SemCast: Semantic Multicast for Content-Based Data Dissemination

Olga Papaemmanouil and Ugur Cetintemel

Department of Computer Science
Brown University
Providence, Rhode Island 02912

CS-04-10
July 2004

SemCast: Semantic Multicast for Content-based Data Dissemination[1]

Olga Papaemmanouil
Brown University
olga@cs.brown.edu

Uğur Çetintemel
Brown University
ugur@cs.brown.edu

Abstract

We address the problem of content-based dissemination of highly-distributed, fast data streams for real-time, stream-based monitoring applications and large-scale event dissemination. Existing content-based dissemination approaches commonly rely on distributed dissemination trees that require filtering at all brokers on the tree. We present a new semantic multicast *approach that eliminates the need for content-based filtering at interior brokers and facilitates fine-grained control over the construction of dissemination trees. The central idea is to split the data streams (based on their contents, rates, and destinations) and spread the pieces across multiple multicast channels, each of which is implemented as an independent tree. We present the basic design and evaluation of* SemCast*, an overlay network based system that implements this semantic multicast approach. We argue and demonstrate through a detailed simulation study and realistic network topologies that, in many scenarios, SemCast significantly improves the efficiency of dissemination compared to traditional approaches.*

1. Introduction

There is a host of existing and newly-emerging applications that require sophisticated processing of high-volume, real-time streaming data. Examples of such stream-based applications include network monitoring, large-scale environmental monitoring, and real-time financial services and enterprises [1, 5]

Many stream-based applications are inherently distributed and involve data sources and consumers that are highly dispersed. As a result, there is a need for data streams to be routed, based on their contents, from their sources to the destinations where they will be consumed. Such *content-based routing* differs from traditional IP-based routing schemes in that routing is based on the data being transmitted rather than any routing information attached to it. In this model, sources generate data streams according to application-specific stream schemas, with no particular destinations associated with them. The destinations are independent of the producers of the messages and are identified by the consumers' interests.

Consumers express their interests using declarative specifications, called *profiles* [3, 15], which are typically expressed as *query predicates* over the application schemas. The goal of the content-based routing infrastructure is to efficiently identify and route the relevant data to each receiver.

In this paper, we present an overview of *SemCast*, an overlay network-based substrate that performs efficient and Quality-of-Service (QoS)-aware content-based routing of high-volume data streams. SemCast creates a number of *semantic multicast channels* for disseminating streaming data. Each channel is implemented as an independent dissemination tree of brokers (i.e., routers) and is defined by a specific channel content expression. Sources forward the streaming data to one or more channels. Destination brokers listen to one or more channels that collectively cover their clients' profiles.

The key advantages of SemCast are twofold: First, SemCast requires content-based filtering only at the source and destination brokers. As each message enters the network, it is mapped to a specific semantic multicast channel and is forwarded to that channel. As a result, the routing at each interior broker corresponds to simply reading the channel identifier and forwarding the message to the corresponding channel, thereby eliminating the need to perform potentially expensive content-based filtering. This approach differs from traditional content-based routing approaches [4, 6, 10, 15] that commonly rely on distributed *discrimination trees*: starting from the root, at each level of the tree, a predicate-based filtering network filters each incoming message and identifies the outgoing links to which the message should be forwarded. Even though a large body of work has focused on optimizing local filtering using intelligent data structures [3, 7, 11, 13, 18], the forwarding times in the presence of a large of number of profiles are typically in the order of tens of milliseconds or higher (depending on expressiveness of the data and

[1] This work is supported in part by the National Science Foundation under grant IIS-0325838.

profile model, and the number of profiles) [3, 7]. As a result, forwarding costs can easily dominate overall dissemination latencies, as well as excessively consume broker processing resources. The problem becomes more pronounced when data needs to be compressed for transmission, as each broker will then have to incur the cost of decompression and recompression, which are commonly employed when transmitting XML data streams [20].

Second, SemCast semantically splits the incoming data streams and sends each of the sub-streams through a different dissemination channel. This approach makes the system quite flexible and adaptive, as different types of dissemination trees can be dynamically created for each dissemination channel, depending on the QoS expected by the clients of the system. On the other hand, existing approaches commonly rely on predetermined overlay topologies, assuming that an acyclic undirected graph (e.g., a shortest path tree) of the broker network is determined in advance [4, 6]. As we demonstrate, these approaches fail to recognize many opportunities for creating better optimized dissemination trees. Moreover, as shown by the recent work [8, 17], using meshes or multiple trees can significantly increase the efficiency and effectiveness of data dissemination.

In the SemCast approach, different channels are created and the content of each channel is identified through a cost-model that uses (1) stream contents and rates, (2) the network locations the stream elements need to be delivered, and (3) profile characteristics. Each channel is implemented as a multicast tree, and clients subscribe to the channels by joining the corresponding multicast trees. The system gathers statistics in order to adapt to dynamic changes in the profile set and stream rates. SemCast uses such statistical information to periodically reorganize the channels' contents, making cost-based decisions, striving to improve the overall bandwidth efficiency of the system.

In this paper we provide a detailed description of SemCast and its semantic, cost-based channelization model. Using realistic network topologies and various workloads, we compare the semantic multicast approach to the representatives of competing approaches using a detailed simulation study. We demonstrate that SemCast not only reduces the processing load on the brokers of the system, but also surprisingly improves the bandwidth efficiency of the system in many scenarios. We should note that a preliminary overview of SemCast was presented earlier [19].

The rest of the paper is structured as follows. In Section 2, we present the basic design and architecture of SemCast. Section 3 outlines the algorithmic challenges and presents our semantic, cost-based channelization algorithm. In Section 4, we discuss our heuristics for QoS–aware dissemination tree construction. In Section 5, we present experimental results that characterize the bandwidth efficiency of SemCast and representatives of traditional content-based dissemination approaches. Section 6 briefly discusses prior related work and, finally Section 7 concludes with a brief summary of our contributions and directions for future research.

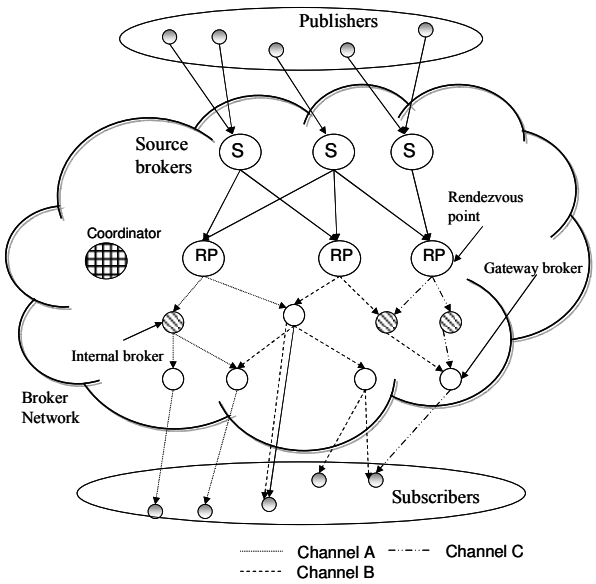

Figure 1: Basic SemCast system model

2. System model

SemCast disseminates XML *streams*[2] consisting of a sequence of XML messages, where each XML message is a single XML document. Subscribers of the system express their profiles using the XPath query language [22]. Each client profile is optionally also associated with a QoS specification that expresses the client's service expectations [5], typically as a constraint on a specific metric of interest. The primary QoS metric address in this paper is *staleness*, which measures the end-to-end delay between the time a message is injected into the dissemination system and the time it is delivered to the interested parties.

The system consists of a set of brokers organized as an application-level overlay network. Publishers are the producers of data and interact with the system through *source brokers* (or simply *sources*). Subscribers are the final destinations of data and are connected to gateway brokers. Brokers that are not at the "edges" of the system (i.e., those with no publishers or subscribers connected to them) are called *interior brokers*.

The *coordinator* node is responsible for maintaining the dissemination channels and determining their contents. For simplicity of exposition, we assume throughout the paper that there is a single coordinator. However, it is possible to distribute coordinator functionality across multiple nodes to improve availability and scalability of the system.

Specific brokers of the overlay serve as *rendezvous points* (*RP*s). Each RP is responsible for at least one

[2] The basic approach is general and not restricted to XML streams.

channel and serves as the root of the corresponding multicast tree. This network model is shown in Figure 1.

The content of each channel is defined from the profiles assigned to it. It could be the more general profile (if such a profile exists) or a disjunction of some of the profiles in the channel. In the following sections, we will describe how contents of the channels are identified. The content expression of a channel is also described using an XPath expression.

Sources receive from the coordinator the current XPath channel content expressions. Any modification to these expressions is pushed to the sources to keep them up-to-date. Sources filter incoming messages according to each channel's content expression and decide to which channels to forward them. They forward these messages to the corresponding rendezvous points, which are responsible for efficiently multicasting them to the subscribed clients through the broker network.

Each broker maintains a simple *routing table* whose entries map channel identifiers to the descendant brokers in that channel. Upon receipt of a message addressed to a specific channel, a simple lookup is performed on the channel's identifier and the message is forwarded to the returned descendant list.

Finally, each gateway broker maintains a local *filtering table* whose entries map the subscribers' profiles to their IP addresses. Received messages are matched against the profiles of the filtering table and are forwarded only to interested clients. This local filtering ensures that end clients do not receive irrelevant data.

3. Content-Based Channelization

SemCast addresses the problem of content-based channelization of data streams. The problem requires determining: (1) how many channels to use, (2) the content of each channel, (3) which channels to use for each incoming message, and (4) which channels each subscriber should listen to. SemCast creates semantic channels on the fly: the number and contents of channels depend on the overlap of the subscribers' profiles as well as stream characteristics (e.g., rates) and thus may change dynamically.

SemCast has two primary operational goals while performing channelization. First, it ensures that there are *no false exclusions*: the subscription of gateway brokers to semantic channels guarantees that every message will be delivered to all the end clients with matching profiles. Second, SemCast strives to *minimize the run-time cost*: the cost metric we use here is the overall bandwidth consumed by the system.

To minimize the run-time cost, we focus on (1) reducing any redundant messages reaching the channels and (2) creating efficient dissemination trees. In order to reduce data redundancy, our channelization algorithm strives to minimize the overlap between the channel contents. Moreover, a decentralized incremental algorithm for constructing low cost multicast trees is used, in order to reduce bandwidth consumption.

We distinguish two phases during SemCast's operation: *initial channelization* and *dynamic channelization*. The former is used when there are no statistics available. As a result, initial channelization is based solely on syntactic similarities among profiles. On the other hand, dynamic channelization is used to reorganize the channels' contents and membership, and is based on run-time statistics. We describe these two phases in the rest of the section.

3.1 Initial Channelization

In SemCast, the initial subscription assignment to channels is based on a syntax-based containment function, which identifies containment relations among the content of the channel and the profile of the subscription request. To achieve this, we rely on existing algorithms for identifying containment relations among XPath queries[3]. We say that an XPath expression P_1 *covers* (or *contains*) expression P_2 if and only if any document matching P_2 also matches P_1. Covering relationships for simple conjunctive queries can be evaluated in $O(nm)$ time, where n and m are the number of nodes of the two tree patterns representing the XPath expressions [23].

Upon receipt of a new profile from a client, a gateway broker checks if the client can be served by the channels to which it has already subscribed. Profiles not covered by these channels are sent to the coordinator. Each new profile, P_{new}, received at the coordinator will be assigned to a channel whose content expression covers P_{new}. If such a channel does not exist, a new channel is created for P_{new}. If there are multiple channels that cover P_{new}, then P_{new} is assigned to the channel with the minimum incoming stream rate. This approach strives to assign profiles to a channel that disseminates the least irrelevant content to the gateway brokers. The coordinator informs the gateway broker of the channel it should subscribe to, and the broker joins the corresponding multicast tree, using one of the tree construction heuristics described in Section 4. We then say that P_{new} *is assigned* to that channel.

SemCast handles also unsubscription requests. If a client wants to remove its subscription, it sends a request to the gateway broker to which it originally connected. This broker identifies the channel from which this subscription is satisfied, and removes the subscription from its local filtering table. Unless there are more clients interested in receiving message from that channel, the gateway broker removes itself from it and the corresponding dissemination tree by sending an unsubscription request to its parent in the multicast tree. When a broker receives such a request from one of its peers, it checks whether it has any more descendents in the tree. If all of its children have been removed from this channel, it also removes itself and forwards the request one overlay hop closer to the source. Once a

[3] In the case of relational data, where profiles are (*attribute, value*) pairs, containment relations are defined in [6].

channel is left with no gateway brokers wishing to receive its content, the coordinator removes the channel from the system.

As profiles are assigned to channels based on containment relations, *containment hierarchies* among profiles are materialized and maintained. In these hierarchies, each profile has as ancestor a more general profile and as descendants more specific profiles. More specifically, each profile P has as parent a profile covering P, and as children profiles covered by P. Among multiple candidate parents, the most specific one (based on syntax-based containment function) is chosen. This implies that if P_i covers P_j and P_j covers P_k, the candidate parents for P_k are P_i and P_j, and P_j is chosen as P_k's ancestor.

Each semantic channel has its own containment hierarchy, including all the profiles assigned to that channel. This profile set corresponds to the gateway brokers that should be connect to this channel. Also the root expression of the hierarchy is the content expression of the channel. Thus, containment hierarchies reflect the current state of the system, as they represent the contents of each channel and the gateway brokers connected to them.

3.2 Dynamic Channelization

The containment algorithms used in the initial channel assignment are based on the syntactic structure of XPath expressions. However, algorithms for discovering such containment relations do not cover the entire domain of the XPath query language, as they work only for a certain subset of the language's constructs. Thus, although a profile may be covered by a channel's content expression, syntactic analysis may not discover this relation, and create a new channel for this profile. This implies that channels may not disseminate disjoint sets of documents, leading to multiple copies of the same document being sent through different channels. The bottom-line is that initial channelization based on syntactic analysis alone is unlikely to produce good results.

To address this issue, SemCast continuously re-evaluates its assignment decisions based on run-time stream rates and profile similarities, to decide whether the channels' contents and membership should be reorganized. This process attempts to minimize the overall bandwidth consumption.

Our first goal is to identify containment relations not discovered by the syntactic analysis of the profiles. SemCast discovers these new coverage relations, and constructs a new set of containment hierarchies to replace the current ones. This new set of hierarchies reflects how the channels' content and brokers subscriptions to these channels should be, based on the latest statistics. If the current state of SemCast "sufficiently" differs from the state shown in the new hierarchies, channel reorganization takes place.

Moreover SemCast maintains statistics about the partial overlap among profiles and checks if partially overlapping hierarchies could be merged and placed in the same channel in order to further reduce the run-time cost. In the following, we explain the dynamic channelization phase in more detail.

3.2.1 Statistical information

As mentioned above, SemCast tries to adapt to changes in the rate of the incoming messages and the overlap of the set of matching documents for each profile. To implement this, we maintain information about the selectivity of the profiles in a *selectivity matrix.* Moreover, we use a profile overlap matrix to calculate the partial overlap between profiles.

Selectivity matrix. This structure is implemented as a sliding window of the incoming messages and maintained by the sources. The size of this window should be large enough to collect statistics with high confidence. Incoming messages are assigned weights. New messages have higher importance, and as different incoming messages are added in the sliding window, the importance of the old ones decades and their weight decreases. By holding a weighted sliding window, we base our statistics on the most recent messages, allowing a graceful aging of messages.

Each row in this matrix refers to a different incoming message and each column to a different profile. Whenever the source creates a message matching a channel, the channel's current syntax-based containment hierarchy is parsed in a top-down manner. At each step, we check whether the message matches the current profile. If it does, we continue with its children, otherwise we stop. If a match is found for a profile, the corresponding entry in the selectivity matrix is set to one, otherwise it remains zero.

The selectivity matrix provides us with information about the set of documents each profile has matched, in the time frame of the sliding window. Based on this information, we can estimate both the rate of the incoming XML messages and the overlap among our profiles.

We note that although an array of size p x w has to be stored for the selectivity matrix (where p is the number of profiles and w the size of the sliding window), only the profiles of the channels disseminating the incoming messages are updated each time. However, only a part of these profiles will be updated for each new message, since the match check will not always reach the leaves of the syntax-based hierarchy.

Profile overlap matrix. We now define a *partial overlap metric* between profiles P_i and P_j: we say that P_i k-overlaps with P_j, denoted $P_i \subseteq^k P_j$, if the ratio of the number of messages matching P_j *and* P_i to that matching P_i is k. Obviously if k=1, then P_j contains (or covers) P_i, and the set of messages matching P_i is a subset of those matching P_j.

Table 1. Overlap matrix

	P_1	P_2	P_3	P_4	P_5	P_6	P_7
P_1	1	0	0	0	1	1	0
P_2	0	1	0.4	0.5	0	0	0
P_3	0	1	1	0.6	0	0	0
P_4	0	1	1	1	0	1	1
P_5	0.8	0	0	0	1	0	0
P_6	0.2	0	0	0.4	0	1	1
P_7	0	0	0	0.2	0	0.5	1

SemCast calculates the pairwise partial overlap metric of the profile set in order to identify containment and partial overlap relations between profiles. An *overlap matrix* is computed from the selectivity matrix. Each entry O_{ij} represents the *k*-overlap metric for the profiles P_i and P_j. In the next section, we describe how this matrix is used in the dynamic channelization phase.

3.2.2 Construction of Containment Hierarchies

As described in the previous sections, the dynamic channelization phase tries to reduce the overlap among the contents of the channels that was probably introduced since the last reorganization phase. Thus, containment relations among profiles are re-evaluated based on the statistics gathered and a new set of hierarchies are identified. In this phase, our approach tries to place profiles matching similar messages under the same channel, *if* this could lead to further improvement of bandwidth. To achieve that, SemCast uses a *cost model* to identify if two profiles, where one is covered by the other, should be in the same channel. More specifically, hierarchy construction tries to reorganize SemCast and move the system to a state with lower run-time cost.

Containment relations. The overlap matrix can be used for identifying containment relations between two profiles. The following rules apply for the overlap matrix: (1) the root profiles of the hierarchies P_j have a single '1' in the j-th column at the entry O_{jj}; (2) candidate parents P_j of each non-root profile P_i have O_{ji} =1, $i \neq j$.

In Figure 2, we show a simple example of the containment hierarchies that exists among seven profiles. Table 1 shows how the overlap matrix could like for these profiles. The dashed line shows that P_1 is covered also from P_6, but P_6 is not its final parent in the containment hierarchy as the bandwidth consumption from placing it under P_4 is higher (this is because they share a higher percentage of documents).

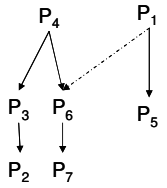

Figure 2. Containment hierarchy from Table 1

To create the new set of hierarchies, SemCast first identifies the root profiles using the overlap matrix. We note that the number of root profiles represents the maximum number of channels the system can have after the reorganization. However, the number of channels will be lower, because we merge some of the hierarchies and place them in the same channel. This merging operation is explained in the next section.

Once the root profiles are discovered, SemCast identifies for every other profile *P*, the set of profiles matching a more general set of messages. These are the *candidate parents*. For each of these candidate parents, we identify the one that achieves the lowest bandwidth consumption.

Thus, in the new set of containment hierarchies, each profile P_j is an ancestor of P_i if P_j covers P_i, i.e. $P_i \subseteq^I P_j$, and if

$$\cos t_total(S, M) > \cos t_total(S', M) \qquad (1),$$

where *cost_total*(*S*, *M*) is the cost of the system with a set *S* of semantic channels and a set *M* of incoming messages. In the above inequality, in S , P_i and P_j are in different channels and in S' they are placed in the same channel. Among multiple candidate parents for P_i, we choose the one with the maximum benefit in terms of bandwidth. This is usually the parent that has the higher percentage of overlap with its child and the lowest rate for the non-overlapping part with this child. Thus, SemCast places two profiles under the same channel if this will improve the bandwidth consumption. The result is a new set of hierarchies that improve the overall cost, compared with the cost of the system for the current hierarchies.

Cost model. The cost of SemCast, in terms of bandwidth consumption, is determined by four factors:

- The channels *S*, in the system,
- The messages each channel disseminates, M_i, $i \in S$,
- The rate of each of these XML documents, $r_m, m \in M_i$,
- The edges in the channels, E_i, $i \in S$.

We define the cost of SemCast as the sum of the cost of each channel:

$$\cos t_total(S, M) = \sum_{i \in S} \text{cost of channel } i \,,$$
$$= \sum_{i \in S} \sum_{e \in E_i} \sum_{m \in M_i} r_m$$

and we use this cost formula in our cost model.

To estimate $\cos t_total(S', M)$ in equation (1), we need to calculate the cost of a "virtual" channel that would include both the profile *P* and its candidate parent. This cost will be compared with the cost of the channels these profiles are currently assigned. We note here that this procedure applies to profiles that are not already placed under the same channel in the current

state of the system. If the candidate parent and the profile are assigned to the same channel, then the bandwidth consumption increase of will be zero. If another candidate parent can provide a higher cost improvement then the profile P changes its parent in the new set of hierarchies.

Thus, for a profile P_i and its candidate parent P_j SemCast checks if

$$cost_channel(P_i) + cost_channel(P_j) > cost_channel(P_j, P_i)$$

where $cost_channel(P_i)$ is the cost of the channel P_i is currently assigned, and $cost_channel(P_j, P_i)$ is cost of the virtual channel containing both P_i and P_j.

If P_i is currently assigned to channel i, $i \in S$ and P_j is the root of the current containment hierarchy in this channel, then:

$$cost_channel(P_i) = \begin{cases} cost_channel(P_j), & i \neq j \\ \sum_{e \in E_i} \sum_{m \in M_i} r_m, & \text{otherwise} \end{cases}$$

As mentioned, this cost refers to an already existing channel i. This implies that the number of edges E_i can be calculated, since this channel is already represented by a multicast tree. Also the number of messages sent thought the channel is the set of messages matching the root profile of the channel. Thus these messages and their rates can be calculated from the selectivity matrix.

In the case of the cost for the 'virtual' channel, $cost_channel$ (P_j, P_i), we need also to identify the possible set of incoming messages, their rates and the number of edges in the channel. The number of documents and their rates could again be derived from the selectivity matrix. These are the documents matching the more general profile P_j. However, calculating the cost of a potential channel, which includes both P_i and P_j, requires some approximation of the number of edges in the channel, since this channel is not already represented by some multicast trees.

Cost approximation. To approximate the number of edges in this channel, the coordinator assumes that one of the existing multicast trees of the two profiles, e.g. P_i, is the base tree and tries to connect all the gateway brokers interested in P_j in this base tree. To achieve that, for each gateway broker (except the ones already in the multicast tree), the coordinator calculates the number of hops it takes to connect to the multicast tree and sums this number over all the gateway brokers. More formally, an approximation of the number of edges in the channel with P_i and P_j, if P_i is currently assigned to channel i, $i \in S$, is given by

$$|E_i| + \sum_{GB \in D_j, GB \notin D_i} Path(GB, T_i),$$

where $Path(GB, T_i)$ represents the length of the path from the gateway broker GB to the closest node in the multicast tree T_i corresponding to channel i, and D_i is the set of gateway brokers having a client with profile P_i. Having an estimation of the number of edges in the virtual channel, the coordinator can decide if P_j and P_i should be placed together in the same channel.

Based on the cost model, the coordinator can decide on the new set of containment hierarchies. This new set is compared with the current set of hierarchies in the system, to identify any changes in the channel membership and content. If there are new roots in the hierarchies, then a new channel will be created for each new root. If one profile ceases to be part of a root, the profile's descendants in the current hierarchy have to migrate to the channel corresponding to their new root. Finally, if one profile is assigned to a hierarchy with a new root profile, then the gateway brokers interested in this profile have to subscribe to the channel of the new root and unsubscribe from the previous one. Brokers are informed by the coordinator about the new channels they should subscribe or unsubscribe and sources receive the new channel list and their updated content expressions.

3.2.3 Hierarchy Merging

Although periodic reevaluation of the containment hierarchies and channel reorganization can improve the efficiency of the system, a highly diverse profile set could result in a large number of channels. This scenario may arise when only a small percentage of the profiles are covered by another. Since containment hierarchies are based on coverage relations between profiles, they fail to identify profiles with high partial overlap and thus the reorganization procedure would place them on different channels, increasing the bandwidth consumption.

At the extreme case, i.e., when no containment relations are discovered, the number of hierarchies could be equal to the number of distinct profiles in the system. This implies that the number of constructed channels will also be equal to the number of distinct profiles in the SemCast. In this case, the system will not be able to exploit the actual partial overlap among the subscribers' interests and will create independent paths for potentially each profile regardless of common messages. Moreover, the number of channels and space requirements for routing tables will increase.

To address this problem, SemCast places the containment hierarchies that partially overlap in the same channel. However, such *merging of hierarchies* would likely increase redundant message transmissions, as gateway brokers attached to the final channel might receive extra messages matching the non-overlapping part between the merged hierarchies. For this reason, SemCast merges only those hierarchies that will decrease the overall cost of system.

Cost model. To implement this approach, SemCast uses the overlap matrix to evaluate a cost function (similar to the one described earlier for hierarchy construction). We make a pair-wise comparison of the root profiles and check their partial overlap, given by O_{ij}. If there is partial overlap between P_j and P_i, i.e. $P_i \subseteq^k P_j$ or $P_j \subseteq^k P_i$ have $k>0$, then we check if:

$$cost_channel(H_i) + cost_channel(H_j) > cost_channel(H_i, H_j)$$

where $cost_channel(H_i)$ is the cost of a channel representing the new containment hierarchy H_i and $cost_channel(H_i, H_j)$ is the cost of a channel where both hierarchies H_i *and* H_j are assigned. As in the case of the hierarchy construction, we give priority to merging operations that have the higher cost benefit. The merging procedure terminates when no further cost reduction can be achieved.

Cost approximation. To calculate the cost of a channel, we again need to approximate the number of edges of the corresponding 'virtual' channel. We calculate the number of gateway brokers interested in the set of profiles in H_i. We assume that they connect to the 'virtual' channel in random order. The first broker is assumed to be connected the source through a shortest path. The consequent brokers connect to the closest one of the brokers already in the virtual channel. The sum of the edges for all these paths is the approximated number of edges for the potential channel. The coordinator calculates the shortest paths, using location information collected from the brokers during run-time. The same procedure is followed for the merged channel, where the set of gateway brokers are those interested in the profiles of both hierarchies.

Similar to the case of hierarchy construction, to calculate the cost of each virtual channel, we use the selectivity matrix to estimate the set of documents matching each hierarchy and their rates. For each hierarchy the set of matching documents is the set of messages that match the root profile. For the case of the merged hierarchy, we use the union of the set of documents matching each hierarchy.

We note that the dynamic channelization implicitly involves hierarchy splitting. Assume that by the time of the next reorganization phase, the selectivity of P_2 in Figure 2 has changed (or even the rate of its matching messages) and it is no longer covered by any profile. In this case the first hierarchy will split into two, one with P_4 as the root, and one with P_2, creating three final hierarchy trees (and possibly three channels).

4. Dissemination tree construction

SemCast relies on a distributed and incremental approach to create two different types of dissemination trees, low-cost or delay-bounded trees, which we describe in the rest of this section.

Low-cost trees. SemCast attempts to organize nodes with same interests in the same dissemination tree. Moreover, the tree construction heuristics strive to create multicast trees that include a small number of internal brokers, since these brokers are not interested in the disseminated information. Thus the main idea of the low cost heuristic is to connect one gateway broker to the closest one in the same semantic channel. In this way, messages will flow among gateway brokers interested in their content, and they will follow a small number of hops between two interested parties.

To implement this, a gateway broker, receives from the coordinator list of the current destinations in the channel to which it wishes to connect. The broker then finds the min-cost path to each destination and connects to channel through the closest one. Join requests are sent all the way from this gateway broker to the first node in the tree and the routing tables of the brokers on this route are updated to reflect the new tree structure. This approach proceeds in a way similar to how Steiner trees are constructed (e.g., [21]), although it is distributed and works in an incremental manner.

Delay-bounded trees. SemCast can create delay-bounded dissemination trees if clients indicate latency expectations. Since a gateway broker serves multiple clients, it has to determine the proper target latency value based on the expectations of all its clients. We here assume that the gateway broker has decided a latency-bound for each of the channels it listens to according to some policy.

The gateway broker first finds a connection node in the multicast tree using the low-cost heuristic described above. All brokers continuously track their "distance" from the root of the channels they maintain. Thus, upon receipt of a delay-bounded join requests, they can estimate if they can satisfy the latency constraints. If the connection node realizes that it will not be able to meet the latency constraint tagged to the join request, it will forward the join request to its parent in the channel. If the parent, which is not necessarily a gateway broker for the channel, can satisfy the latency bound, then the gateway broker connects to the channel through the shortest path to the parent. This process will continue until either a broker that is sufficiently close to the root is found or it is decided that the constraint cannot be met. In the latter case, the client will be notified and, optionally, the gateway broker will be connected through the shortest path to the root.

5. Experimental Evaluation

We now describe our performance evaluation that characterizes the efficiency of the SemCast approach, comparing it against the representatives of more traditional content-based filtering approaches using detail simulations over realistic network topologies.

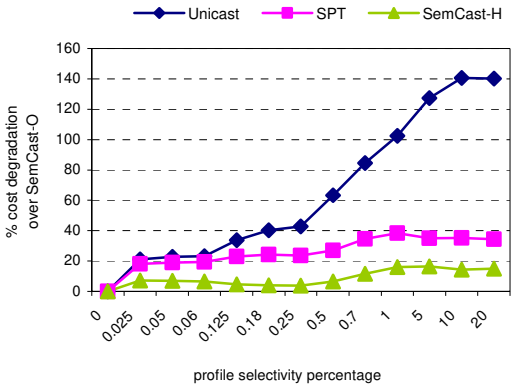

Figure 3. Relative bandwidth efficiency for varying profile similarity factors

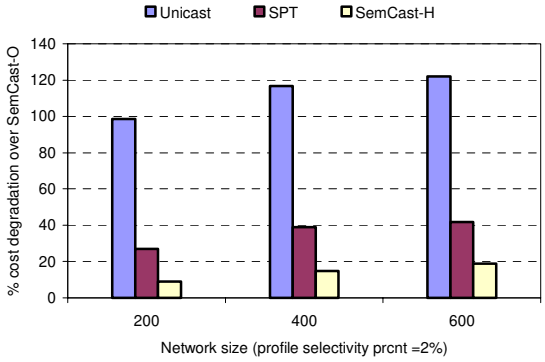

Figure 4. Relative bandwidth efficiency for different network sizes

Simulation setup. In the simulation, we use an overlay network of brokers, based on the transit-stub model generated with the GT-ITM tool [24]. In the experiments, we used networks of up to 700 broker nodes and 7000 clients. In order to control the impact of profile similarities, we use a parameter called the *profile selectivity percentage*, which determines the number of distinct profiles. A selectivity factor of x% gives us a set of $1/x$% distinct profiles. Thus, the higher the selectivity factor is, the smaller the set of distinct profiles is. Here, the term "distinct" refers to profiles that do not match the same set of messages. However, partial overlap or containment relations may still exist among these profiles. Other simulation parameters of importance are described in the text wherever appropriate.

Clients pick their subscriptions randomly from the set of distinct profiles and connect to a single broker, which serves as their gateway broker. Publishers also connect to one broker and forward their messages to them. For simplicity, every publisher connects to the same source broker, and rendezvous points are placed on the same node. Since each channel is constructed independently, this setup does not affect our experimental results.

Bandwidth efficiency is a key goal for all large-scale networked systems. We thus use the overall bandwidth consumed by the broker network as our primary evaluation metric. We also present results on the size of the filtering networks at each broker, in order to characterize the processing required for message filtering at brokers.

Studied dissemination approaches. We simulated several approaches for content-based data dissemination. First, we simulated a Unicast (UC) approach, where messages are filtered at the source brokers and are forwarded to the interested parties using shortest paths. This approach models the delivery scheme used by a centralized filtering system (such as XFilter [3]) that uses unicast-based dissemination of filtered messages. Second, we simulated an approach that models a traditional, distributed publish-subscribe system (e.g., [10, 15]). In this model, profiles are propagated and aggregated up to the source over shortest paths from the clients to the source. If a broker receives a subscription and its local filtering table already contains a more general filter, the subscription is not forwarded further up towards the source. In other words, we create a predicate-based overlay filtering tree on top of a shortest path spanning tree of the broker network. Filter-based routing tables are placed at each broker and messages are matched at each hop against the filters in the filter table to determine the next hop(s) towards the clients. We refer to this algorithm as SPT (Shortest Path Tree).

Finally, we simulated two versions of the SemCast approach: SemCast-H simulates the distributed tree construction heuristic described in the previous section. SemCast-O employs the same algorithm for determining the number and contents of the dissemination channels, but applies a centralized Steiner tree construction algorithm [21] that uses global knowledge of all the profiles and the destinations for each channel.

5.1 Basic Performance

We first evaluate the performance of SemCast's tree construction heuristics when no partial overlap exists among profiles. In this case, subscriptions match disjoint sets of messages and no partial overlap exists among the channels' contents.

Figure 3 shows the cost degradation (i.e., extra bandwidth consumption) over SemCast-O for all approaches (for 700 nodes and 7000 profiles). SemCast-H performs the closest to the optimal, incurring up to only 6.4% extra cost on the average. SPT performs better than UC; it does, however, increase the cost by 23% on the average, whereas UC's average cost increase is 41%. Note the non-uniform scale on the x-axis and the curves pretty much flatten out beyond a selectivity percentage of 10%.

The poor efficiency of UC is not surprising since UC requires every message to follow an independent shortest path to each of the destinations. Thus, UC disseminates the same message multiple times through the same links.

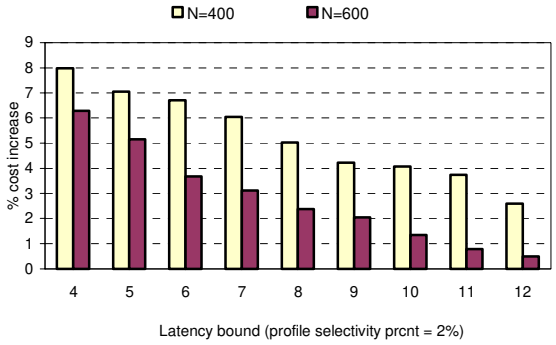

Figure 5: Bandwidth consumption increase due to latency bounds

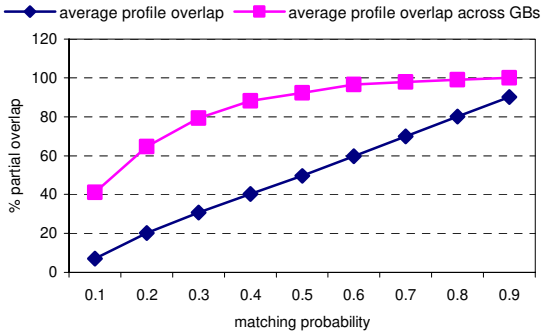

Figure 6: Profile overlap for varying matching probability values

In SPT, some of this redundancy is reduced, due to the filters at each broker. However, notice that every broker is connected to the source through a single, predefined path. Unless all the interior brokers on this path are also interested in receiving the exact same messages, there will be redundancy in message transmissions. This redundancy is quantified in the figure.

SemCast-H strives to minimize this redundancy by placing nodes interested in the same messages on the same multicast tree. Moreover, the construction algorithm constructs trees so as to minimize the number of internal brokers that are not interested in the same content.

Figure 4 shows the results of the previous experiment for different network sizes. We observe that the cost increases with increasing network size for all approaches. Larger networks typically imply longer paths for reaching the final destinations. Thus, message redundancy increases as we increase network size, more so for UC and SPT than for SemCast-H. SemCast-H is able to keep the bandwidth consumption lower than 18% of SemCast-O, for all network sizes.

5.2 Latency-aware dissemination

In this experiment, we study the performance of SemCast's heuristic for delay-bounded dissemination. We assumed for simplicity that all subscribers express the same latency bounds and that latency is measured as the number of broker hops each message has to take from the source to the destination gateway broker. Here, we observe a fundamental tradeoff between dissemination costs and latencies: the stricter the delay expectations are, the higher the cost of dissemination is. Our results demonstrate this tension.

We run our low-cost heuristic and delay-bounded heuristic using the same set of profiles. However, in the latter case, we vary the latency expectations of our subscribers. Figure 5 shows the percentage of cost increase for the delay-bounded case compared with the low-cost one, for two different network sizes. As we relax the delay bounds, the cost of the delay-bounded approach decreases and approximates the cost of the low-cost heuristic. The cost increase due to the latency constraints typically does not go beyond 10%, even with low profile selectivity factors. The implication is that even when latency bounds are imposed by the users, SemCast's heuristics achieve lower bandwidth consumption than SPT, since the difference of SPT and SemCast-H in most of the cases is above 10%.

We also measured the depth of the multicast trees created by the SemCast-H and compared them to those created by SPT. We observed that, for an overlay network of 600 brokers and for varying profile overlap values, SemCast-H created trees that are on the average 2-2.5 hops deeper than those created by SPT. Note that the depth of the tree does not affect the throughput of the system, only the end-to-end latency. If latency is crucial, SemCast can use the latency bounding techniques described in Section 4 to effectively create shortest path trees at the expense of increased bandwidth consumption, while still retaining lower processing requirements.

5.3 Partial overlap effect

We now explore scenarios that involve partial overlap of the profiles in the profile set. Recall that SemCast attempts to reduce data redundancy by merging channels. In these experiments, we use a parameter, referred to as *matching probability*, to control the partial overlap among the profiles in our profile set. We define the *average profile overlap* as the average partial overlap over all profile pairs (the partial overlap between two profiles is defined in Section 3.2.1).

We now study SemCast's bandwidth efficiency as a function of the partial profile overlap across gateway brokers. For each gateway broker, we computed the cumulative set of messages matching the local profiles and we calculated the average partial overlap between these sets over all brokers' pairs. This represents the average overlap between any two gateway brokers' interest. Figure 6 shows how the average profile overlap

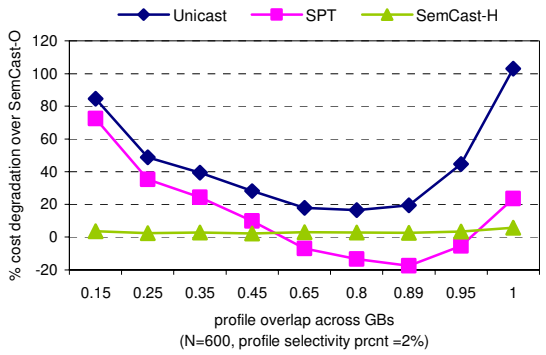

Figure 7. Relative bandwidth efficiency for varying profile overlap values

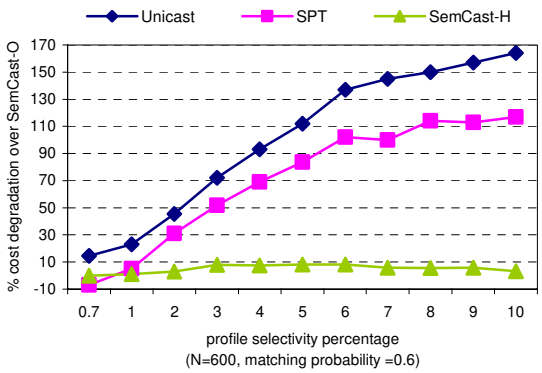

Figure 8. Relative bandwidth efficiency for varying profile similarity factors (partial overlap case)

changes as we increase the matching probability from the perspective of clients and gateway brokers.

Figure 7 shows the cost degradation of Unicast, SPT, and SemCast-H over the SemCast-O approach for increasing profile overlap across gateway brokers. The results exhibit an interesting behavior of SemCast's heuristic. SemCast-H outperforms SPT in two cases: when the profile overlap across gateway brokers is low (from less than 10% to 36% - 40%, depending on the network size), and when the overlap is very high (above 97%). Between those two extremes SPT has lower bandwidth consumption than our approach. Unicast always performs worse than SemCast, and its cost degradation compared with SemCast-O could go up to 80% for low profile overlap. In the following paragraphs we explain the reasons why.

We first justify the better performance of SemCast-H, when the matching probability is small and thus there is low overlap. Thus, the majority of the profiles match a different set of XML messages and only a small fraction of the clients' subscriptions have similarities. SemCast-H identifies the set of gateway brokers interested in the same content, while our tree construction heuristic connects them in a low-cost multicast tree. However, in the case of SPT, gateway brokers are connected to the source broker through predefined paths. The construction of the network's spanning tree does not take into account common interests among clients. Thus, SPT uses paths from the source to the gateway brokers that include brokers not interested in the same information.

We now turn to explaining the reason why SemCast's efficiency temporarily decreases and makes a dip, before it starts increasing again. Recall that the contents of the created channels are not completely disjoint, since there is some similarity in the profile set. Therefore it is clear that some messages are disseminated through multiple channels, and if these channels have common gateway brokers, there is data redundancy; the same information will be sent more than once to the same final brokers. As the profile overlap increases, this redundancy also increases, causing SemCast-H to consume more bandwidth. However it is not until the point where the average profile overlap is 45% that SemCast-H starts to perform worse than SPT.

Between a profile overlap of about 45% and 90% across gateway brokers (at which point the average profile overlap is around 60%), SemCast's efficiency is worse than that of SPT. In this range, the content redundancy among channels is very high, but the partial overlap among channels does not allow merging operations to reduce the bandwidth consumption sufficiently to outperform SPT. Once the profile overlap increases enough, it becomes possible to profitably merge and place in the same channel more rate-based hierarchies, allowing SemCast-H to incur less bandwidth than SPT.

Figure 8 shows the percentage of the cost increase of the three approaches over SemCast-O, for increasing selectivity values. The figure reveals that higher selectivity values result in higher cost improvement for SemCast when compared with UC and SPT. As we increase the selectivity values, the difference between the costs of UC and SPT and that of SemCast-H gets significantly higher. This is because of the fact that high selectivity values result in fewer distinct profiles in the profile set, and thus the content redundancy problem we described above occurs to a lesser extent.

5.4 Processing cost benefits

As mentioned earlier, one of the main advantages of SemCast is that it eliminates the need for local filtering at intermediate brokers. In SemCast, messages are filtered only at two network locations: (1) at the source broker, to identify the channels each message should go to and; (2) at the gateway brokers, to send only the matching documents to each client.

Since we are not studying matching algorithms, for experimentation purposes, we choose to use the number of profiles that need to be maintained and matched against incoming messages as an indirect but representative measure of the processing requirements of different approaches.

Table 2 shows the number of profiles that need to be maintained by SemCast, at the source and gateway brokers, as a percentage of the profiles that SPT root needs to maintain for varying broker-profile overlap values. The table also shows the number of profiles placed at the non-root brokers of the tree constructed by SPT.

Table 2. Profiles on local filtering tables

Profile overlap	**0.02**	**0.03**	**0.05**	**0.08**
SemCast (Source)	69.84%	61.42%	51.6%	50.7%
SemCast (GB)	0.3484%	0.32%	0.42%	0.52%
SPT (Hop 1)	7.29%	9.07%	10.37%	11.26%
SPT (Hop 2)	3.83%	0.52%	6.40%	7.03%

In SPT, the number of local filters at the source is quite high. The reason is that in our profile set, there are not many containment relations among the profiles, so most of the subscriptions are propagated upstream until they reach the source. However, as we move closer to the subscribers, the average number of profiles among all broker in the second level decreases significantly.

In SemCast, the source broker always contains fewer profiles than the source (i.e., root) of SPT. Moreover the amount of filtering performed at the gateway broker is very small. These results indicate that the processing benefits at SemCast could be significant. This allows SemCast to scale up to high data rates, or equivalently support lower data rates with much less processing horsepower. We leave as future work the demonstration of the processing cost benefits for the interior brokers using a prototype implementation running over real XML messages.

6. Related Work

XML filtering. A large body of work has focused on optimizing local filtering using intelligent data structures. For instance, XFilter [3] uses indexing mechanisms and converts queries to finite state machine representations to create matching algorithms for fast location and evaluation of profiles. YFilter [14] also provides matching of XML documents to a large number of profiles, and transformation of the matching XML data, based on recipient-specific requirements. In [11], a space-efficient index structure (that decomposes tree patterns into collections of substrings and index them using a trie) is proposed to improve filtering efficiency.

However, in all these approaches, all profiles are propagated to a single broker. Generated XML streams are forwarded to this central broker for filtering and are disseminated to the interested destinations with unicast connections. The centralized nature of these solutions, and their high bandwidth consumption, demonstrated by our experimental results, limit their scalability.

Content-based routing. Unlike SemCast, existing approaches for content-based networking do not the address the issue of constructing the dissemination overlays. Gryphon [4] assumes that the best paths connecting the brokers are known a priori. Likewise, Siena [6] assumes that an acyclic spanning tree of the brokers is given. Moreover these systems have low expressiveness as they support only pairs of (attribute, values) as subscriptions.

Both XRoute [10] and ONYX [15] provide solutions for content-based publish-subscribe systems with high expressiveness, as subscriptions are expressed using XPath queries. However these systems also have the disadvantage of high processing cost at intermediate brokers. Moreover, our results showed, that by creating multiple multicast channels, instead of sending all the information through a single underlying overlay spanning tree, could decrease the overall bandwidth consumption, even when the profile overlap is low.

Content-based routing has been also studied in [16]. This work refers to routing path queries, which are path expressions in an XPath-like query language, among nodes in P2P system storing XML documents. Here, the peers with similar content are clustered, and nodes are organized in a hierarchical structure, enabling fast content-driven searches.

Finally, a main difference of SemCast from previous approaches on content-based data dissemination is the fact that SemCast adapts to dynamic changes in profile overlap and stream rate. Moreover SemCast estimates and maintains the partial overlap among profiles. To the best of our knowledge, previous solutions relay on full coverage and containment relations among profiles.

Application-level multicast. Many recent research proposals (e.g., [9, 25]) employ *application-level multicast* for delivering data to multiple receivers. Members of the multicast group self-organize into an overlay topology over which multicast trees are created. In all these cases, members of a group have exactly the same interests. Constructing a different multicast group for each distinct profile is not a scalable solution. SemCast estimate partial overlap among the profile set, and could place in the same multicast channel clients with partial similarity between their subscriptions.

Similar to SemCast, SplitStream [8] aims to achieve high-bandwidth data dissemination by striping content across various multicast trees. However, SplitStream does not address profile-based dissemination and channelization. Bullet [17] uses an overlay construction algorithm for creating a mesh over any distribution tree, improving the bandwidth throughput of the participants. As a result, SemCast can utilize the techniques introduced in this work to improve its bandwidth efficiency. XML-based filtering using overlays was investigated in [20]. This work uses multiple interior brokers in the overlay to redundantly transmit the same data from source to destination, reducing loss rates and improving delivery latency.

Channelization. The channelization problem was studied in [2] in the context of simple traffic flows and a fixed number of channels. The problem was shown to be NP-complete. In our model, the fact that the number of multicast groups is not fixed, and is dependent on the overlap of the receivers' interests, makes the problem more challenging. Finally topic-based semantic multicast was introduced in [12]. In this work, users express their interests by specifying a specific topic area. Similarity ontologies are used to discover more general topics served by existing channels. This work does not consider content-based routing, and the focus is not on minimizing bandwidth consumption.

7. Conclusions & Future Work

We described a novel semantic multicast system, called *SemCast*, which takes a radically different approach for content-based dissemination of data streams. The key idea is to split the data streams based on their contents, rates, and destinations, spreading the pieces across multiple dissemination trees. Using a practical cost model, SemCast continuously reevaluates its channelization decisions based on run-time statistics and profile information.

The primary advantages of the SemCast approach over traditional approaches are that (1) it does not require multi-hop content-based filtering of streams at interior brokers, and (2) it facilitates fine-grained control over the construction of dissemination trees. We argued that, in addition to reducing the processing load on the system, SemCast exhibits better bandwidth efficiency in many cases (but not always) as it creates more efficient dissemination trees.

There are two immediate directions in which we are extending our work. First, we are designing batched dissemination techniques to improve system throughput. Second, we are incorporating aggressive caching functionality into SemCast. We are also currently in the process of building a SemCast prototype to verify the practicality and efficiency of the approach, as well as to demonstrate the claimed performance benefits through deployment on PlanetLab.

8. References

[1] D. Abadi, D. Carney, U. Cetintemel, M. Cherniack, C. Convey, C. Erwin, E. Galvez, M. Hatoun, J. Hwang, A. Maskey, A. Rasin, A. Singer, M. Stonebraker, N. Tatbul, Y. Zing, R.Yan, and S. Zdonik. Aurora: A Data Stream Management System (Demo). In *SIGMOD*, 2003.

[2] M. Adler, Z. Ge, J. F. Kurose, D. Towsley, and S. Zabele. Channelization problem in large scale data dissemination. In *ICNP'01*, November 2001.

[3] M. Altinel and M. J. Franklin. Efficient Filtering of XML Documents for Selective Dissemination of Information. In *VLDB*, 2000.

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

[5] D. Carney, U. Cetintemel, M. Cherniack, C. Convey, S. Lee, G. Seidman, M. Stonebraker, N. Tatbul, and S. Zdonik. Monitoring Streams: A New Class of Data Management Applications. In *VLDB*, 2002.

[6] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf. Design and Evaluation of a Wide-Area Event Notification Service. *ACM Transactions on Computer Systems*, 19(3):332-383, 2001.

[7] A. Carzaniga and A. L. Wolf. Forwarding in a Content-Based Network. In *SIGCOMM*, 2003.

[8] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. SplitStream: High-bandwidth content distribution in cooperative environments. In *SOSP*, 2003.

[9] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron. SCRIBE: A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in communications (JSAC)*, 20(8), October 2002.

[10] C.-Y. Chan and P. Felber. A Scalable Protocol for Content-Based Routing in Overlay Networks. In *NCA*, 2003.

[11] C.-Y. Chan, P. Felber, M. N. Garofalakis, and R. Rastogi. Efficient Filtering of XML Documents with XPath Expressions. *VLDB Journal, Special Issue on XML*, 11(4):354-379, 2002.

[12] S. Dao, E. Shek, A. Vellaikal, R. R. Muntz, L. Zhang, M. Potkonjak, and O. Wolfson. Semantic multicast: intelligently sharing collaborative sessions. *ACM Computing Surveys (CSUR)*, 31(2es), 1999.

[13] Y. Diao, M. Franklin, P. Fischer, and R. To. YFilter: Efficient and Scalable Filtering of XML Documents (Demo). In *ICDE*, 2002.

[14] Y. Diao and M. J. Franklin. Query Processing for High-Volume XML Message Brokering. In *VLDB*, 2003.

[15] Y. Diao, S. Rizvi, and M. J. Franklin. Towards an Internet-Scale XML Dissemination Service. In *VLDB*, 2004.

[16] G. Koloniari and E. Pitoura. Content-based Routing of Path Queries in Peer-to-Peer Systems. In *EDBT*, 2004.

[17] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat. Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh. In *SOSP*, 2003.

[18] L. Opyrchal, M. Astley, J. Auerbach, G. Banavar, R. Strom, and D. Sturman. Exploiting IP Multicast in Content-Based Publish-Subscribe Systems. In *Middleware*, 2000.

[19] O. Papaemmanouil and U. Cetintemel. Semantic Multicast for Content-Based Stream Dissemination. In *WebDB*, 2004.

[20] A. C. Snoeren, K. Conley, and D. K. Gifford. Mesh-Based Content Routing using XML. In *SOSP*, 2001.

[21] H. Takahashi and A. Matsuyama. An Approximate Solution for the Steiner Tree Problem in Graphs. *Mathematica Japonica*, 1980.

[22] W3C. XML Path Language (XPath) 1.0. 1999.

[23] P. T. Wood. Containment for XPath fragments under DTD constraints. In *ICDT*, 2003.

[24] E. Zegura, K. Calvert, and S. Bhattacharjee. How to Model an Internetwork. In *INFOCOM96*, 1996.

[25] S. Q. Zhuang, B. Y. Zhao, A. D. Joseph, R. H. Katz, and J. D. Kubiatowicz. Bayeux: An Architecture for Scalable and Fault-tolerant Wide-area Data Dissemination. In *NOSSDAV*, 2001.