trees requires complicated control procedures. We desire distributed mechanisms in our design.

In addition, till now, we have not distinguished two types of costs associated with the matchmaking signaling protocols: the *storage overhead* corresponding to the number of content descriptors being maintained in the network, and the *control message overhead* corresponding to the signaling rate (in messages per second) of signaling messages transmitted to perform matches between publishers and subscribers. Consider a publisher publishing a certain number of PCDs, maintaining these content descriptors at the active matchmaker system on it introduces storage overhead that is proportional to the number of PCDs that the publisher publishes. If message overhead is considered, not only the total number of PCDs but also the frequency with which these PCDs are updated should be considered. We may choose either metric of cost, the total number of CDs or the aggregated frequency, to address different optimization problems. In this paper, we focus on the problem of minimizing control message overhead.

In the next section, we propose our design of the active matchmaker signaling protocol in detail.

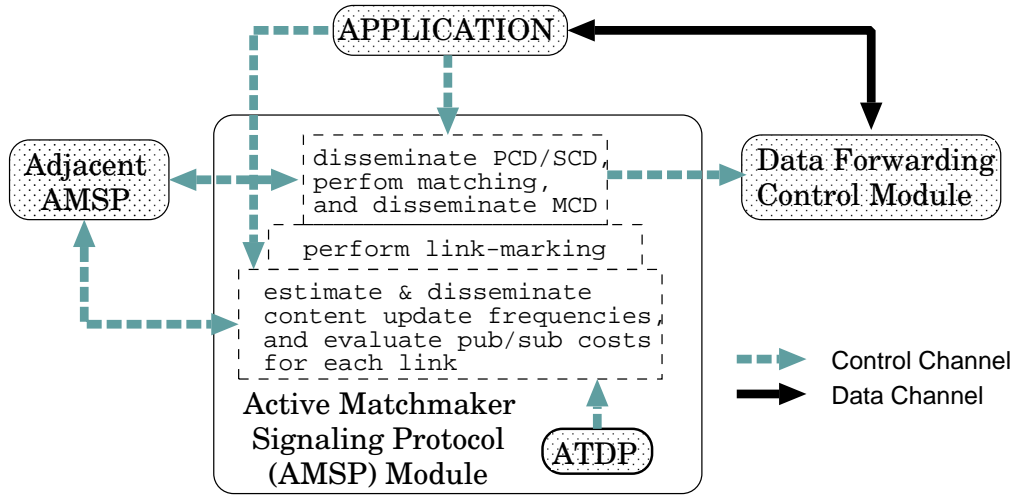


Fig. 6. Architecture of active matchmaker signaling protocol (AMSP) module

IV. ACTIVE MATCHMAKER SIGNALING PROTOCOL DESIGN

In this section, we design an active matchmaker signaling protocol (AMSP) based on the link-marking scheme.

The control message overhead of a pub/sub system is, in fact, the aggregated number of control messages propagated over each link of a multicast network. For each link, there is a *publish cost* associated with sending PCDs, and a *subscribe cost* associated with sending SCDs. Thus, the task of the link-marking algorithm is to let each link know the potential publish cost and subscribe cost associated with it, so that link marking can be conducted locally for each link by choosing the marking with smaller cost.

Being aware of this, we design AMSP to be consisting of three sub-components, as shown in Figure 6.

1. AMSP estimates the PCD/SCD update frequency based on the publication announcements/subscription requests signaled by the applications (publishers/subscribers), and disseminates the estimated content update frequencies to AMSP(s) sitting at an adjacent active node. In addition, based on the PCD/SCD update frequencies, each AMSP evaluates the publish/subscribe costs associated with each link connecting to it.
2. Based on the publish/subscribe cost, AMSP use a link marking algorithm to assign *p-link* or *s-link* attribute for each connected link.
3. When link-markings are assigned, AMSP propagates the publish announcements (PCDs) and the subscribe requests (SCDs), performs matches between PCDs and SCDs, and signals the MCDs correspondingly.

In the following three subsections, we examine each of these three parts of AMSP in detail.

A. Estimating and Disseminating CD Update Frequency

Let f_x denote the publish content update frequency, the number of PCDs updated per unit time (e.g., minute) at publisher x , and g_x the subscribe content update frequency, the number of SCDs being updated per unit time (e.g., minute) at subscriber x . Note, each end host (publisher/subscriber) may have multiple objects publishing/subscribing contents, and the content update frequencies of different objects may differ. Therefore, f_x and g_x are determined by the content update rates of all objects at publisher/subscriber x respectively. In addition, when a new object joins or an old object leaves, it is counted as one CD update.

To capture the dynamics of the application, f_x and g_x are periodically updated at the AMSP component of each publisher or subscriber. The update period is t . To avoid synchronization between different publishers and subscribers, a small random value is added to t . At the end of each interval, we use exponential smoothing to compute the average frequencies as follows,

$$f_x^t = (1 - \alpha)f_x^{t-1} + \alpha \frac{\text{No. of PCD updates}}{\text{interval length}} \quad (1)$$

$$g_x^t = (1 - \alpha)g_x^{t-1} + \alpha \frac{\text{No. of SCD updates}}{\text{interval length}} \quad (2)$$

where, α is the smoothing parameter.

If the new estimate is significantly different from the previous announced value, the AMSP at the end host announces this new estimate to the active nodes within the same multicast network through a signaling message, COST_ANNOUNCE, which has the following format:

- COST_ANNOUNCE(<sdr rcvr>, originator, psFlag, value)

Table I describes each of the message parameters.

Parameter	Description
sdr	active node sending the message
rcvr	active node receiving the message
originator	publisher/subscriber, x , initiates the message
psFlag	1 (f_x update), 0 (g_x update)
value	new value of f_x or g_x

TABLE I

PARAMETER DESCRIPTIONS OF COST_ANNOUNCE MESSAGE

On receiving a COST_ANNOUNCE message from the application module or from an adjacent AMSP, indicating a new f_x or g_x , AMSP propagates this COST_ANNOUNCE message to the adjacent upstream/downstream active node(s) along the multicast data dissemination paths.

At each active node, AMSP maintains a table of content update frequencies, f_x and g_x , for end host x , which is used to compute the publish and subscribe costs for related links. When a COST_ANNOUNCE message is received, AMSP updates the corresponding table entry to the new value of f_x or g_x contained in the message, and computes the publish and subscribe costs for each link related to end host x .

We define $C_i^{(p)}$ to be the **publish cost** of link i and $C_i^{(s)}$ to be the **subscribe cost** of link i . They are given as

$$C_i^{(p)} = \sum_{x \in P_i} f_x \quad (3)$$

$$C_i^{(s)} = \sum_{x \in S_i} g_x \quad (4)$$

where P_i denotes the set of publishers that disseminate data through link i , and S_i denotes the set of subscribers that receive data sent through link i . $C_i^{(p)}$ and $C_i^{(s)}$ are estimates of the update frequencies of PCDs and SCDs that either are or could be propagated over link i .

The values of $C_i^{(p)}$ and $C_i^{(s)}$ are used by the link-marking algorithm as we describe in the next subsection.

B. Link Marking Algorithm

Recall that in the signaling overlay, each link connecting two active nodes is bi-directional. We define the node sitting at the upstream end of the data flow as the *upstream node* of that link, and the node sitting at the downstream end of the data flow as the *downstream node* of that link. The upstream-downstream relationship of the two nodes can be reversed for different multicast sources. To ensure the link-marking scheme works, the marking attribute of a link should be maintained at both ends of the link.

It was proved in [6] that, setting the marking attribute of each link i to correspond with the smallest of $C_i^{(p)}$ and $C_i^{(s)}$ ensures a valid overall link-marking configuration and achieves the global optimal marking in terms of the total cost in the shared multicast tree. However, this is not true for per-source multicast networks, where assigning the attribute to each link locally based on its publish cost and subscribe cost may result in an invalid link-marking configuration. The algorithm to find the optimal valid link-markings for per-source based multicast tree is complicated and difficult to implement in a distributed manner.

Because of these complications, we place emphases on obtaining a valid link-marking configuration and not on obtaining the optimal marking when designing a distributed link-marking algorithm. We correspondingly propose a distributed *top-down marking* scheme that can be used in both per-source tree based multicast and shared tree based multicast networks.

In the top-down marking algorithm, the marking attribute for each link is independently determined only by the AMSP at the upstream node of a link. For each node, we define a link i as an *upstream link* of link j when data received from i is disseminated over j within some multicast tree(s). For instance, in the multicast tree shown in Figure 3(a), link AB is an upstream link of link BC and link BD .

The top-down link marking algorithm is given in Figure 7. According to this algorithm, a link is forced to be an *s-link* if any of its upstream links is marked as *s-link*. Otherwise, the link-marking decision is made by comparing the publish cost and the subscribe cost of this link. Furthermore, to avoid frequently changing link-marking attributes, a hysteresis threshold, δ , is introduced in the algorithm. The link-marking algorithm is executed every time the publish cost $C_i^{(p)}$ or the subscribe cost $C_i^{(s)}$ is changed for link i .

```

for an outgoing link  $i$ :
1.   if (any upstream link of link  $i$  is s-link)
2.     label link  $i$  as an s-link;
3.   else {
4.     if ( $(C_i^{(s)} - C_i^{(p)}) > \delta$ )
5.       label link  $i$  as p-link;
6.     if ( $(C_i^{(p)} - C_i^{(s)}) > \delta$ )
7.       label link  $i$  as s-link;
8.   }

```

Fig. 7. Top-down marking algorithm

When a link-marking is changed, the upstream node of link i notifies the downstream node of link i about this change. Two signaling messages, CHANGE_MARKING

message and CHANGE_MARKING_REPLY message, are designed for this purpose. They have the following formats

- CHANGE_MARKING(<sdr rcvr>, newMarking, [pcdList]),
- CHANGE_MARKING_REPLY(<sdr rcvr>, newMarking, [scdList]).

Table II describes the meaning of the parameters in these two signaling messages. We will describe the usage of *pcdList* and *scdList* in the next subsection.

Parameter	Description
sdr	active node sending the message
rcvr	active node receiving the message
newMarking	new marking of a link
pcdList	list of PCDs of a link
scdList	list of SCDs of a link

TABLE II

PARAMETER DESCRIPTIONS OF CHANGE_MARKING AND CHANGE_MARKING_REPLY MESSAGES

The CHANGE_MARKING message is used by the upstream node of a link to announce the change of link-marking attribute to the downstream node. The CHANGE_MARKING_REPLY is used by the downstream node of a link to acknowledge a CHANGE_MARKING message sent by the upstream node.

The distributed link-marking algorithm proposed in Figure 7 yields the optimal solution for the *Min-Cost Matchmaker* problem of shared multicast tree. It does not ensure the optimal solution for per-source based multicast trees. However, we will observe in Section V that this algorithm can still substantially reduce the control message overhead in comparison to the broadcast publication or broadcast subscription approaches.

C. Propagating Content Descriptors Using Link-Markings

AMSP forwards content descriptors based on the link-marking of each link. PCDs are forwarded only through p-links and SCDs are forwarded only through s-links. A signaling message, CD_UPDATE, is therefore used to propagate the content descriptors over the corresponding links. The format of CD_UPDATE message is

- CD_UPDATE (<sdr rcvr>, originator, psmCDFlag, [cdList]),

Table III describes the meaning of the parameters in this message.

Parameter	Description
sdr	active node sending the message
rcvr	active node receiving the message
originator	publisher, and/or subscriber
psmFlag	1 (PCD), 0 (SCD), 2 (MCD)
cdList	list of CDs sent by originator

TABLE III

PARAMETER DESCRIPTIONS OF CD_UPDATE MESSAGE

In our design of AMSP, the PCDs/SCDs are maintained at the downstream/upstream node of a link respectively. When the link-marking changes, the corresponding PCDs/SCDs is propagated to the *correct* end-node of a link. Therefore, when the upstream node of *i* changes the marking attribute of *i* from *s-link* to *p-link*, it also gathers all PCDs related to *i*, and sends the *pcdList* to the downstream node of *i* with the CHANGE_MARKING message. Similarly, when the downstream node of link *i* receives a CHANGE_MARKING message informing that the attribute of link *i* is changed to *s-link*, the downstream node of *i* gathers all SCDs related to *i*, and sends the *scdList* to the upstream node of *i* with the CHANGE_MARKING_REPLY message.

Having gathered all needed PCDs/SCDs information, the active matchmakers, sitting on the “meeting points” of p-links and s-links, conduct content matchings between PCDs and SCDs. If matched content is detected, the CD_UPDATE message is used to signal the MCDs to the data forwarding control component and to upstream and downstream active nodes along the multicast data dissemination paths. Note, since each MCD associates with one publisher and one subscriber, when a CD_UPDATE message carries an MCD, the *originator* field should be the publisher and the subscriber pair.

In a pub/sub network, when the interest of a publisher/subscriber is changed, the related *old* content descriptors stored at each active node should be removed. To remove these out-of-date content descriptors, there are two schemes, the soft-state scheme and the hard-state scheme. In the soft-state scheme, publishers and subscribers periodically refresh their publication announcements and subscription requests. A content descriptor is removed if a refresh message is not received before timeout. In the hard-state scheme, publishers and subscribers assign a unique ID for each PCD or SCD, and are responsible for explicit withdraw or implicit withdraw (by sending update using the same ID) of the content descriptors. The withdraw of a PCD or SCD may invalidate one or more MCD(s). Thus, the matchmaking nodes at the

“meeting point” is responsible to withdraw those MCDs. The choice of using either scheme in AMSP with different application behaviors remains an interesting research problem. In addition, we note that, choosing the soft-state scheme or hard-state scheme to remove content descriptors maintained at the active nodes is also a common problem for other signaling approaches (e.g., broadcast publication approach and broadcast subscription approach).

We implemented AMSP, as well as the broadcast subscription signaling approach and the broadcast publication signaling approach in a simulation. In the next section, we present our simulation settings and the simulation results.

V. SIMULATION AND PERFORMANCE EVALUATION

In this section, we present our simulation studies on the performance of the proposed signal protocol. We will first describe our simulation settings and then present the simulation results.

A. Simulation Settings

In our simulations, we assume a hierarchical network topology represented by a transit-stub structure. We use the topology generator developed by Georgia Tech [7] to generate such network topologies. Each network generated contains 100 nodes, 4 of which are transit nodes, forming one transit domain, and the remainder stub nodes forming 12 stub domains. We assume that all transit nodes and stub nodes are active and, hence, are able to maintain publication and subscription state and perform edge marking and interest matching.

In each set of simulations, we randomly choose publishers and subscribers from the stub nodes and do not exclude the possibility that a publisher and a subscriber are collocated on the same node. Once the physical topology is generated and a set of publishers and subscribers are selected, we construct the multicast data dissemination topology. To simulate a shared multicast tree, we choose one of the transit nodes as the “core” and use CBT as the underlying multicast routing protocol; to simulate per-source multicast trees, for each publisher, we build a multicast tree rooted at the publisher composed of the shortest paths toward all the subscribers.

In our simulation, we concentrate on simulating the applications such as distributed simulation simulating multi-dimensional virtual world, and we build the following application model to capture the application behaviors.

Each application (i.e., publisher/subscriber) runs multiple *simulation entities* (e.g., tanks in distributed game). Each simulation entity owns a distinct PCD or SCD, indicating its publishing content or subscription interest. To

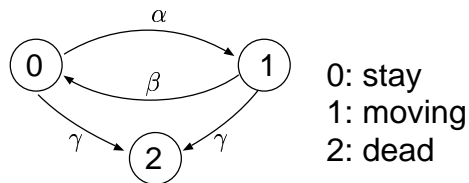


Fig. 8. Life cycle of simulation entities

each application x , an initial population of “simulation entities”, N_x , is assigned, where N_x is uniformly chosen between 50 and 150. New entities are introduced to each application according to a Poisson process with arrival rate $\lambda = 1/60$, (i.e., on average one new simulation entity per minute is added to each application). Each entity has a 3-state life cycle, represented by the Markov-model in Figure 8. When in state 0, an entity continues with its previous announced PCD or SCD; when in state 1, an entity intermittingly updates its PCD or SCD, with the intervals randomly drawn according to a Gaussian distribution, whose mean in turn is uniformly distributed between a lower bound, $I_{min} = 3$ seconds, and an upper bound, $I_{max} = 120$ seconds; when entering state 2, an entity removes itself from the application.

Consider the case where simulation entities move around in a multi-dimensional virtual world. When an entity is in state 0, it remains at some fixed location. When it transits to state 1, the entity is in motion, and its PCD/SCD is updated along with its movements. Accordingly, the speed with which an entity moves determines its PCD or SCD update rate. When an entity is “terminated”, it transits to state 2, withdraws its previously announced PCD or SCD and removes itself from the simulation application.

Since different entities can have different behaviors, for each entity, we randomly assign values to its transition rate α (from state 0 to state 1), β (from state 1 to state 0) and γ (from state 0, 1 to state 2). In our simulation, we use the following values:

$$\alpha = 10^t, t \in [-1, -3]$$

$$\beta = 10^t, t \in [-2, -4]$$

$$\gamma = \lambda/N_x = 1/60/N_x$$

where, t is uniformly chosen within the range.

Using this application model, we simulate the three signaling approaches, broadcast subscription, broadcast publication and AMSP. We present our simulation results in the next subsection.

B. Simulation Results

The performance metric of interest to us is the control message rate (messages per second). This corresponds to

the number of control messages being signaled throughout the network divided by the simulation time. In particular, we focus on the number of messages sent to accomplish content matches, i.e., the messages carrying PCD/SCD updates, messages required for distributing f_x , g_x , and messages used to exchange link-marking decisions. However, we do not include the messages generated after a match between a PCD and an SCD is detected. This is because the latter signaling overhead is the same as that in the broadcast publication and the broadcast subscription approach and is inevitable for content based data forwarding — when a match is detected, the corresponding MCD is forwarded upstream towards the related publisher and downstream towards the related subscriber in all three approaches.

We simulated the three signaling approaches discussed in this paper — broadcast publication (BP), broadcast subscription (BS), and AMSP. For each experiment, we first generate a network topology, randomly select publishers and subscribers and determine all the user publishing and subscribing behaviors in advance. We then simulate these three approaches on the same set of configuration. In Figure 9, we plot the smoothed control message rate (with windows size 30-second) over time for different pub/sub network architectures, where there are 20 publishers and 20 subscribers in the system, using per-source multicast trees. We observe a significant reduction on the total control messages required using AMSP. With the exception of some fluctuations at the beginning of the simulation, the total number of messages for PCD/SCD updates using either broadcast scheme is around 1400 per second, while the total number of control messages adapting a link-marking active matchmaker is less than 300 per second, 10% of which are used to perform the link-markings and the remainder for disseminating PCD/SCD updates.

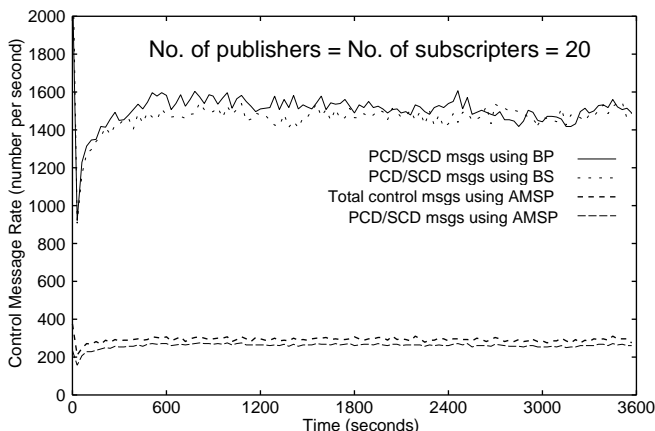


Fig. 9. Comparison of different signaling architectures on running control message rate

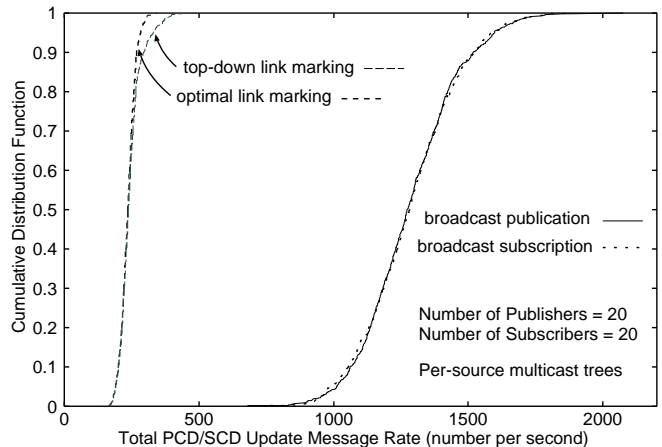


Fig. 10. Evaluation of top-down link-marking algorithm in per-source tree multicast

As discussed in Section IV-B, the top-down marking scheme used in implementing AMSP does not ensure that the resulted marking is optimal when the underlying multicast topology uses per-source trees. Thus, one would like to know how good is the solution found by the top-down marking scheme comparing to the optimal solution. However, since the optimal marking algorithm presented in [6] is difficult to be realized in a distributed manner, we generate static network instances where we base on the application module described in Section V-A to randomly assign a fixed PCD/SCD update rate for each publisher or subscriber, and compute the optimal link-marking using the algorithms provided in [6]. We then compare the corresponding total PCD/SCD update message rate associated with the top-down marking configurations and the corresponding optimal marking solutions. Figure 10 shows the cumulative distribution function of the total PCD/SCD update message rate for 1000 randomly generated sample networks, each of which has 20 publishers and 20 subscribers. We observe that the proposed top-down marking algorithm can find solutions very close to the optimal link marking solution, while both of the marking algorithms significantly outperform the two broadcast approaches.

We also investigate scenarios with an asymmetric number of publishers and subscribers. In Figure 11, we fix the number of publishers to 20 and vary the number of subscribers from 5 to 50. Each point in the graph corresponds to the average of 10 independent experiments and we also plot their 90 percentile confidence intervals. Figure 11 (a) presents the results where the underlying multicast routing uses per-source trees and Figure 11 (b) presents the results where a single shared-tree multicast tree is used. In both Figures, we observe an increasing

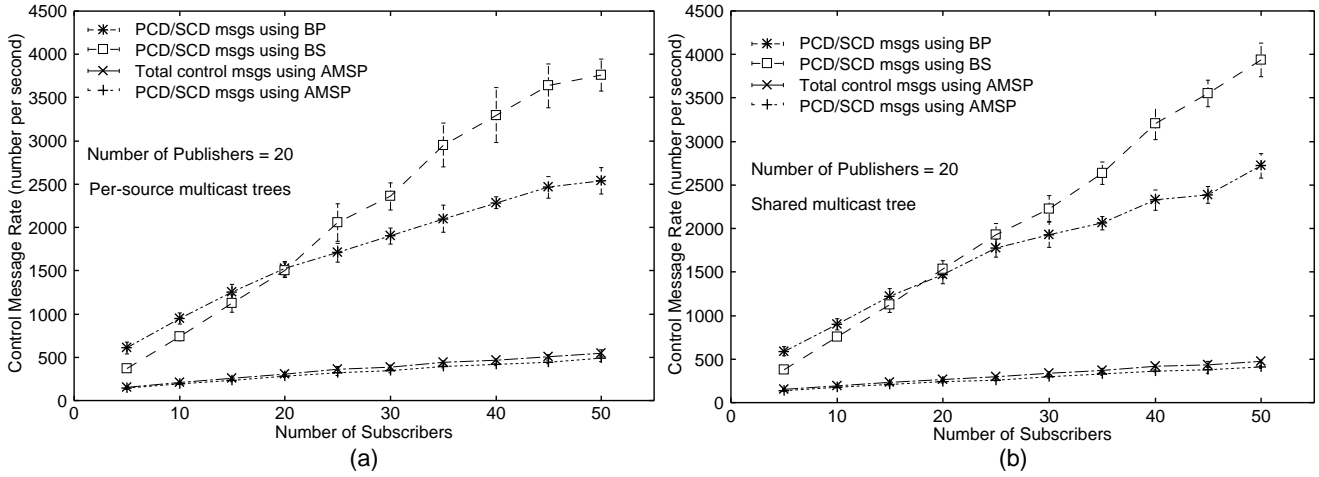


Fig. 11. Impact of number of subscribers on signaling overhead for per-source multicast trees (a) and shared multicast tree (b)

number of total control messages for all three approaches with an increasing number of publishers. When the number of publishers exceeds the number of subscribers, the broadcast subscription approach outperforms the broadcast publication approach, otherwise, the broadcast publication approach outperforms the broadcast subscription approach. This is explained by the symmetric behavior of publisher/subscriber application. Moreover, the slope of increase associated with the broadcast subscription approach is lower than that with the broadcast publication approach, indicating that the broadcast subscription approach scales better with the number of subscribers. We also conduct symmetric experiments with fixed number of subscribers and varying number of publishers. The result is also symmetric to Figure 11 with the curves for the two broadcast approaches switched, indicating that the broadcast publication approach scales better with the number of publishers. In all cases, AMSP consumes the lowest number of total messages and shows the least trend to increase, demonstrating its capability of handling both a large number of publishers/subscribers.

Comparing Figure 11 (a) and Figure 11 (b), we observe that the average control message rate for shared multicast tree is only slightly higher than that for per-source multicast trees (within confidence intervals). Intuitively, a per-source tree multicast topology is more efficient than a shared multicast tree in term of the size of their data dissemination trees. However, since we assume a hierarchical network represented by a transit-stub structure, and use a transit node as the core in building shared multicast tree, the two kinds of multicast trees constructed differ to each other by only a few links. This explains the fact that the difference between their average control message overheads is so small.

Last, we investigate the impact of the dynamics of ap-

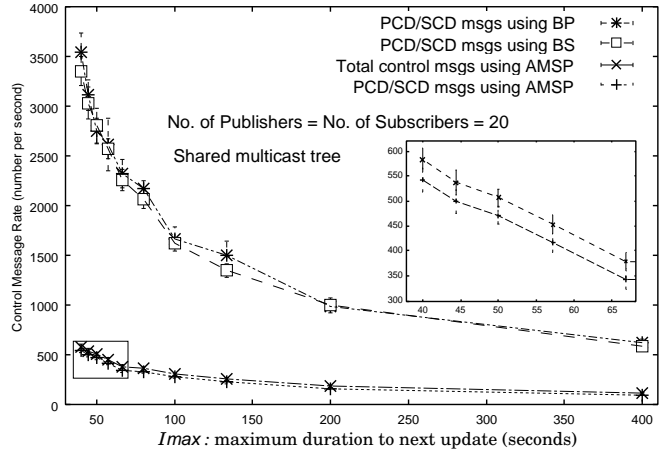


Fig. 12. Impact of application update frequency on signaling overhead for per-source multicast trees

plication behaviors on the performance of the three different signaling approaches. As described in the simulation settings, the dynamics of an application's publishing or subscribing behavior are captured by the rate with which its simulation entities update their previously announced PCD or SCD. In our simulation, this is controlled by I_{min} and I_{max} , which determine the bounds of the mean duration between two consecutive updates. The smaller I_{min} and I_{max} are, the more frequently PCDs/SCDs are updated. Figure 12 presents the result where there are 20 publishers and 20 subscribers in the system and the underlying multicast topology uses a shared tree. By reducing I_{max} from 400 seconds to 40 seconds while keeping $I_{min} = 3$ seconds, the frequency of updating applications' publication announcements and subscription requests is increased. From Figure 12, we observe that AMSP does not suffer much with small value of I_{max} , while the average control message rate of either broadcast publication or

broadcast subscription increase dramatically when I_{max} decreases. Figure 12 also shows an enlarged graph on the average control message rate of AMSP when I_{max} is small. We observe that although the control messages required for propagating PCD/SCD increases with decreasing I_{max} , the control messages required for performing link-markings (indicated by the differences of the two curves) stay almost constant. This demonstrates a good scaling behavior of AMSP with respect to the dynamics of applications' publication announcements and subscription requests.

VI. CONCLUSION

In this paper, we have presented a distributed active matchmaker structure for multicast-based pub/sub applications. Consisting of all participating end hosts and interior active routers along the data dissemination paths, the active matchmaker system intercepts publication announcement from publisher and subscription requests from subscriber, conducts matches between them and triggers setting up content based data forwarding controls along the underlying multicast data dissemination paths. We discussed two common signaling approaches, broadcast publication and broadcast subscription, and introduced another link-marking approach of accomplishing the functionality of the active matchmaker system. Furthermore, we developed a simple and fully distributed active matchmaker signaling protocol (AMSP) that dynamically adapts to applications' publishing and subscribing behaviors and adjusts link-marking configurations so as to optimize the overall control overhead of conducting "matchmaking" among publishers and subscribers. To evaluate the performance of our design, we implemented AMSP as well as the broadcast publication and the broadcast subscription approaches in a simulation. Our simulation results demonstrate that AMSP significantly reduces the system control message overhead in comparison to the existing broadcast publication or broadcast subscription approaches under various network and application configurations.

In our future work, we would like to implement and integrate the active matchmaker signaling protocol into our existing active-network-facilitated large-scale distributed simulation system [15]. We will evaluate the performance of the proposed active matchmaker structure under realistic network and application configurations.

REFERENCES

- [1] T. Ballardie, P. Francis, and J. Crowcroft. Core based trees (CBT), an architecture for scalable inter-domain multicast routing. In *ACM SIGCOMM*, 1993.
- [2] Guruduth Banavar, Tushar Chandra, Bodhi Mukherjee, Jay Nagarajarao, Robert E. Strom, and Daniel C. Sturman. An efficient multicast protocol for content-based publish-subscribe systems. In *Proceedings of the 19th IEEE International Conference on Distributed Computing Systems*, Austin, Texas, June 1999.
- [3] Luis Felipe Cabrera, Michael B. Jones, and Marvin Theimer. Herald: Achieving a global event notification service. In *Proceedings of the Eighth Workshop on Hot Topics in Operating Systems (HotOS-VIII)*, Elmau, Germany, May 2001.
- [4] A. Carzaniga, D. Rosenblum, and A. Wolf. Design and evaluation of a wide-area event notification service. *ACM Transactions on Computer Systems*, 19(3), Aug 2001.
- [5] Bill Fenner, Mark Handley, Hugh Holbrook, and Isidor Kouvelas. Protocol independent multicast - sparse mode (pim-sm): Protocol specification (revised), March 2002. INTERNET DRAFT, <http://www.ietf.org/internet-drafts/draft-ietf-pim-sm-v2-new-05.txt>.
- [6] Zihui Ge, Ping Ji, Jim Kurose, and Don Towsley. Min-cost matchmaker problem in distributed publish/subscribe infrastructures. Technical report, University of Massachusetts at Amherst, 2002.
- [7] Georgia Tech. *Modeling Topology of Large Internetworks*. <http://www.cc.gatech.edu/fac/Ellen.Zegura/graphs.html>.
- [8] Yang hua Chu, Sanjay G. Rao, and Hui Zhang. A case for end system multicast. In *Proceedings of ACM SIGMETRICS*, Santa Clara, CA, June 2000.
- [9] G. Mühl, L. Fiege, and A. Buchmann. Filter similarities in content-based publish/subscribe systems. In *International Conference on Architecture of Computing Systems (ARCS)*, 2002.
- [10] Gero Mühl. Generic constraints for content-based publish/subscribe. In *Proceedings of the 6th International Conference on Cooperative Information Systems (COOPIS)*, 2001.
- [11] Manuel Oliveira, Jon Crowcroft, and Christophe Diot. Router level filtering for receiver interest delivery. In *Second International Workshop on Networked Group Communication (NGC)*, Palo Alto, California, Nov. 2000.
- [12] Lukasz Opyrchal, Mark Astley, Joshua S. Auerbach, Guruduth Banavar, Robert E. Strom, and Daniel C. Sturman. Exploiting IP multicast in content-based publish-subscribe systems. In *Middleware*, pages 185–207, 2000.
- [13] Antony I. T. Rowstron, Anne-Marie Kermarrec, Miguel Castro, and Peter Druschel. SCRIBE: The design of a large-scale event notification infrastructure. In *Networked Group Communication*, pages 30–43, 2001.
- [14] Jonathan K. Shapiro, Jim Kurose, Don Towsley, and Stephen Zabele. Topology discovery service for router-assisted multicast transport. In *Openarch*, 2002.
- [15] S. Zabele, M. Dorsch, Z. Ge, P. Ji, M. Keaton, J. Kurose, J. Shapiro, and D. Towsley. Sands: Specialized active networking for distributed simulation. In *DARPA Active Networks Conference and Exposition (DANCE)*, San Francisco, California, USA, May 2002.