

Semantic Multicast for Content-based Stream Dissemination¹

Olga Papaemmanouil
Department of Computer Science
Brown University
Providence, RI, USA

olga@cs.brown.edu

Uğur Çetintemel
Department of Computer Science
Brown University
Providence, RI, USA

ugur@cs.brown.edu

ABSTRACT

We consider the problem of content-based routing and dissemination of highly-distributed, fast data streams from multiple sources to multiple receivers. Our target application domain includes real-time, stream-based monitoring applications and large-scale event dissemination. We introduce *SemCast*, a new semantic multicast approach that, unlike previous approaches, eliminates the need for content-based forwarding at interior brokers and facilitates fine-grained control over the construction of dissemination overlays. We present the initial design of SemCast and provide an outline of the architectural and algorithmic challenges as well as our initial solutions. Preliminary experimental results show that SemCast can significantly reduce overall bandwidth requirements compared to traditional event-dissemination 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.

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*, 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, 5] 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, 6, 9, 11, 13], the forwarding times in the presence of a large number of profiles are typically in the order of tens of milliseconds or higher (depending on expressiveness of the data and profile model, and the number of profiles) [3, 6]. 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 [15].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Copyright is held by the author/owner.
International Workshop on the Web and Databases (WebDB),
June 17-18, 2004, Paris, France.

¹ This work was supported by the National Science Foundation under grant ITR-0325838.

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 required 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, 5]. As we demonstrate, these approaches fail to recognize many opportunities for creating better optimized dissemination trees. Moreover, as shown by the recent work [7, 12], 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 based on stream and 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, striving to improve the overall bandwidth efficiency 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 content-based channelization algorithm. In Section 4, we discuss our heuristic for QoS-aware tree construction. In Section 5, we present preliminary experimental results that characterize the bandwidth savings that can be achieved by SemCast over traditional content-based routing approaches. Section 6 briefly discusses prior related work and, finally Section 7 concludes with a brief discussion of our contributions and directions for future research.

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 [17]. Each client profile is also associated with a QoS specification that expresses the client's service expectations [1]. A QoS specification is a constraint on a specific metric of interest: our initial concern is *staleness*.

The system consists of a set of brokers organized as a P2P overlay network. Each publisher and subscriber is connected to some broker in the network; we refer to these brokers as *source brokers* and *gateway brokers*, respectively. Brokers with no publishers or subscribers connected to them are called *interior brokers*.

SemCast also includes a *coordinator* node. The coordinator is responsible for maintaining the dissemination channels and deciding their content. Specific brokers of the overlay will also serve as *rendezvous points (RPs)*. Each RP is responsible for at least one channel and serves as the root of the corresponding multicast tree. This network model is shown in Figure 1.

Sources receive from the coordinator the current XPath channel content expressions. Any modification to these

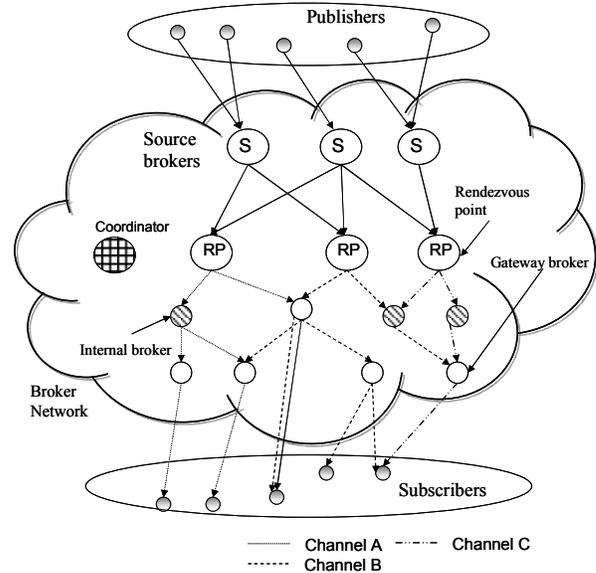


Figure 1: Basic SemCast system model

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 *RPs*, 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. 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.

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 content of the 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 XML 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

² Our approach is not restricted to XML streams and is also readily applicable to relational data.

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 statistical information obtained from the system at run time. In the rest of the section, we describe these two phases in more detail.

3.1 Initial Channelization

In SemCast, the initial content assignment to channels is based on a syntax-based containment function, which identifies containment relations among profiles. To achieve this, we use existing algorithms for identifying containment relations among XPath queries [18]³. We say that an XPath expression P_1 *covers* (or *contains*) another 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 [18].

Upon receipt of a new profile from a client, a gateway broker *GB* checks if the client can be served by the channels to which *GB* is 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 relevant content to the gateway brokers. The coordinator informs the *GB* of the channel C it should subscribe to, and the *GB* joins the corresponding multicast tree. We then say that P_{new} is assigned to channel C .

As profiles are assigned to channels based on containment relations, *syntax-based 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.

3.2 Dynamic Channelization

The containment algorithms used in the initial channel assignment are based on the structure of XPath expressions. However, no conclusions can be inferred for the overlap of matching documents when no containment relation is found among a profile and a channel content expression. This implies that channels may not disseminate disjoint sets of documents, leading to multiple copies of the same document being sent

through different channels. Thus, initial channelization based on syntactic analysis alone is unlikely to produce good results.

SemCast continuously re-evaluates its channel assignment decisions based on run-time stream rates and profile similarities, attempting to minimize the overall bandwidth consumption. To implement this, SemCast creates *rate-based containment hierarchies* existing among profiles. Periodically, SemCast creates the rate-based hierarchies to decide whether the channels’ contents and membership should be reorganized. Rate-based hierarchies reflect how the channels’ membership and content should be organized, based on the latest stream rates and profiles overlap. If the current state of SemCast “sufficiently” differs from the state shown in the rate-based hierarchies, channel reorganization takes place. In the following, we explain the dynamic channelization phase in more detail.

3.2.1 Rate-Based Containment Hierarchies

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 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 .

In order to reduce the overlap among the contents of the channels, SemCast places profiles matching similar messages under the same channel. Thus, we construct the rate-based hierarchies such that each profile P_j is an ancestor of P_i if P_j covers P_i . Among multiple candidate parents for P_i , SemCast chooses the one with (1) higher overlapping part with P_i and (2) lower stream rate of messages matching the non-overlapping part between P_i and the candidate parent. Using the k -overlap metric defined above, the best parent P_j is the one with $P_i \subseteq^k P_j$ and

$$k = \max_{P_j, j \neq i} \left\{ k / r_{j-i} \mid P_j \subseteq^k P_i \right\},$$

where r_{j-i} is the stream rate matching the non-overlapping part between P_i and P_j .

To create the rate-based hierarchies, SemCast maintains information about the selectivity of the profiles in a *selectivity matrix*. This matrix 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 decays 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 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.

We note that although an array of size $p \times w$ has to be stored (where p is the number of profiles and w the size of the sliding window), only the profiles of the matching channels 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 hierarchy.

³ In the case of relational data, where profiles (*attribute, value*) pairs, containment relations defined in [5].

Profile overlap estimates. SemCast calculates the pairwise partial overlap metric of the profile set in order to identify the current rate-based hierarchies. 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 . One example is shown in the Figure 2 and Table 1, where we assume that r_{j-i} is equal to 1. The dashed line shows that P_1 is a candidate parent for P_6 , but it is not its final parent because of the low overlap they share.

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 $O_{ji}=1, i \neq j$; (c) the best candidate parent of P_i , is P_j such that

$$j = \max_{j, i \neq j} \{O_{ij} / r_{j-i} \mid O_{ji} = 1\}.$$

Once the coordinator has decided on the new rate-based hierarchy, it identifies if there are new channels to be created or removed and which profiles should migrate to other channels. This is a straightforward procedure involving comparisons between the root profiles of the current channels and the roots of the new rate-based hierarchies. 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.

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

3.2.2 Hierarchy Merging

Highly diverse profiles could result in a large number of channels. This scenario may arise when only a small percentage of the profiles is covered by another. At the extreme case, the number of channels could be equal to the number of distinct profiles in the system. In this case, the system will not be able to exploit the actual 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 in the same channel the containment hierarchies that overlap. However, *hierarchy merging* could increase redundancy, 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 channels with a content overlap above a predefined threshold. Moreover, the channels to be merged should have low stream

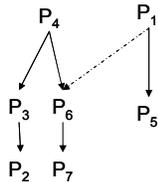


Figure 2. Containment hierarchy from Table 1

rate for their non-overlapping parts. This strategy attempts to allow merging operations which cause the least possible data redundancy among channels.

To implement this SemCast consults the overlap matrix. We compare two root profiles P_j and P_i and check their partial overlap, given by O_{ij} . If this overlap is above the merging threshold, then the hierarchies of P_i and P_j are placed on the same dissemination channel. The root profile of the channel is set to the disjunction of P_i and P_j .

We note that the dynamic channelization contains implicitly the hierarchy splitting operation. Assume that by the time for 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 briefly describe in the rest of this section.

Low-cost trees. In order to connect to a channel, a gateway broker, GB , first receives a list of the current destinations in the channel from the coordinator. GB 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 the GB 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. In order to reduce message replication, we try to reduce the number of edges in the trees. This algorithm is essentially a distributed and incremental adaptation of centralized Steiner tree construction algorithms (e.g., [16]).

Delay-bounded trees. SemCast can also create delay-bounded dissemination trees if clients indicate latency expectations (as part of their QoS specifications). As before, a GB first finds a connection point, CP , in the tree using the low-cost heuristic described earlier. All brokers continuously track their “distance” from the root of the channels they maintain. If the CP 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. 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 GB will be connected through the shortest path to the root.

5. PRELIMINARY EVALUATION

We performed a preliminary evaluation of SemCast using simulations over broker networks, generated using GT-ITM [19]. In the experiments, we used networks of up to 700 nodes and 7000 profiles. In order to study the impact of profile similarities, we use a parameter called the *profile selectivity factor*, which determines the number of distinct non-overlapping profiles. Unless otherwise noted, no partial overlap exists among channels; so there is no need to merge hierarchies.

We simulated four different approaches to content-based dissemination. First, we simulated an approach that models a traditional, distributed Publish-Subscribe system (such as SIENA). In this model, profiles are propagated and aggregated

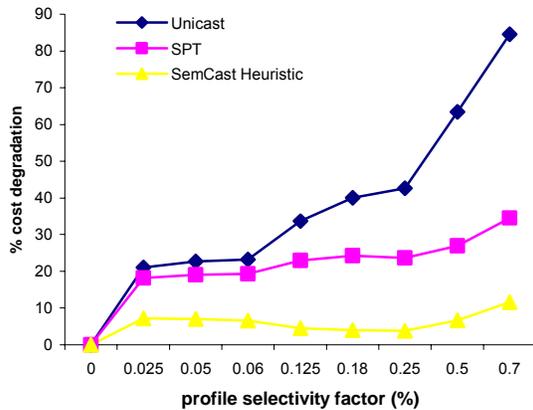


Figure 3. Relative bandwidth efficiency for varying profile similarity factors

up to the source over shortest paths from the clients to the source. In other words, we created 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*.

Second, we simulated our tree construction heuristic described in the previous section. Third, we simulated a centralized Steiner tree construction algorithm [16] where the coordinator knew a priori all the profiles and the destinations for each channel. This approach provides a good approximation to the optimal solution; we therefore refer to it as the *Optimal* approach.

Finally, we simulated a *Unicast* approach, where messages are filtered at the sources and are forwarded to the interested parties using shortest paths. This last approach models a centralized filtering system that uses unicast dissemination of filtered messages.

Basic Performance. Figure 3 shows the cost degradation (i.e., extra bandwidth consumption) over the optimal algorithm for the other three approaches for 700 nodes. SemCast performs the closest to the optimal, incurring up to only 6.4% higher cost on the average. SPT performs better than Unicast; it does, however, increase the cost by 23% on the average, whereas Unicast’s average cost increase is 41%.

Figure 4 shows that the cost degradation increases with increasing network size for Unicast and SPT. At the same time, SemCast’s heuristic is able to keep the bandwidth consumption lower than 8.5% of the optimal solution, for all network sizes.

Benefits of Dynamic Reorganization. We also measured how dynamic reorganization can improve bandwidth consumption when there is partial overlap among channels and, thus, hierarchy merging is used to reduce data redundancy. We observed that, using a merging threshold of 40% partial overlap among hierarchies can reduce the cost up to 23%, compared to SPT. As the partial overlap among profiles increases, SemCast also improves its bandwidth efficiency. In cases where a profile is contained by another with probability less than 0.7, SemCast has a cost improvement up to 21% (with a 30% merging threshold and 1% profile selectivity factor).

We also conducted experiments with varying number of profiles and selectivity factors. The results show that even with large profile sets, SemCast can significantly improve bandwidth

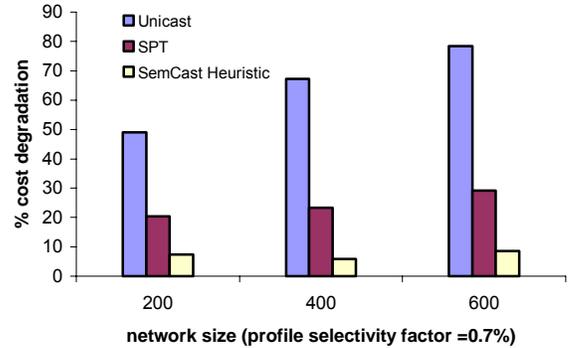


Figure 4. Relative bandwidth efficiency for different network sizes

efficiency compared to the other approaches. Finally, as the profile selectivity factor in the system increases, we can dynamically set the merging threshold to higher values and continue to achieve significant cost improvements.

QoS-Aware Dissemination. We also studied SemCast’s performance when subscribers express latency constraints. Here, there is a fundamental tradeoff between dissemination costs and latencies: the stricter the delay expectations are, the higher the cost of dissemination is. Our results clearly demonstrated this tension, but also showed that the cost increase due to the latency constraints typically does not go beyond 10%, even with low profile selectivity factors (i.e., 1%). Due to space limitations, we do not include these results.

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. In [9], a space-efficient index structure that decomposes tree patterns into collections of substrings and index them using a trie is used to improve the efficiency of XML data.

Content-based routing. Unlike SemCast, existing approaches for content-based networking do not address the issue of constructing the dissemination overlays. Gryphon [4] assumes that the best paths connecting the brokers are known a priori. Likewise, Siena [5] assumes that an acyclic spanning tree of the brokers is given.

Application-level multicast. Many recent research proposals (e.g., [8, 14]) 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 the same interests. Similar to SemCast, SplitStream [7] 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 [12] 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 [15]. 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 [10]. 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 introduced a semantic multicast system, called *SemCast*, which implements content-based routing and dissemination of data streams. The key idea is to split the data streams based on their contents and spread the pieces across multiple multicast channels, each of which is implemented as an independent dissemination tree. *SemCast* continuously reevaluates its channelization decisions (i.e., the number channels and the contents of each channel) based on current stream rates and profile similarities.

The key advantages of the *SemCast* approach over traditional approaches are that (1) it does not require multi-hop content-based forwarding of streams, and (2) it facilitates fine-grained control over the construction of dissemination trees. Preliminary experimental results show that *SemCast* can significantly improve the bandwidth-efficiency of dissemination, even when the similarity among client profiles is low.

Currently, we are finalizing *SemCast*'s design and pertinent channelization protocols. There are several directions in which we plan to extend our current work. First, we plan to extend our QoS notion to include data quality — the idea is to have multiple channels carrying the same data stream at different “resolutions”. Second, we would like to extend our basic channelization approach to also take into account the network topology and physical locality of the brokers, in order to be able to create better optimized dissemination trees. Finally, we plan to implement a *SemCast* prototype to verify the practicality and efficiency of the approach as well as to demonstrate the claimed processing cost benefits for the interior brokers.

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. Xing, R. Yan, and S. Zdonik. Aurora: A Data Stream Management System. In *Proc. of the International SIGMOD Conf*, 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 *26th VLDB Conference*, 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 *19th ICDCS*, May 1999.
- [5] 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, August 2001.
- [6] A. Carzaniga and A. L. Wolf. Forwarding in a Content-Based Network. In *SIGCOMM*, 2003.
- [7] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. SplitStream: High-bandwidth content distribution in cooperative environments. In *19th ACM Symposium on Operating Systems Principles*, October 2003.
- [8] 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.
- [9] 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.
- [10] 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.
- [11] Y. Diao, M. Franklin, P. Fischer, and R. To. YFilter: Efficient and Scalable Filtering of XML Documents (Demonstration Description). In *ICDE*, 2002.
- [12] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat. Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh. In *19th ACM Symposium on Operating Systems Principles*, October 2003.
- [13] L. Opyrchal, M. Astley, J. Auerbach, G. Banavar, R. Strom, and D. Sturman. Exploiting IP Multicast in Content-Based Publish-Subscribe Systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, April 2000.
- [14] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Application-Level Multicast using Content Addressable Networks. In *3rd International Workshop on Networked Group Communication (NGC '01)*, November 2001.
- [15] A. C. Snoeren, K. Conley, and D. K. Gifford. Mesh-Based Content Routing using XML. In *20th ACM Symposium on Operating Systems Principles*, 2001.
- [16] H. Takahashi and A. Matsuyama. An Approximate Solution for the Steiner Tree Problem in Graphs. *Mathematica Japonica*, 1980.
- [17] W3C. XML Path Language (XPath) 1.0. 1999.
- [18] P. T. Wood. Containment for XPath fragments under DTD constraints. In *9th International Conference of Database Theory*, January 2003.
- [19] E. Zegura, K. Calvert, and S. Bhattacharjee. How to Model an Internetwork. In *INFOCOM96*, 1996.