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

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, high-volume data streams for stream-based monitoring applications and large-scale data delivery. Existing content-based dissemination approaches commonly rely on distributed filtering 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 efficient dissemination trees. The central idea is to split the incoming data streams (based on their contents, rates, and destinations) and then spread the pieces across multiple channels, each of which is implemented as an independent dissemination tree. We present the basic design and evaluation of* SemCast*, an overlay-network based system that implements this semantic multicast approach. Through a detailed simulation study and realistic network topologies, we demonstrate that SemCast significantly improves the efficiency of dissemination compared to traditional approaches.*

1. Introduction

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

Many stream-based applications are inherently distributed and involve data sources and consumers that are geographically dispersed. As a result, there is a need for data streams to be routed, usually 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 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 schemas, with no particular destinations associated with them. Thus, the destinations are independent of the producers of the messages and are instead identified by the consumers' interests, which are commonly expressed through declarative specifications called *profiles* [3, 15]. Profiles are usually specified as *query predicates* over application schemas. The goal of a content-based routing system is to efficiently identify and route the relevant data to each consumer.

In this paper, we present *SemCast*, an overlay-network system that performs distributed content-based routing of highly-distributed, 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., application-level routers) and is defined by a specific content expression (i.e., predicate). Sources forward each incoming message to one or more channels. Destinations listen to one or more channels that collectively cover their profiles.

The key advantages of SemCast are twofold. First, SemCast requires content-based filtering only at the source and destination brokers. As each incoming message enters the network, it is mapped and forwarded to a specific channel(s). As a result, the routing at each interior broker only involves reading a channel identifier and forwarding the message to the corresponding channel. This approach eliminates the need to perform expensive content-based filtering at interior brokers. In this respect, SemCast radically differs from the traditional content-based routing approaches (e.g., [4, 6, 10, 15]) that commonly rely on distributed filtering trees that require filtering at each level.

Even though a large body of work has focused on optimizing local filtering using intelligent data structures [3, 7, 11, 13, 18], filtering times in the presence of a large number of profiles are typically in the order of tens of milliseconds or even higher, depending on the expressiveness of the data/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. This

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

"per-message-overhead" problem becomes more pronounced when data need to be compressed for transmission (typically employed when transmitting XML data streams [20]), as each broker will then have to incur the cost of decompression and recompression. Furthermore, data may also be encrypted for security reasons (e.g., in financial feeds, networked games etc.). In these scenarios, there will be an additional overhead for encryption/decryption at each broker. The only straightforward way to eliminate these overheads is to eliminate the need to perform content-based filtering, which is the approach SemCast takes.

Second, SemCast semantically splits the incoming data streams and sends each *sub*-stream through a different dissemination channel. This approach makes the system quite efficient and flexible, because multiple bandwidth-efficient dissemination trees can be created, optionally based on the QoS (Quality-of-Service) 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 entire 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 recent work [8, 17], using multiple trees can significantly increase the efficiency and effectiveness of data dissemination.

In the SemCast approach, a cost model is used to perform *channelization*; i.e., deciding how many channels to use as well as the contents and destinations of each channel. The model uses (1) stream contents and rates, (2) profile characteristics, and (3) network locations of the message destinations. Each channel is implemented as a multicast tree, and clients subscribe to the channels by joining the corresponding trees. The system gathers statistics in order to adapt to changes in the profile and stream characteristics. Using statistical as well as syntactic information (i.e., profile containment relationships), SemCast periodically revises its channelization decisions, striving to improve the overall 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 workloads, we compare the SemCast approach to the representatives of traditional content-based routing approaches through a detailed simulation study. Our results show that SemCast not only reduces the processing load on the brokers, but also significantly reduces the overall bandwidth requirements of the system.

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 approach. In Section 4, we discuss our heuristics for latency–aware dissemination tree construction. In Section 5, we present experimental results that characterize the efficiency of SemCast and representatives of traditional content-based dissemination approaches. Section 6 briefly discusses prior related work. Finally, Section 7 concludes with a brief summary of our contributions and directions for future research.

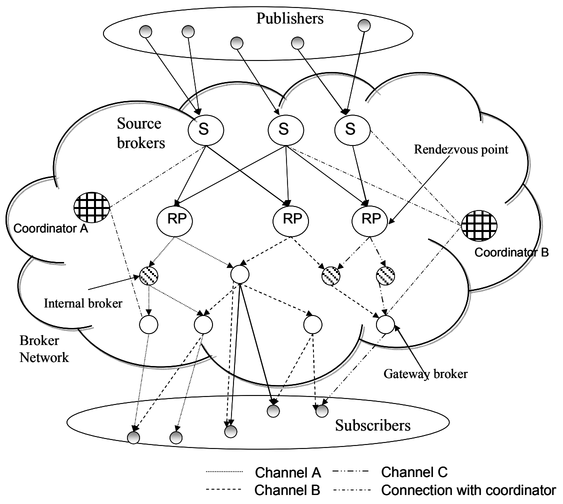

Figure 1. Basic SemCast system model

2. System model

Data and profile model SemCast is based on a content-based publish/subscribe model where clients declaratively express their data needs by specifying predicate-based expressions called *profiles*. SemCast can work with either relational or XML data. Profiles are expressed using simple predicates over the relational schema in the former case, and using a query language (e.g., XPath [23]) in the latter case.

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 only QoS metric addressed in this paper is *staleness*, which measures the end-to-end delay between the time a message is "injected" into the system and the time it is delivered to the client.

Components SemCast consists of a set of servers (a.k.a. brokers or nodes) organized into an application-level overlay network, which is illustrated in Figure 1. 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 the system through *gateway brokers*. Each client can maintain connections with more than one gateway broker. 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 performing channelization and managing the dissemination channels. It communicates with the sources and the gateway brokers in order to inform them about the content of the existing channels. For simplicity of exposition, we assume throughout the paper that there is a single coordinator. In Section 3.1.5, we discuss how to distribute coordinator functionality across multiple nodes to improve the availability and scalability of the system.

Some brokers of the overlay network serve as *rendezvous points*. A rendezvous point is responsible for at least one channel in the sense that it serves as the root of the corresponding dissemination tree.

Channel expressions The content of each channel is defined by a *channel expression*, which is basically a combination of disjunctions and conjunctions of client profiles. A profile is *assigned* to a channel if the channel expression *overlaps* with the profile. In Section 3, we describe the form of the content expressions and explain how they are derived from the client profiles. For a given channel, the corresponding dissemination tree includes all the gateway brokers that have at least one client whose profile is assigned to the channel.

Basic data-flow The coordinator periodically makes channelization decisions and forwards to each source (1) the channel to rendezvous point assignments and (2) the content expression for each channel. Sources receive incoming messages and match them against these expressions to decide which channel(s) to forward them. Once a message is received by a rendezvous point, it is efficiently disseminated, over the corresponding dissemination tree, to the relevant gateway brokers. Due to the potential overlap among the channels, a rendezvous point can receive the same message multiple times, through different channels, and is also responsible for eliminating duplicates. Each (interior) broker maintains a simple *routing table* whose entries map channel identifiers to the descendant brokers in the underlying dissemination tree(s). Upon receipt of a message addressed to a specific channel, only a simple lookup is performed on the channel's identifier and the message is forwarded to the returned descendant list.

Each gateway broker maintains a local *filtering table* mapping its subscribers' profiles to their IP addresses. It also maintains a mapping of channels to its subscribers. Received messages from a channel are matched against the profiles of the subscribed clients and are forwarded only to the interested ones. This local filtering ensures that end clients do not receive irrelevant data.

3. Content-based channelization

3.1.1 Overview

We now discuss the problem of content-based channelization of data streams. The problem requires addressing the following questions: (1) how many channels to create, (2) the contents of each channel, and (3) which channels each broker should listen to. SemCast utilizes a cost-model that uses profile semantics, stream statistics, and network characteristics. Because these factors may change over time, SemCast periodically reevaluates its channelization decisions.

SemCast has two primary operational goals while performing channelization. First, SemCast ensures that there are *no false exclusions*: the assignment of profiles to channels and the channel expressions guarantee that every message will be delivered to all the 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, SemCast's algorithms strive to:

- *eliminate content redundancy* by minimizing the overlap among channel contents.
- *create cost-efficient dissemination trees* using an incremental, distributed tree construction algorithm (inspired by Steiner trees [22]).

3.2 Channelization algorithm

SemCast leverages the knowledge of "overlap" among profiles—we say that two profiles *overlap* if the intersection of the set of messages both match is non empty. Assigning overlapping profiles to the same channel allows the common matching messages to be forwarded only once, through the same dissemination tree, thereby eliminating redundant transmissions. As a second optimization, SemCast attempts to include as part of each dissemination tree only the brokers with overlapping profiles.

As we describe below, SemCast relies on *both* syntactic and statistical information to identify overlapping profiles. An extreme case of profile overlap arises in the form containment relationships—a profile P_1 *covers* (or *contains*) a profile P_2 if and only if all messages that match P_2 also match P_1. SemCast discovers *containment hierarchies* among the profiles, where each profile has as its ancestor a profile that covers it, and as descendants profiles that are covered by it.

SemCast periodically reorganizes the channels' content and membership to reflect the latest statistics in the system. This reorganization includes two phases.

First, SemCast constructs containment hierarchies using both syntactical similarities among profiles and statistics on the stream rates and profile overlap. These hierarchies form the basis of our channelization decisions: each hierarchy is a good candidate for serving as a dissemination channel, because of the overlap among the constituent profiles. However, creating channels solely based on containment information will miss further optimization opportunities as partial overlaps among profiles are ignored. The second phase of the reorganization addresses this issue by identifying partially overlapping profiles and placing them in the same channel in order to further reduce the run-time cost.

3.2.1 Statistical information

SemCast strives to adapt to changes in the rate of the incoming messages and the overlap among profiles. To implement this approach, source brokers continuously collect information about the profiles' selectivity in a *selectivity matrix.* We define the *selectivity* of a profile as the set of messages matching this profile. On each reorganization phase, the coordinator gathers these statistics from each source broker, creating the global selectivity matrix. Moreover, it creates a *profile overlap matrix* to calculate the partial overlap between profiles.

Selectivity matrix This structure is implemented as a sliding window on a sample of the incoming messages and is maintained by each source broker.

Each row refers to a message and each column to a different profile. For each sampled message, the profiles matching this message are identified and the corresponding entries are set to one. The size of this window (i.e., number of rows) should be large enough to collect statistics with high confidence. Samples are given weights based on an exponential decay function. Thus, decisions are biased to favor more recent statistics. This matrix can be efficiently stored as a bit vector.

The selectivity matrix provides us with information about the set of messages each profile has matched during the timeframe of the sliding window. Based on this information, we can easily estimate the message rates (per profile) and the overlap among profiles.

Profile overlap matrix We now define a *partial overlap* metric between profiles. Given two 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.

SemCast computes an *overlap matrix* using the selectivity matrix. Specifically, each entry, O_{ij}, of the overlap matrix represents the overlap value for the profiles P_i and P_j; i.e., $O_{ij}=k$, if $P_i \subseteq^k P_j$.

In the next section, we describe how the overlap information is used in the reorganization.

3.2.2 Construction of containment hierarchies

The reorganization phase attempts to move the system to a configuration with lower run-time cost. Placing two profiles, one more general than the other, on the same channel does not necessarily lead to a reduction in cost, especially if the overlapping part is very small, or if the *non*-overlapping part has high rate.

In order to make the right decision, SemCast places overlapping profiles under the same channel, *if* this could lead to improvement of bandwidth consumption. To achieve that, we use a *cost model* to identify if two profiles, where one is covered by the other, should be assigned to the same channel.

The cost of SemCast, in terms of bandwidth consumption, is determined by three factors:

- The set of semantic channels S, in the system;
- The expected dissemination rate of a channel, r_i, $i \in S$. This rate is defined as the number of messages that are transmitted through this channel per unit time; and
- The edges in the dissemination tree implementing each of the channels, E_i, $i \in S$.

Given this notation, we define the cost of a channel c_i, with profiles $\{P_1, \ldots, P_n\}$ assigned to it, as:

$$c_i = c(\{P_1, \ldots, P_n\}) = \sum_{e \in E_i} r_i \qquad (1),$$

We can now define the cost of SemCast as the sum of the cost of all channels, c_i, $i \in S$:

$$\sum_{i \in S} c_i \, .$$

Containment relations To create the containment hierarchies, SemCast first identifies the containment relations based on the syntax of the profiles. Existing algorithms for identifying containment relations can be found in [6], in the case of relational data, and in [24], in the case of XML data.

Second, SemCast uses the overlap matrix to extend the syntax-based hierarchies as follows. We identify as additional roots profiles those not covered by any other profile; i.e., in the overlap matrix, a root profile P_j should have a single '1' in the j-th column, at the entry O_{jj}. Each non-root profile P_i may have multiple "candidate parents", which are those that contain P_i (can be identified using the syntax-based containment algorithm), as well as those that matched a superset of the messages that matched P_i, in the current sampling window; i.e., in the overlap matrix, the candidate parents P_j of P_i are those having $O_{ji} = 1$, $i \neq j$.

Among all of P_i's candidate parents, we identify the one that gives the most cost reduction (computed based on a simple cost model that we define below), if assigned to the same channel with P_i. This parent is the one that has the highest percentage of overlap with P_i and the lowest rate for the non-overlapping part with P_i.

In Figure 2, we show a simple example of the containment hierarchies that exist among seven profiles. Table 1 shows a possible overlap matrix for these profiles. Here, P_6 has two candidate parents, P_1 and P_4. P_4 is eventually chosen to be the parent of P_6 because it shares a higher percentage of messages with P_6.

Cost model We now describe the details of the cost model used for non-root profiles to pick their parent from among all candidate parents. For each non-root profile P_{child}, we compute the cost of two scenarios. First, we compute the cost of assigning P_{child} as a root to a new hierarchy:

$$c(H_{parent})+c(\{P_{child}\}).$$

where H_{parent} refers to the set of profiles in the hierarchy rooted by P_{parent}. Furthermore, for each candidate parent P_{parent}, we compute the cost of assigning P_{child} to H_{parent}:

$$c(H_{parent}\cup\{P_{child}\})$$

We then identify and apply the lowest-cost scenario.

Cost approximation To estimate the cost of the above scenarios, we use the cost formula (1). Thus, we need to identify the rate of messages matching the profiles involved in each scenario and the number of edges in the dissemination tree that spans all the corresponding gateway brokers. The rate of the matching messages can be derived from the selectivity matrix.

In order to be able to use formula (1) we need to estimate the number of edges in a dissemination tree spanning a given set of gateway brokers. To achieve this, we assume the corresponding gateway brokers connect to the dissemination tree in random order. The first broker is assumed to be connected to the rendezvous point through a shortest path. The consequent brokers connect to the closest one of the gateway brokers already attached to the tree. The coordinator calculates the shortest paths between two brokers, using location information collected from the brokers during run-time.

Channel content expressions Some containment relations derived during a reorganization phase are based on the selectivity history. However, relying only on statistical information can lead to false containment relations. Therefore, to preserve the no-false-exclusion property, SemCast represents the root expression of each hierarchy as a disjunction of the profiles in the hierarchy. Thus, if profiles P_i, i=1,…,k, are assigned to the hierarchy H, the root expression of the hierarchy is:

$$E = P_1 \vee P_2 \vee ... \vee P_k .$$

Moreover, SemCast assigns a profile that is not present in any containment hierarchy, either because its selectivity is zero or because no syntactic containment information could be derived, to a new channel. This assures that all profiles are assigned to at least one channel. In this way, every client interested in these profiles can subscribe to the corresponding channels and therefore receive all relevant messages.

$$\begin{tikzcd}
P_4 \arrow[d] \arrow[dr] & & P_1 \arrow[dl, dashed] \arrow[d] \\
P_3 \arrow[d] & P_6 \arrow[d] & P_5 \\
P_2 & P_7 &
\end{tikzcd}$$

Figure 2. Containment hierarchy from Table 1

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

3.2.3 Hierarchy merging

Even though periodic re-evaluation of the containment hierarchies and channel reorganization can improve the efficiency of the system, a small degree of containment among profiles could result in a large volume of redundant messages. The reason is that containment hierarchies cannot identify profiles with only partial overlap, and, thus, the reorganization phase would assign them to different channels, increasing the overall bandwidth consumption.

To address this problem, SemCast places different containment hierarchies with significant partial overlap in the same channel. However, perhaps counter to intuition, such a *merging of hierarchies* may not necessarily reduce redundant transmissions. Even though the messages common to both hierarchies will now be transmitted only once, the *un*common messages will be forwarded to the destinations not interested in them. For this reason, SemCast merges the common parts of two hierarchies and forwards their uncommon parts through two "noise channels" *if* this operation is expected to decrease the bandwidth consumption.

Specifically, if two hierarchies with root expressions E_i and E_j are merged, then three channels with the following content expressions and destinations are created:

1. $E_i \wedge E_j$ and $D(E_i) \cap D(E_j)$,
2. $E_i \neg E_j$ and $D(E_i)$, and
3. $E_j \neg E_i$ and $D(E_j)$,

where $D(E)$ represents the set of gateway brokers having at least one client interested in one or more of the profiles that appear in expression E. The first channel above contains the content common to both hierarchies. The second and third channels are the noise channels that carry content assigned to only one of the hierarchies.

Cost model To implement the merging operation, SemCast makes a pair-wise comparison of the root profiles using the overlap matrix: if there is partial overlap between P_j and P_i, i.e., $P_i \subseteq^k P_j$ or $P_j \subseteq^k P_i$ where $k>0$, then we compute the cost of the current configuration:

$$c(H_i)+c(H_j),$$

and compare it with the cost of the configuration after the hierarchies are merged and the noise channels created:

$$c(H_i \cap H_j)+c(N_i)+c(N_j).$$

We define the cost of a noise channel, N_i, as:

$$c(N_i)=\sum_{e\in E} r,$$

where E is the set of edges of the dissemination tree that spans the gateway brokers whose clients have the profiles in hierarchy H_i, and r is the expected dissemination rate of the noise channel N_i. This latter value denotes the number of messages matching the profile set of hierarchy H_i, but not the profile set of H_j, per unit time. To estimate the number of edges in the dissemination tree, we follow the same procedure as in the case of hierarchy construction.

Among all pair-wise merging operations, we identify and apply the one that provides the highest cost benefit. We continue considering pair-wise merging operations among the new merged hierarchies and the non-merged ones, until no further cost reduction can be achieved. The noise channels are not considered for further merging (in Section 5, we study an alternative approach that also considers the noise channels for merging). Since this merge procedure has exponential cost, it is stopped after a fixed number of iterations or a specific time period.

Once the merging procedure terminates, a new set of channels is created to reflect the new set of hierarchies (non-merged and merged ones). Brokers are informed by the coordinator about the new channels to which they should subscribe to, while sources receive the new channel lists and the corresponding content expressions.

3.2.4 Subscription management

In this section, we describe how SemCast handles subscription and unsubscription requests. Upon receipt of a new profile from a client, the gateway broker sends the profile to the coordinator. Using a syntax-based containment algorithm, the coordinator checks if an existing channel covers the profile. If a covering channel exists, then the profile is simply assigned to this channel. If there is no covering channel, a new channel is created for the profile. In either case, the gateway broker is notified of the decision, and joins the corresponding dissemination tree.

If a client wants to remove its subscription, it sends a request to its gateway broker. The 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 messages from that channel, the gateway broker removes itself from the channel and the corresponding dissemination tree by sending an unsubscription request to its parent in the dissemination 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 hop closer to the source. Once a channel is left with no gateway brokers, it is removed from the system by the coordinator.

3.2.5 Multiple coordinators

The functionality of the coordinator can be easily distributed to improve the scalability and availability of the system. If multiple coordinators exist, each will be responsible for maintaining information (i.e., containment hierarchies, channel expressions) related to a specific subset of the attributes appearing in the profiles. If a channel expression or a containment hierarchy contains profile expressions including more than one attribute, this information will be maintained in each one of the coordinators responsible for these attributes. Gateway brokers will forward their subscription requests to a coordinator among the ones responsible for the attributes appearing in a profile. In the reorganization phase, the coordinators synchronize their data, and one of them is elected to perform channelization.

3.2.6 Filtering cost of content expressions

After hierarchy construction and merging operations, the channel content expressions can have one of the following forms:

1. $E_i = P_1 \vee P_2 \vee ... \vee P_k$, for non-merged channels,
2. $E_i \wedge ... \wedge E_K$, for merged channels and
3. $(E_i \wedge ... \wedge E_k) \neg (E_j \wedge ... \wedge E_m)$, for the noise channels.

For correctness, all SemCast's sources need to maintain a list with the content expressions of every channel available in the system. Every incoming message must then be matched against these expressions and be forwarded to the appropriate channels. The filtering cost at the source can be reduced by using a centralized filtering engine (e.g., [3, 7]). Since each distinct profile will appear in at least one content expression, the filtering engine is used to match the incoming messages against all distinct profiles.

Once we determine all the profiles the incoming message matches, we then need to identify the corresponding channels. Therefore, the content expression of each channel, which can be in one of the forms shown above, should be evaluated efficiently. This procedure boils down to an evaluation of a set of Boolean expressions, one for each channel. To improve the time of expression evaluation, an algorithm like UNISON [21] can be used. This algorithm performs fast evaluation of arbitrary Boolean expressions in software with small execution time and memory space requirements.

4. Dissemination tree construction

SemCast relies on a distributed and incremental approach to create two 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 similar interests in the same dissemination tree. Moreover, the tree construction heuristics strive to create dissemination 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, dissemination trees will mainly include brokers interested in the same content, avoiding messages forwarding to an increased number of internal brokers.

To implement this approach, a gateway broker receives from the coordinator the 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 the 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., [22]), 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 dissemination tree using the low-cost heuristic described above. All brokers continuously track their “distance” from the root of the channels they maintain. Thus, upon the receipt of a delay-bounded join request, the brokers can estimate if they can satisfy the latency constraint. 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 SemCast, comparing it against representatives of traditional content-based filtering approaches, using detail simulations over realistic network topologies.

Simulation setup In the simulation, we used an overlay network of brokers, where the topologies are random graphs generated with GT-ITM [26]. In the experiments, we used networks of up to 600 broker nodes and 7000 clients.

In order to control the impact of profile similarities, we used a parameter called the *profile selectivity factor*, which determines the number of distinct profiles in the profile set. Specifically, 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. Note that partial overlap or containment relationships may still exist among distinct profiles. Other simulation parameters of importance are described in the text wherever appropriate.

Each client picks its profile randomly from the profile set and then chooses a single gateway broker to connect, again randomly. To distinguish between edge nodes and internal nodes, the number of gateway brokers is fixed at 20% of the network size, unless stated otherwise. Each message publisher also connects to a single source broker that it randomly chooses, and forwards all its messages to that broker. The coordinator picks the rendezvous point for each channel, also at random.

Metrics 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 tables at each broker, in order to characterize the processing cost required for message filtering.

Studied dissemination approaches We simulated several approaches for content-based stream 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]). Second, we simulated an approach that models a conventional distributed publish-subscribe system (e.g., [10, 15]). In this model, profiles are propagated and aggregated up to the root over shortest paths from each client to the source. In this way, a predicate-based filtering tree is created on top of a shortest-path spanning tree of the broker network. Filter-based routing tables are

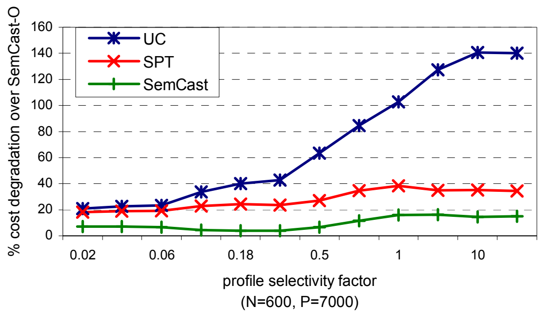

Figure 3. Relative bandwidth efficiency for different approaches

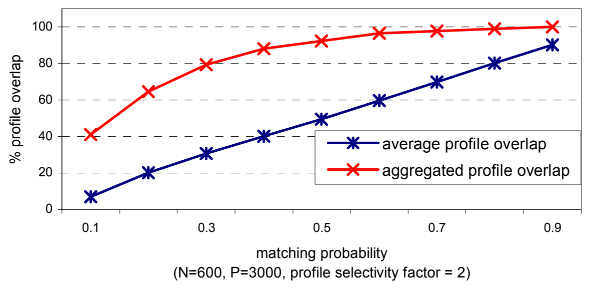

Figure 4: Profile overlap for varying matching probability values

placed at each broker and messages are matched at each hop against the filters in the filtering table to determine the next hop(s) towards the clients. We refer to this algorithm as *SPT* (Shortest Path Tree).

Besides the alternative approaches, we also simulated three different versions of SemCast in order to gain more insight on the effects of our merging and tree construction heuristics. *SemCast-NoiseMerge* is an extension of SemCast where the noise channels are also considered for merging along with hierarchies. *SemCast-Merge* is another variant where the merging operations do not create any noise channels, but only one channel containing the profiles of the two participant hierarchies. Finally, *SemCast-O* is identical to SemCast except that it employs a centralized Steiner-tree construction algorithm [22], which uses global knowledge of all the profiles and the destinations for each channel.

5.1 Basic performance

Disjoint profiles We first study the case when there is no partial overlap among the profiles. In this scenario, SemCast does not execute its second phase that merges partially-overlapping containment hierarchies. Thus, different merging heuristics do not come into play and all SemCast's variants perform the same. The only exception is SemCast-O, which uses a different tree construction algorithm.

Figure 3 shows the cost degradation (i.e., extra bandwidth consumption) of the UC, SPT, and SemCast over SemCast-O, for 600 nodes and 7000 profiles.

SemCast performs very close to SemCast-O, incurring up to only 6.4% extra cost on the average. This result reveals that SemCast's distributed tree construction heuristic performs reasonably close to an optimized, centralized algorithm (SemCast-O is omitted from the rest of the results, as it always slightly outperforms SemCast). Note that the scale on the x-axis is non-uniform. SPT performs better than UC but worse than SemCast. SPT incurs an extra 23% overhead on the average, whereas UC incurs an average increase of 41%.

In the remainder of the experiments, we use a selectivity factor of two, as this setting pretty much coincides with the knees of the curves of all approaches, and therefore serves as a good representative value.

Overlapping profiles We now turn to cases that involve partial overlap among profiles. We use a parameter called *matching probability* to control the degree of profile overlap. We define *average profile overlap* as the average partial overlap over all profile pairs. We study SemCast's bandwidth efficiency as a function of the *aggregated profile overlap*, which represents the average overlap between any two gateway brokers' "interests". Specifically, for each gateway broker, we computed the cumulative set of messages matching the local profiles and calculated the average partial overlap between these sets over all gateway brokers' pairs. Figure 4 shows how the aggregated profile overlap changes as we increase the average matching probability.

Figure 5 shows the cost improvement of SemCast and UC compared with SPT for increasing aggregated profile overlap values. The results show that SemCast always performs better that SPT and UC, even in the presence of partial overlap.

The poor efficiency of UC regardless of profile overlap (Figure 3 and Figure 5) is not surprising since UC requires every message to follow an independent (shortest) path to its destinations. SPT eliminates some of this redundancy, due to the filtering at each broker. However, the 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 including brokers not interested in the same messages. SemCast strives to minimize this message redundancy *by placing on the same dissemination tree only the brokers interested in receiving the same messages*. Our algorithm identifies the set of gateway brokers interested in the same content, while our tree construction heuristic connects them using a low-cost dissemination tree. Thus, SemCast can achieve

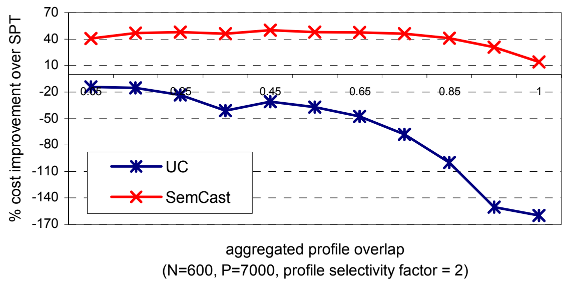

Figure 5. Relative bandwidth efficiency for varying profile overlap values

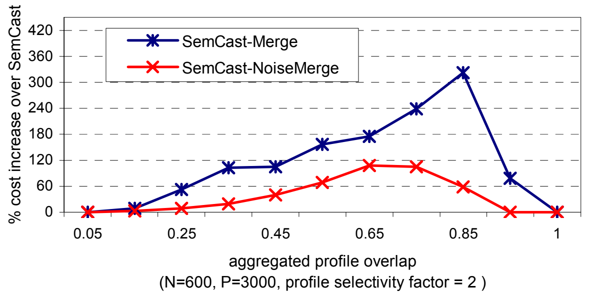

Figure 6. Relative bandwidth efficiency of SemCast's variations

a cost reduction up to 50% over SPT. These results reveal the significant advantages of the multi-tree semantic multicast approach over the single-tree distributed filtering-based approaches.

Figure 5 also shows that the advantage of SemCast over SPT decreases as the profile overlap increases, when profiles partially overlap. Although SemCast attempts to eliminate all redundancy through its merge operation, some message redundancy is still present in the system due to the overlap among the channels that partially overlap but are not merged. As the profile overlap increases, this overlap also increases, increasing the message redundancy and thus the bandwidth consumption. However, even when all the profiles match the same set of messages and a single tree is created, SemCast performs better than SPT. The reason is that SemCast creates an approximation of a Steiner tree, including only the brokers interested in the messages, while SPT uses a shortest path spanner of the entire broker network.

5.1.1 SemCast's variations

Figure 6 shows the extra cost incurred by SemCast-Merge and SemCast-NoiseMerge over the plain SemCast approach. Both SemCast's variants perform worse than SemCast for all overlap values. Figure 7 provide complementary information by plotting the number of channels constructed by the three approaches. These results provide more insight on two important issues that we address in SemCast: (1) whether to create noise channels, and (2) whether to include these channels in the merging process.

Initially, when the profile overlap is low, all approaches create the same number of channels and incur similar run-time costs. As the overlap increases (between 25% and 95%), SemCast-Merge merges some of the containment hierarchies. However, the number of channels remains almost constant. As the partial overlap increases so does the bandwidth consumption due to the increase in redundant messages. SemCast is able to avoid part of this redundancy by creating the noise channels, thus proving the benefits of the noise channels. Clearly, the construction of noise channels increases the number of channels compared with SemCast-Merge. Once the profile overlap is sufficiently high (i.e., above 95%), it becomes possibly for SemCast-Merge to merge more profitably, without creating much "noise", and improve its performance. Figure 7 clearly shows that when the profile overlap is high, most of the profiles are placed on the same channel.

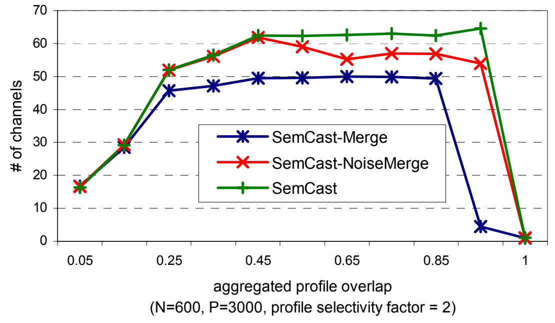

Figure 7. Number of channels for varying profile overlap values

We also observe that, beyond a 10% profile overlap, both SemCast and SemCast-NoiseMerge start creating noise channels, the reason why they both create more channels than SemCast-Merge. It also seems that the merging of noise channels is not beneficial, since SemCast-NoiseMerge has higher cost than SemCast for most of the shown overlap range (i.e., 25%-95% profile overlap).

It is also clear that there is a tradeoff between the number of channels and overall bandwidth consumption—creating more channels with less overlap can decrease run-time costs significantly, at the expense of an increase in the total number of channels that need to be stored and maintained.

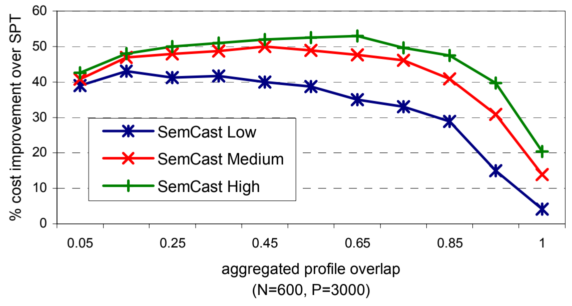

Figure 8. Relative bandwidth efficiency for different profile selectivity values

5.1.2 Other results

Profile Selectivity Figure 8 shows percentage cost improvement of SemCast over SPT, for three selectivity values: 1 (Low), 2 (Medium), and 4 (High). We see that higher selectivity factors result in higher cost improvement for SemCast when compared with SPT. The reason is that high selectivity values result in fewer distinct profiles in the profile set, and thus the content redundancy problem occurs to a lesser extent.

Network size We now investigate the scalability of SemCast as a function of the number of brokers in the system. Figure 9 compares SemCast with SPT for different network sizes, while keeping the number of profiles per broker fixed. The results show that, the larger the network size, the higher the improvement SemCast achieves. On the contrary, the unicast approach performs worse as the network size increases (not shown).

5.2 Latency-aware dissemination

We now study the efficiency of SemCast for delay-bounded dissemination. For simplicity, we assume that all subscribers express the same latency bounds and no partial overlap exists among profiles. Latency is abstractly measured by the number of broker hops each message takes, from the rendezvous point to the destination gateway broker(s). We ran our low-cost heuristic and delay-bounded heuristic using the same set of profiles. For the latter case, however, we varied the latency expectations of our subscribers.

Figure 10 shows the tradeoff between bandwidth efficiency and dissemination latencies by plotting the penalty over the unbounded-latency case, as a function of latency bounds, for three different network sizes. As we relax the latency bounds, the penalty diminishes. 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 in the presence of tight latency bounds, SemCast is still able to achieve lower bandwidth consumption than SPT (since the difference between SPT and SemCast is always above 10%).

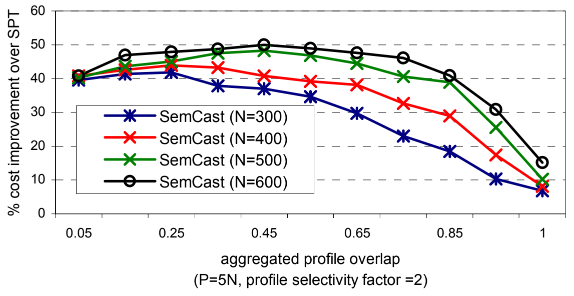

Figure 9. Relative bandwidth efficiency for different network sizes

We also measured the depth of the dissemination trees and observed that, for an overlay network of 600 brokers, SemCast creates trees that are on the average 2-2.5 hops deeper than the corresponding trees in SPT. Thus, if latency is crucial, SemCast can use the latency bounding techniques to effectively create shorter trees at the expense of increased bandwidth consumption.

5.3 Processing cost benefits

One of the main advantages of SemCast is that it eliminates the need for content-based filtering at intermediate brokers. In SemCast, messages are filtered only at two network locations: (1) at the source brokers, to identify the channels each message should be forwarded to, and (2) at the gateway brokers, to send only the matching messages to each client. The need for filtering at the gateway brokers may not be obvious. Even though the

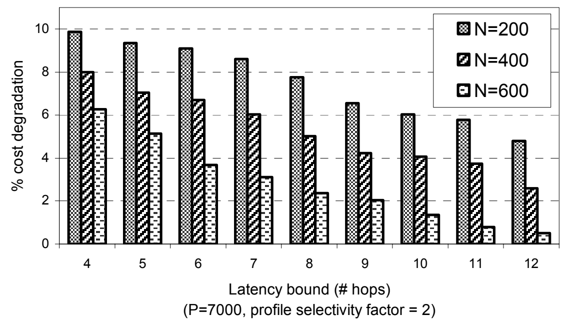

Figure 10. Bandwidth vs. latency tradeoff

gateway broker knows, for each of its clients, the channel(s) to which the client is assigned, those channel(s) likely carry messages irrelevant to the 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 gateway brokers, as a percentage of the profiles that the root in SPT needs to maintain, for different aggregated profile overlap values. The table also shows the number of profiles placed at the non-root brokers of the tree constructed by SPT. The number of profiles in the source of SemCast is similar to that at the source of SPT, so they are not included in the table.

In SPT, the number of profiles 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 clients, the average number of profiles at the brokers decreases significantly.

Table 2. Profiles on local filtering tables (N=600, P=3000, profile selectivity factor = 2)

Aggregated Profile overlap	**0.02**	**0.03**	**0.05**	**0.08**
SPT (at hop 1)	5.75%	8.45%	10.79%	15.48%
SPT (at hop 2)	3.11%	6.44%	5.88%	9.59%
SemCast (at gateway)	0.19%	0.45%	0.33%	0.59%

In SemCast, the amount of filtering performed at the gateway brokers 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 from another perspective, require much less processing horsepower for the same data rates.

6. Related work

Filtering systems A large body of work has focused on optimizing centralized filtering using intelligent data structures. For instance, XFilter [3] and YFilter [14] convert queries to finite-state machine representations and create matching algorithms for fast location and evaluation of profiles. In [11], a space-efficient index structure, which decomposes tree patterns into collections of substrings and index them using a trie, was proposed to improve filtering efficiency. The focus of SIFT [25] was also on the efficient filtering of data against attribute-value-based client subscriptions.

None of these approaches addressed networked dissemination. SemCast can use these approaches for efficient content-based filtering at its source and gateway brokers.

Content-based routing Unlike SemCast, existing approaches for content-based networking do not address the issue of constructing application-level dissemination networks. Gryphon [4] assumes that the best paths connecting the brokers are known a priori. Likewise, Siena [6] also assumes that an acyclic spanning tree of the brokers is given.

More recently, XRoute [10] and ONYX [15] introduced efficient solutions for content-based publish/subscribe for XML data and XPath-based profiles. These systems primarily focused on efficient distributed filtering, whereas SemCast's focus is on networked dissemination. As with the SPT approach that we studied, these systems have the disadvantage of requiring content-based filtering at the intermediate brokers. Moreover, our results showed that creating multiple dissemination trees, instead of sending all the information through a single tree, decreases the overall bandwidth consumption, even when there is low profile overlap.

Content-based routing was 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, another key difference of SemCast from previous approaches is that it leverages knowledge of stream statistics as well as profile and network characteristics in order to optimize and adapt its operation. To the best of our knowledge, previous solutions relied primarily on containment relationships identified through syntactic analysis.

Application-level multicast Many recent research proposals (e.g., [9, 27]) employ *application-level multicast* for efficiently delivering data to multiple receivers. Members of a multicast group typically self-organize into an overlay topology, over which dissemination trees are created. These approaches assume a traditional multicast model, where all members of the multicast group have exactly the same interests. In our context, constructing a different multicast group for each distinct profile is not a scalable solution. To avoid this scalability problem, SemCast uses containment and partial overlap relationships in the profile set, and thus can place in the same multicast channel clients with similar profiles.

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 or 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 application-level overlays was investigated in [20]. This work used multiple interior brokers on the overlay for redundant transmission of messages in order to reduce loss rates and delivery latencies.

Channelization The channelization problem was first 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 among the receivers' interests, makes the problem more even harder. Topic-based semantic multicast was introduced in [12]. In this work, users express their interests by indicating a specific topic area. Similarity ontologies are then 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. Finally, a preliminary design of our semantic multicast approach was presented earlier [19].

7. Conclusions & future work

We described a novel semantic multicast system, called *SemCast*, which takes a radically different approach to content-based dissemination of data streams. The key idea is to split the data streams based on their contents, rates, and destinations, and spread the pieces across multiple *channels*, each implemented as an independent application-level dissemination tree. Because even simpler forms of the channelization problem are NP-complete, we use a practical, efficient cost-based approach that leverages the knowledge of profile, stream, and network characteristics.

The primary advantages of the SemCast approach over traditional approaches are that (1) it does not require content-based filtering of streams at interior brokers of the overlay network, and (2) it facilitates fine-grained control over the construction of efficient dissemination trees. We argued and quantitatively demonstrated through simulations that, in addition to reducing the processing load on the system, SemCast exhibits significantly better bandwidth efficiency, compared to traditional approaches.

There are two immediate directions for future work. First, we are designing batched and priority-driven dissemination techniques to improve system throughput and effectiveness. Second, we are incorporating storage and ad hoc query functionality into SemCast.

We are currently in the process of building a SemCast prototype to verify the practicality of the basic approach and to demonstrate the claimed performance benefits through deployment on PlanetLab.

8. References

[1] D. Abadi, et al. Aurora: A Data Stream Management System (Demo). In *SIGMOD*, 2003.

[2] M. Adler, et al. 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, et al. An Efficient Multicast Protocol for Content-Based Publish-Subscribe Systems. In *ICDCS*, 1999.

[5] D. Carney, et al. Monitoring Streams: A New Class of Data Management Applications. In *VLDB*, 2002.

[6] A. Carzaniga, et al. 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, et al. SplitStream: High-bandwidth content distribution in cooperative environments. In *SOSP*, 2003.

[9] M. Castro, et al. SCRIBE: A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in communications (JSAC)*, 20(8), 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, et al. Efficient Filtering of XML Documents with XPath Expressions. *VLDB Journal, Special Issue on XML*, 11(4):354-379, 2002.

[12] S. Dao, et al. Semantic multicast: intelligently sharing collaborative sessions. *ACM Computing Surveys (CSUR)*, 31(2es), 1999.

[13] Y. Diao, et al. 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, et al. 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, et al. Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh. In *SOSP*, 2003.

[18] L. Opyrchal, et al. 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, et al. Mesh-Based Content Routing using XML. In *SOSP*, 2001.

[21] R. Sosic, et al. The Unison algorithm: fast evaluation of Boolean expressions. *ACM TODAES*, Volume 1(Issue 4), 1996.

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

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

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

[25] T. W. Yan and H. Garcia-Molina:. Efficient Dissemination of Information on the Internet. *IEEE Data Eng. Bull.*, 19(3):48-54, 1996.

[26] E. Zegura, et al. How to Model an Internetwork. In *INFOCOM96*, 1996.

[27] S. Q. Zhuang, et al. Bayeux: An Architecture for Scalable and Fault-tolerant Wide-area Data Dissemination. In *NOSSDAV*, 2001.