Scalable Content-Based Routing in Pub/Sub Systems

Anirban Majumder*, Nisheeth Shrivastava *, Rajeev Rastogi ‡, Anand Srinivasan §
* Bell Labs, Bangalore, {*manirban,nisheeths*}*@alcatel-lucent.com*
‡ Yahoo! Labs, Bangalore, *rrastogi@yahoo-inc.com*
§ Google, Bangalore, *anandsr@google.com*

***Abstract*—In this paper, we develop a framework for achieving scalable and communication-efficient dissemination of content in pub/sub systems. To maximize communication sharing across subscriptions, our routing framework groups subscriptions based on similarity, and transmits content matching one or more subscriptions in a group over a single dissemination tree for the group. We develop a cost model that uses published content samples in conjunction with the knowledge of consumer subscriptions to estimate the communication cost of a set of routing trees for subscription groups. The problem of computing a communication-optimal set of routing trees is then formulated as an optimization problem that seeks to find trees with the minimum cost. It turns out that the problem of computing a minimum-cost tree for a subscription group is a new generalization of the well-known Steiner tree problem, and an interesting problem in its own right. We develop an approximation algorithm that uses *low-stretch* spanning trees to compute a tree whose communication cost is within a polylogarithmic factor of the optimum. We use this to compute trees for various subscription-grouping configurations generated using a greedy clustering strategy, and select the one with the lowest cost. Our experimental study demonstrates the effectiveness of our content-aware routing approach compared to traditional routing based on content oblivious spanning trees.**

I. Introduction

The past decade has seen a rapid growth in the amount of data that is being consumed by end-users, primarily due to the widespread popularity of the World Wide Web. As the volume of information available on the web increases, end-users are increasingly looking for convenient and timely ways to access relevant information. This is the basis on which systems based on the *publish/subscribe* paradigm are starting to be deployed.

In a pub/sub system, information *producers* publish content by injecting *documents* into the system, while information *consumers* express their interests to the system via declarative *subscriptions*. The pub/sub system is then responsible for delivering published documents to all interested consumers with matching subscriptions. This enables scalable information dissemination between millions of publishers and subscribers. Information producers need not keep track of the consumers interested in their published content, and can simply push documents into the system. Likewise, information consumers simply receive interesting content without knowing who the producers are. Producers do not attach explicit destination addresses to the documents; rather, delivery of documents to consumers is governed entirely by their content.

RSS feeds, which have gained considerable popularity among users for accessing information on weblogs and other websites, are an example of a pub/sub system, in which users subscribe to a URL and receive the latest updated content as and when it becomes available on that URL. Of course, it is possible to have a much richer syntax for subscriptions instead of just a specific URL; a number of pub/sub system prototypes built within the research community allow a range of subscriptions that span (attribute, value) pairs [3], predicates over attributes with conjunctions and disjunctions [5], [15], *tags* or *keywords* [16], and even general XPath expressions [7], [8], [15].

A. Routing in Pub/Sub Systems

An important challenge in pub/sub systems is the efficient delivery of published documents from information producers to interested consumers. Unlike traditional IP-based routing where the sender explicitly specifies the single destination address for each packet, pub/sub routing may require a document to be transmitted to multiple destinations that are determined based on the contents of the document. Thus, the shortest path routing strategy of IP networks in which each packet is forwarded by a node along the shortest path towards the destination may not be the best solution in a pub/sub system. Note that, in order to efficiently route billions of documents between millions of producers and consumers, the routing scheme should keep both the server load as well as the amount of data transmitted on network links as low as possible.

It is easy to see that any centralized solution in which every document gets routed through a centralized server that determines the appropriate consumers cannot be sufficiently scalable. In such an approach, the central server will quickly become a bottleneck, and hurt overall system scalability. Hence, in this paper, we consider a pub/sub infrastructure consisting of a distributed set of *broker* nodes organized into an overlay network, where each producer/consumer connects directly to a single broker. Given such a system, consider a scheme in which a producer individually transmits each document to interested consumers over separate unicast sessions. Clearly, such a scheme will be highly inefficient since the same document will likely be transmitted over a link multiple times. To improve link bandwidth utilization, broker connections form a *tree* topology and each producer multicasts documents over the tree. This serves to ensure that each document is transmitted at most once over a tree link, and the document transmission to a group of consumers reachable by the link is shared.

Moreover, knowledge of consumer subscriptions can be used to prune the tree links along which each document is forwarded. Specifically, a document is only forwarded over a tree link by a broker node if there are consumers with

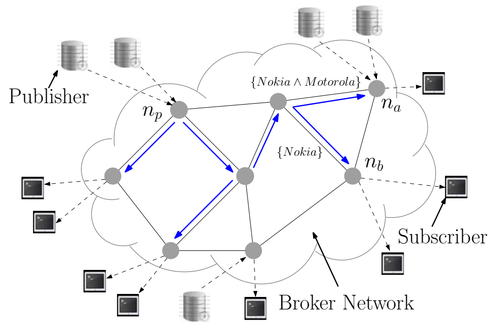


Fig. 1. Pub/Sub System Model. Each node in the Broker Network hosts both publishers and subscribers and also acts as forwarding broker for other nodes.

matching subscriptions reachable by the link. This approach thus exploits data content to reduce communication overhead, and is popularly referred to as content-based routing/filtering in the literature [3], [10], [5]. The effectiveness of content-based filtering is based on the fact that the user interests tend be *localized* to a large extent. For example, many local news and city events are typically of interest mostly to the residents of a certain locality or geographic area. With content-based filtering, the dissemination of these events can be restricted to the few users that are interested in them as opposed to flooding the events to everybody, thus conserving precious bandwidth.

Example I.1. *Figure 1 illustrates our model of the content distribution network, which consists of publishers, subscribers and broker nodes. The system hosts a publisher P at the broker node n_p, and also two end-users who are registered to broker nodes n_a and n_b with subscriptions "Nokia $\wedge$ Motorola" and "Nokia" respectively. Now, any content produced by P which includes the keyword "Nokia" and "Motorola" is inducted into the network by n_p and is subsequently forwarded to nodes n_a and n_b along the directed edges as shown in the figure. On the other hand, content that matches "Nokia" but has no reference to "Motorola" will only be sent to n_b, pruning the last-hop link to n_a.* $\square$

B. Issues with Single Tree Selection

Much of the previous work has focused on developing content-based filtering protocols for minimizing data transmission cost, with the assumption that the underlying multicast tree is first computed completely oblivious of the published documents and user subscriptions. Typically, either a minimum-cost spanning tree for the network graph or a shortest path tree (rooted at one of the nodes) is selected for efficient loop-free dissemination of content. The content-based filtering layer is then implemented on top of this pre-computed content distribution tree.

As we now show, document transmission costs are very sensitive to the choice of the multicast tree used to distribute data. Specifically, approaches in which the dissemination tree is selected to be a spanning or a shortest-path tree without taking into account the distribution of documents and subscriptions can result in significantly larger communication costs compared to the optimal tree. We illustrate this point in the example below, that shows that the minimum-cost Steiner tree, which is computed oblivious of the subscriptions, can result in trees that are sub-optimal. (In the example, we use the number of number links traversed by a document as a measure of its transmission cost.)

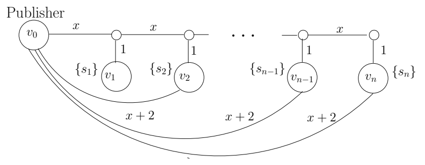


Fig. 2. An example illustrating the bad performance when the minimum cost Steiner tree is used.

Example I.2. *Consider the broker network topology shown in Figure 2 containing $n+1$ nodes and path lengths 1, x, and $x+2$. Each node v_i has a unique subscription s_i that matches a disjoint set of documents. The Steiner tree for this topology connecting publisher v_0 to subscribers v_i will consist of the long horizontal path (that contains sub-paths of length x) and the vertical edges (of length 1). Note that this is independent of the documents that are required at the different nodes. If each of the nodes have non-overlapping document requirements, then the cost of transmitting n documents (one matching each subscription s_i) over the Steiner tree will be $\sum_{i=1}^{n}(i \cdot x + 1)$, which is $O(n^2 \cdot x)$. On the other hand, the shortest path tree results in a cost of $n \cdot (x+2) - 1$ for transmitting the n documents, which is $O(n \cdot x)$.* $\square$

Notice that shortest path tree rooted at the publisher is the optimum tree in the above example. However, we can give similar examples where any of these common tree construction techniques such as shortest path, MST or Steiner tree will have extremely poor performance compared to the optimum. It is also easy to imagine that the situation considerably worsens in the presence of multiple publishers.

C. Issues with Multiple Tree Selection

So far, in our discussion, we have only considered a single dissemination tree to distribute content to subscriber nodes. However, in many cases, there may be sufficient networking resources to support multiple trees to disseminate content. By partitioning the subscriber nodes into multiple clusters and using a separate tree to disseminate content to subscriber nodes in each cluster, we can significantly reduce communication costs.

Intuitively, a single tree reduces communication costs when all subscriber nodes are interested in similar content. In fact, if all nodes are interested in the same content, then a single Steiner tree connecting the publisher and subscriber nodes can be shown to be optimal in terms of communication costs. However, when nodes subscribe to very different content, then having separate trees for nodes with very different subscriptions can help to lower costs. This is because separate trees allow for much richer content distribution topologies - for instance, even though each individual tree is not permitted to have cycles, the union of the various trees may contain cycles. This can lead to lower communication costs as illustrated in the following example.

Example I.3. *Consider the circular broker network topology with unit length edges depicted in Figure 3(a), where v_0 is the publisher and subscriber nodes $v_1, \ldots, v_{\frac{n}{2}}$ have subscription*

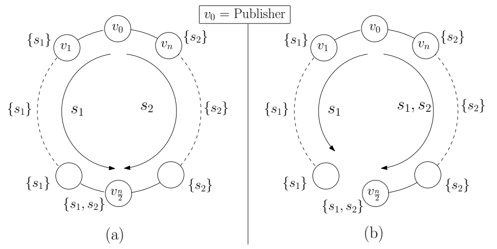


Fig. 3. Example illustrating bad performance of a single tree. The figure shows the network topology along with two trees (a) and the optimal single tree (b).

s_1 and nodes $v_{\frac{n}{2}}, \ldots, v_n$ have subscription s_2. Now let us consider the cost of transmitting two documents (published at v_0), one matching s_1 and the other s_2. Clearly, with the restriction that we have a single dissemination tree, the communication cost is $\frac{3n}{2} - 1$ (see Figure 3(b)).

In contrast, the communication cost with two trees is n. Here one tree for sending the document matching s_1 to subscriber nodes $v_1, \ldots, v_{\frac{n}{2}}$, and another tree is used to send the document matching s_2 to nodes $v_{\frac{n}{2}}, \ldots, v_n$ (see Figure 3(a)). □

Observe that this $\frac{3}{2}$ factor lower communication cost is only possible as a result of permitting cycles in the final topology due to the various trees. In fact, there are instances of graphs for which multiple trees give an $O(\log n)$ factor reduction in communication costs over a single tree (we omit details of this construction due to lack of space).

Several previous works [8], [5] have considered using multiple trees for disseminating data in pub/sub systems. For instance, the XTreeNet system [11] maps documents and subscriptions to a set of *content descriptors* (or subjects) with a separate dissemination tree per content descriptor. In the Semcast system [14], subscriptions are clustered into groups, whose content descriptor is given by the disjunction of subscriptions; any document matching the content descriptor for a group is then broadcast on the corresponding tree. An important distinction of our work from existing approaches is that we use the subscription information to compute the routing tree for each cluster, which leads to a significant reduction in the communication cost. Further, in the above mentioned approaches it is not clear how many content descriptors should be formed or (in XTreeNet) how should they be chosen. We give a more detailed comparison with existing literature in Section II.

D. Our Contributions

In this paper, we present novel schemes that exploit the knowledge of published documents and consumer subscriptions to construct *near-optimal* routing trees that minimize the amount of data transmitted on network links. Our main contributions can be summarized as follows.

- Our routing strategy for pub/sub systems constructs multiple trees for disseminating documents. It groups similar subscriptions and maintains one tree per subscription group. A document is broadcast on a tree if it matches any subscription in the group, and content-based filtering is applied when forwarding the document along tree links. This is a departure from previous approaches [8], [5] that advocate having one tree rooted at each broker node.
- For a single subscription group, the graph-theoretic formulation of the minimum-cost tree computation problem gives rise to a new variant of the well-known Minimum Steiner Tree problem, and is an interesting theoretical problem in its own right. We show that *low-stretch* spanning trees [9] yield a solution whose cost is provably within $O(\log^2 n \log\log n)$ of the optimum (n being the number of nodes in the network). Further, we show a lower bound of $\Omega(\log n)$ for the optimum tree cost to optimum graph (permitting cycles) routing cost ratio - so the costs of our computed trees are fairly close to the costs for optimal graph structures.
- We present a greedy strategy for grouping subscriptions which, in each step, merges a pair of subscription groups that leads to the maximum reduction in the cost of routing trees for the subscription groups. To reduce computation overhead, we also consider an alternate clustering scheme for grouping subscriptions where the similarity between a pair of subscriptions is defined based on the fraction of sampled documents that match both subscriptions.
- Finally, through extensive simulations, we show that our content-aware routing schemes perform well in practice and lead to a substantial reduction in communication overhead compared to existing approaches. In the multiple tree scenario, we are able to achieve a factor of 2 improvement over Semcast [15].

The rest of this paper is organized as follows. Section II discusses related research, followed by a description of the system model for our content distribution architecture in Section III. In Section IV, we present techniques to compute a communication-efficient routing tree for a single subscription group. In Section V, we describe our subscription grouping techniques to compute the best set of routing trees. Section VI details some extensions to handle large numbers of publications and dynamic subscriptions. We then present a detailed experimental analysis of our techniques in Section VII. Finally, we end with some concluding remarks in Section VIII.

II. Related Work

Pub/Sub systems such as Gryphon [1], [3], Siena [6], [5], Corona [16], provide distributed topic-based or content-based data dissemination services. Similar in spirit, Diao *et al.* [8] presented ONYX, which is an XML-based Pub/Sub system over an application-level overlay network. These works lay emphasis on the design of application- level networks with support for content-based filtering. They simply assume that content is disseminated over either a single minimum spanning tree, or multiple per-source shortest-path trees. As we showed earlier in Section I, these could be sub-optimal with respect to communication overhead.

In [14], the authors present the design of the XPORT system, where the structure of the overlay network is optimized based on a variety of cost metrics (*e.g.*, path latency, data rate, *etc.*). A key difference here is that the tree computation heuristics proposed in [14] apply to a single tree whereas we use the subscription information to simultaneously optimize the number and structure of multiple routing trees.

Our work has similarities to the SemCast [15] system which creates multiple channels for disseminating data. A major drawback of the SemCast approach is that it may have trouble scaling to large numbers of subscriptions. This is because during each step of channel computation in SemCast, a pair of channels is merged, and three separate channels containing exclusive traffic are created. Furthermore, in SemCast, a subscription can appear in multiple channel expressions, causing the aggregate size of channel expressions to exceed the total size of subscriptions. Thus the storage requirement at nodes can also be substantially higher in SemCast.

Concepts similar to SemCast's channels are also proposed in [11], [4], [12] to reduce the computation overhead and memory bottleneck at the intermediate broker nodes. However, none of the papers address the problems of how to select channels, or how to map the published content and user subscriptions to channels, so that the overall volume of traffic is minimized. In other related work, [8], [5] maintain dissemination trees rooted at each broker node over which published documents from the broker are delivered to other nodes in the network. But as the number of such brokers can be large (potentially every node in the network), these schemes will not scale in a practical system because they require maintaining too much routing state at each intermediate node (one per tree).

III. System Model

In this section, we describe the architectural details of our content-based pub/sub system, and also formally define the content dissemination problem that we study in this paper.

Broker nodes form the core of the pub/sub system. They are organized as an overlay network, and serve as the intermediaries responsible for efficiently routing content from publishers to subscribers. We will represent the overlay network topology as an undirected graph $G = (V, E)$ with vertices in V corresponding to brokers, and edges in E, the communication links between brokers. We will use n to denote the number of vertices in V; thus, $|V| = n$.

In our pub/sub system (see Figure 1), each publisher and subscriber are individually assigned to broker nodes (called *host* brokers), which are responsible for satisfying their demands. Note that our routing optimizations happen inside the system, to route the document from the host broker of the publisher to that of the subscriber(s). Hence for all practical purposes, we assume that the document (d) gets published at the broker of the publisher ($host(d)$), and consumed at the host broker ($host(s)$) of the subscriber whose subscription (s) matches the document d.

At regular intervals, we collect samples of published documents over the most recent sliding window, and use the samples in conjunction with outstanding subscriptions to compute the best routes.

A. Content-Based Routing Model

Let S denote the set of subscriptions across all the brokers. We cluster subscriptions in S based on their similarity, and for each (disjoint) subscription cluster S_i, we construct a separate routing tree T_i. Any document (originating at any broker node) matching a subscription in S_i is routed on tree T_i. At the broker node where the document is published, the node checks to see if for any tree T_i, the document matches any of the subscriptions in S_i. If so, it transmits the document on all of T_i's links that are incident on it. Note that T_i needs to be a spanning tree since any broker can publish documents. At a minimum, T_i needs to connect publisher nodes with nodes in $\{host(s) : s \in S_i\}$.

Our objective is to compute the subscriptions sets S_i, and a communication-efficient tree T_i for each subscription set S_i. We compute the structure of tree T_i based on the subscriptions S_i of various subscriber nodes as well as the (sampled) statistics of the documents being routed (see section VI for discussion on the sampling process). Further, the intermediate brokers on each routing tree T_i employ content-based filtering [10], [6] to reduce data transmission when forwarding documents.

We must point out here that a document may match multiple subscription groups S_i and so may be transmitted on multiple trees by the originating broker. This may cause the same document to be transmitted multiple times on a given link (for different trees). One way to reduce these redundant transmissions is to cluster similar subscriptions so that the various subscription groups have minimal overlap.

B. Transmission Cost Model

We now formally define document transmission costs for our routing strategy and then re-state our communication-efficient tree computation problem as an optimization problem. First, we introduce some notation.

Let D be a sample of published documents across broker nodes over a sliding window. Thus, D is representative of the recently published documents in the system. Also, let $D_i \subseteq D$ be the set of documents that match a subscription in S_i. Recall that for each document $d_j \in D_i$, $host(d_j)$ is the broker node that publishes the document. Further, let $N_{ij} = host(d_j) \cup \{host(s) : s \in S_i \text{ and } d_j \text{ matches } s\}$. The second term in the expression denotes the set of broker nodes containing subscriptions in S_i that match d_j. Thus, N_{ij} is the set of nodes between which the document d_j gets routed over tree T_i. One of the nodes in N_{ij} generates the document, and the remaining nodes are consumers with matching subscriptions. Now, for the set of nodes N_{ij}, let $T_i^{N_{ij}}$ denote the minimum connected sub-tree of T_i containing all the nodes in N_{ij}. Intuitively, when document d_j is routed on tree T_i, it only traverses the links in $T_i^{N_{ij}}$ that lie on the paths in T_i between nodes in N_{ij}. Finally, $|T_i^{N_{ij}}|$ denotes the number of edges in sub-tree $T_i^{N_{ij}}$.

We are now in a position to define $cost(S_i, T_i)$, which is the communication cost of transmitting documents in D_i that match subscriptions in S_i over its corresponding routing tree T_i.

$$cost(S_i, T_i) = \sum_{d_j \in D_i} |T_i^{N_{ij}}| \tag{1}$$

Note that in Equation (1) above, we assume that transmitting a document over a link incurs a uniform communication cost of 1 unit. However, if communication costs are proportional to document sizes, then we can easily incorporate this in Equation (1) by weighting the term $|T_i^{N_{ij}}|$ with a constant w_j that reflects the size of document d_j. Different routing costs for each link in the network can also be handled similarly.

Finally, for a routing strategy $R = \{(S_1, T_1), \ldots, (S_k, T_k)\}$, we estimate the communication cost as the sum of the costs of routing documents in D over all the k routing trees $T_1, \ldots, T_k$. Thus, $cost(R) = \sum_{(S_i,T_i) \in R} cost(S_i, T_i)$.

C. Problem Formulation

Our routing optimization problem can be formally stated as follows.

Routing Tree Computation Problem: *Given a graph $G = (V, E)$ over broker nodes, a set of subscriptions S, and a document sample D, find a routing strategy $R = \{(S_1, T_1), \ldots, (S_k, T_k)\}$ such that $cost(R)$ is minimized.* □

It is easy to show that this problem is NP-hard; the special case in which D contains a single document is equivalent to connecting the subset N of nodes that include the publisher and all its subscriber nodes with a single tree. This is essentially the well known NP-hard Steiner tree computation problem: *Given a graph $G = (V, E)$ and a set $X \subseteq V$ of vertices, find a tree of G with the fewest edges that spans all the vertices in X.* To the best of our knowledge, our multiple node set variant is a new generalization of the Steiner tree problem and so far, has not been studied in the research literature.

In the following sections, we first describe how to construct a single tree T_i that minimizes the document transmission costs for a single subscription set S_i. We then build on this tree computation solution to compute the multiple subscription sets S_i and the near-optimal trees for distributing the documents matching those subscriptions.

IV. Algorithm for Computing Single Tree

In this section, we tackle the problem of constructing a cost-optimal routing tree T_i for a subset S_i of subscriptions. Thus, we are interested in computing a spanning tree T_i such that $cost(S_i, T_i) = \sum_{d_j \in D_i} |T_i^{N_{ij}}|$ is minimum. In the remainder of this section, we will drop the subscript i for notational convenience.

A. A Poly-Logarithmic Approximation

We now present an algorithm for computing a routing tree whose cost is at most $O(\log^2 n \log\log n)$ times the cost of the optimal tree. In order to achieve this, we use the results of Elkin et al. [9] on low-distortion embeddings of graphs that can be used to solve the *minimum average stretch spanning tree* (MAST) problem [2], [9].

1) Low-stretch Spanning Trees: Let $G = (V, E)$ be a graph with edge weights $w : E \rightarrow \mathcal{R}$, and edge multiplicities $m : E \rightarrow \mathcal{N}$. Given a spanning tree $T = (V, E')$, let $dist_T(u, v)$ denote the sum of the weights of the edges on the unique path between u and v in T. Then, the stretch of an edge $(u, v) \in E$ in T is defined as follows.

$$stretch_T(u, v) = \frac{dist_T(u, v)}{w(u, v)}$$

The *average stretch* of T is simply

$$avg\_stretch(T) = \frac{1}{M} \sum_{(u,v) \in E} m(u, v) \cdot \frac{dist_T(u, v)}{w(u, v)} \quad (2)$$

Algorithm 1 ComputeTree(G, S, D): Computes spanning tree T with near-optimal $cost(S, T)$.

```
1:  for all documents $d_j \in D$ do
2:      $N_j = host(d_j) \cup \{host(s) : s \in S$ and $d_j$ matches $s\}$;
3:      Let $apx\_steiner(d_j)$ be the (approximate) Steiner tree connecting nodes in $N_j$;
4:  end for
5:  Let $G'$ be the union of $apx\_steiner(d_j)$ for $d_j \in D$;
6:  for all edges $e \in G'$ do
7:      Set weight $w(e) = 1$;
8:      Set multiplicity $m(e)$ = number of Steiner trees that $e$ appears in;
9:  end for
10: Compute the low-stretch spanning tree $T$ of $G'$ using the algorithm of Elkin et al. [9];
11: Connect nodes in $G - G'$ to $T$ using shortest paths;
12: return $T$;
```

Here, $M = \sum_{(u,v) \in E} m(u, v)$ represents the total number of edges taking into account the multiplicities. The goal in MAST is to determine the spanning tree T that minimizes $avg\_stretch(T)$. [9] presents a $O(|E| \log n + n \log^2 n)$ algorithm to deterministically compute a tree with average stretch $O(\log^2 n \log\log n)$.

2) An $O(\log^2 n \log\log n)$ Approximation Algorithm: Algorithm 1) shows how to use low-stretch trees for computing a poly-logarithmic approximation algorithm for our problem. We begin by computing approximate Steiner trees for disseminating each document. Each Steiner tree can be approximated in polynomial time to within a factor of 1.55 [17]. Thus, the union of these approximate Steiner trees is a subgraph G' (not necessarily a tree) with routing cost at most twice the optimal. Now, we assign each edge in G' a weight of 1. Further, we know the set of documents that are routed on each edge of G'. We define the multiplicity of an edge as the number of documents routed on that edge[1]. In G', it is possible that two documents are routed in opposite directions on an edge; however, for our problem, it is enough to simply use the total number of documents, while ignoring directions. After computing the *low-stretch tree* T using the algorithm of [9], we augment it with shortest paths to nodes in G not contained in T. This ensures that the final returned tree is a spanning tree of G.

B. Analysis

Runtime complexity. Compute each of the approximate Steiner trees for the sets N_j – requires $O(n^2 \log n)$ time. Hence the total complexity until step 4 is $O(|D| \cdot n^2 \log n)$. In steps 5-9, we go through the edges in each of the Steiner trees once which requires a total of $O(|D| \cdot n)$ time (observe that each tree has at most $n - 1$ edges). Step 10 invokes the low-stretch tree algorithm of Elkin et. al. [9] that runs in $O(|E| \log n + n \log^2 n)$ time. Adding these up and keeping only the dominant terms gives the total time complexity of the ComputeTree procedure as $O(|D| \cdot n^2 \log n)$.

[1]To handle documents with associated weights, the multiplicity of an edge can be defined as the sum of the weights of documents routed on that edge.

Approximation bound. We now show that Algorithm 1 computes a tree with cost at most $O(\log^2 n \log\log n)$ times the optimum.

Theorem IV.1. *Let T be the tree computed by procedure* COMPUTETREE *for a set S of subscriptions, and let T^{opt} be the minimum-cost tree. Then, $cost(S,T) \leq O(\log^2 n \log\log n) \cdot cost(S, T^{opt})$.*

Proof: Let $steiner(d_j)$ be the minimum-cost Steiner tree connecting nodes in N_j. Our approximate Steiner tree algorithm computes a 2-approximation and so we get that $\sum_{d_j} |apx\_steiner(d_j)| \leq 2 \cdot |steiner(d_j)|$. Further, observe that $\sum_{d_j} |steiner(d_j)|$ is a lower bound on the cost of routing documents d_j on any tree. This is because for any routing tree T, $|steiner(d_j)| \leq |T^{N_j}|$. Thus, since T^{opt} is the optimal tree, we get that $cost(S,T^{opt}) \geq \sum_{d_j} |steiner(d_j)| \geq |apx\_steiner(d_j)|/2$. Note that the sum M of the multiplicities of edges in graph G' is equal to the sum of the approximate Steiner tree costs. Thus, $cost(S,T^{opt}) \geq M/2$.

Recall that the cost of routing documents on tree T computed by Algorithm 1 is given by $cost(S,T) = \sum_{d_j \in D} |T^{N_j}|$. We claim that $cost(S,T) \leq \sum_{(u,v)\in E} m(u,v) \cdot dist_T(u,v)$. This is because the term on the LHS corresponds to the routing cost when each document d_j is sent along the edges in tree T^{N_j}. This is the smallest possible set of edges in tree T connecting the nodes in N_j. On the other hand, the term on the RHS corresponds to the routing cost when we *simulate* the routing of each document d_j on T according to its approximate Steiner tree in G'. Specifically, if a document d_j is routed on edge (u,v) in $apx\_steiner(d_j)$, we route it along the unique path between u and v in T. Thus, the edges over which d_j is routed in T form a connected component that connects all the nodes in N_j. Hence, they subsume T^{N_j}.

Now, from [9], we get that $avg\_stretch(T) = O(\log^2 n \log\log n)$. Hence, Equation (2) implies that $\frac{1}{M}\sum_{(u,v)\in E} m(u,v) \cdot dist_T(u,v) \leq O(\log^2 n \log\log n)$. Combining this with $cost(S,T) \leq \sum_{(u,v)\in E} m(u,v) \cdot dist_T(u,v)$ and $M \leq 2 \cdot cost(S,T^{opt})$, we get $cost(S,T) \leq O(\log^2 n \log\log n) \cdot cost(S,T^{opt})$. □

Lower bound. The approximation factor in Theorem IV.1 directly follows from the approximation bound for the stretch of spanning trees; any improvement in that bound can be directly carried over to our approach, giving a better approximation ratio. But, given that the lower bound for this stretch is known to be $\Omega(\log n)$ [2], it will not be possible to extend this technique to get a better than $O(\log n)$ approximation to our routing problem. However, we have also shown that our routing tree computation problem does not admit an approximation ratio better than $O(\log n)$. (We omit the proof of the following lemma due to space constraints.)

Theorem IV.2. *There exists a family of graphs on n vertices such that the cost of the optimal routing tree is $\Omega(\log n)$ times the cost of the union of the per-document Steiner trees.*

V. ALGORITHM FOR COMPUTING MULTIPLE TREES

In the multiple tree approach, we partition the input set S of subscriptions into subsets $S_1, \ldots, S_k$ with corresponding spanning trees $T_1, \ldots, T_k$. Each document d originating at node $host(d)$ is then broadcast on tree T_i if it matches a subscription in S_i. Thus, every document is delivered to all the nodes with matching subscriptions.

Algorithm 2 COMPUTEMULTIPLETREES1(G,S,D): Computes low-cost routing strategy $R = \{(S_1,T_1),\ldots,(S_k,T_k)\}$.

```
1: $Clusters = \{\{s_l\} : s_l \in S\}$;
2: while there exist $S_i, S_j$ in $Clusters$ such that $cost(S_i \cup S_j) <$
   $cost(S_i) + cost(S_j)$ do
3:     Choose pair $S_i, S_j$ with maximum value of
       $\frac{cost(S_i)+cost(S_j)-cost(S_i \cup S_j)}{cost(S_i)+cost(S_j)}$;
       /* Merge clusters $S_i$ and $S_j$ */
4:     $Clusters = Clusters - \{S_i, S_j\} \cup \{S_i \cup S_j\}$;
5: end while
6: return $\{(S_i,T_i) : S_i \in Clusters$ and $T_i =$
   COMPUTETREE$(G, S_i, D_i)\}$;
```

Clearly, to minimize document transmission costs, we should group similar subscriptions together. Here, by similar subscriptions, we mean subscriptions with a significant overlap in their matching documents. Putting similar subscriptions in the same partition ensures that their matching documents are transmitted at most once over each link, thus resulting in reduced costs through increased document sharing. In contrast, if dissimilar subscriptions are included in the same partition, then the (single) spanning tree for the partition needs to be optimized for different document dissemination patterns, and the final tree may be sub-optimal for many of the patterns, thus leading to high dissemination costs.

Thus, our goal is to find a clustering of the subscriptions S into subsets $S_1, \ldots, S_k$ such that similar subscriptions are grouped together, and dissimilar subscriptions are separated out into different partitions. For each cluster S_i, we independently construct the minimum-cost dissemination tree T_i using our approximation procedure COMPUTETREE. Thus, the cost of transmitting documents in D to nodes with matching subscriptions in S_i is given by $cost(S_i) = cost(S_i, T_i)$. The overall content cost dissemination is then the sum of the costs of the individual clusters.

We present two agglomerative clustering style algorithms Algorithm 2 and Algorithm 3 to compute (disjoint) clusters of subscriptions with low communication cost. Initially, the algorithms treat each subscription as a separate cluster. Then in each step, they greedily merge the most similar pair of clusters. However, the two algorithms use different notions of similarity for clustering subscriptions.

In Algorithm 2, the fractional cost reduction when two subscription clusters S_i and S_j are merged together is used as a measure of the similarity between S_i and S_j. More specifically, $sim(S_i,S_j) = \frac{cost(S_i)+cost(S_j)-cost(S_i \cup S_j)}{cost(S_i)+cost(S_j)}$. Observe that computing this similarity function between S_i and S_j can be expensive since this requires computing the tree for $S_i \cup S_j$ using Algorithm 1 which has a worst-case time complexity of $O(|D|n^2 \log n + n \log^2 n)$.

In Algorithm 3, we use a less expensive notion of similarity based on document overlap. If D_i and D_j are the document sets matching subscriptions S_i and S_j, then $sim(S_i,S_j) = \frac{|D_i \cap D_j|}{|D_i \cup D_j|}$. Intuitively, subscriptions with many documents in

Algorithm 3 ComputeMultipleTrees2(G, S, D): Computes low-cost routing strategy $R = \{(S_1, T_1), \ldots, (S_k, T_k)\}$.

```
1:  $CurClusters = \{\{s_l\} : s_l \in S\}$;
2:  $MinCostClusters = CurClusters$;
3:  $MinCost = \sum_{S_i \in MinCostClusters} cost(S_i)$;
4:  while $|CurClusters| > 1$ do
5:      Choose pair $S_i, S_j$ in $CurClusters$ with maximum value of $\frac{|D_i \cap D_j|}{|D_i \cup D_j|}$;
        /* Merge clusters $S_i$ and $S_j$ */
6:      $CurClusters = CurClusters - \{S_i, S_j\} \cup \{S_i \cup S_j\}$;
7:      if $\sum_{S_i \in CurClusters} cost(S_i) < MinCost$ then
8:          $MinCostClusters = CurClusters$;
9:          $MinCost = \sum_{S_i \in MinCostClusters} cost(S_i)$;
10:     end if
11: end while
12: return $\{(S_i, T_i) : S_i \in MinCostClusters$ and $T_i =$ ComputeTree$(G, S_i, D_i)\}$;
```

common are considered to be more similar than subscriptions with few overlapping documents. This is similar to the well-known Jaccard's Similarity Coefficient.

Algorithm 2 terminates if no further cost reductions can be realized from merging clusters, but Algorithm 3 continues merging cluster pairs until there is a single remaining cluster. During the clustering procedure, it keeps track of the minimum-cost cluster configuration, that is, the set of clusters whose dissemination trees have the total minimum cost. Both algorithms return the cluster configuration with the minimum cost along with the corresponding dissemination trees for each subscription cluster.

We remark that a hybrid approach combining Algorithms 2 and 3 is also possible, where we can use Algorithm 3 to bring the number of clusters down to a manageable number, and subsequently, apply the more expensive Algorithm 2 to compute the final set of subscription clusters.

VI. Handling Subscription and Document Distribution Changes

Our routing strategy is optimized to reduce communication costs for a given set of subscriptions and document distribution. However, in practice, both user subscriptions as well as the document distribution can change dynamically. This can cause the current set of dissemination trees to become sub-optimal due to changes in document dissemination patterns.

In order to adapt to document and subscription changes, a centralized node (called the *coordinator*) continuously collects samples of published documents from broker nodes. This is implemented by having each broker node sample documents with a (small) probability p, and then ship the sample stream to the coordinator. To conserve space, the coordinator discards older samples and only retains those samples that fall within the most recent time window (of some pre-specified duration). Further, the coordinator also keeps track of all the subscriptions S in the system – each time a new subscription is added or an existing one deleted from the system, the coordinator is informed of the change. Suppose that the current subscription clusters in the system are $S_1, \ldots, S_k$ (with routing trees $T_1, \ldots, T_k$). Then when a subscription s is deleted, it is deleted from the cluster S_i that contains it, and also from

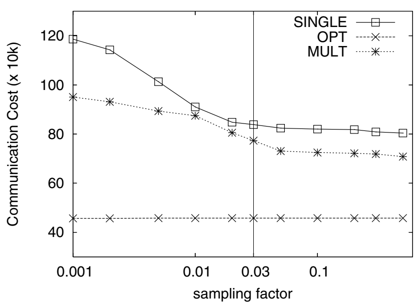


Fig. 4. Effect of sampling on the communication cost.

all the routing tables for T_i. When a new subscription s is added to the system, then the coordinator first selects a cluster to add it to. The selected cluster S_i is the one for which the increase in communication cost is minimum, that is, $cost(S_i \cup \{s\}, T_i) - cost(S_i, T_i)$ is minimum. Subscription s is then inserted into the appropriate routing tables for T_i's links.

Thus, at all times, the coordinator has a document sample D from the most recent sliding window, and it has complete knowledge of all the subscriptions S and (cluster, routing tree) pairs $R = \{(S_i, T_i)\}$. Now, periodically, the coordinator runs our routing tree computation algorithm with inputs as the subscriptions S and the document sample D. Let us denote the newly computed routing strategy as $R' = \{(S'_i, T'_i)\}$. If $\frac{cost(R) - cost(R')}{cost(R)}$ exceeds a certain pre-defined threshold, then the new routing strategy R' is installed and the old routes are torn down.

VII. Experiments

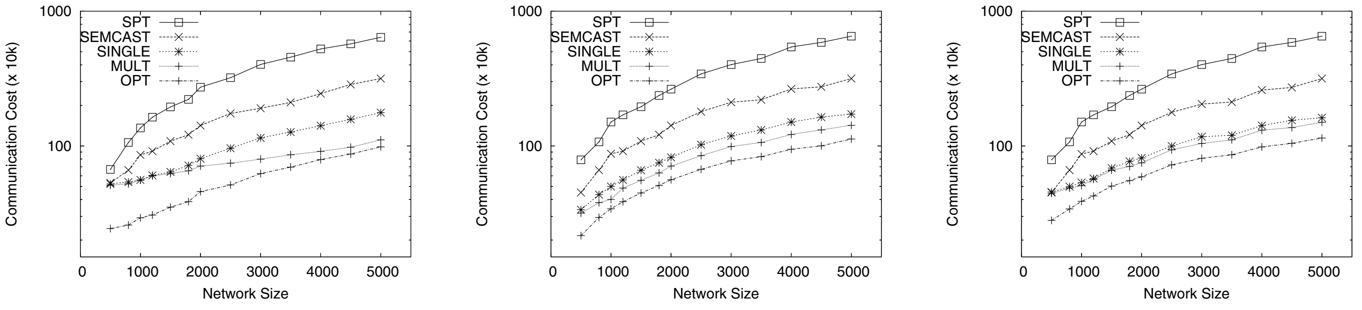
In this section, we evaluate the performance of our routing techniques and compare them with traditional content-based data dissemination approaches through detailed simulations. Due to the lack of real-world data-sets, we only considered synthetic workloads in our experiments. We performed our experiments for various combinations of the pub-sub topology and subscriptions. We thoroughly evaluate the performance of our algorithms compared to the SemCast [15] system through detailed experimentation.

A. Testbed Setup

In our experiments, we used different overlay networks of broker nodes with sizes ranging from 500 to 5000 nodes. The broker networks were generated using the standard GT-ITM tool which has been shown to model the inter-network topologies fairly well [18] and has also been used in earlier research papers related to content distribution networks[15]. For simplicity, we used keyword-based subscriptions, where both documents and subscriptions are modeled as sets of keywords. The total number of keywords was fixed at 1000. We assume a disjunctive semantics for subscriptions, i.e., if any keyword of a document matches a keyword of a subscription, then the document will be delivered to the corresponding subscriber.[2]

The next step in our experimental setup is to assign keywords to the subscriptions and documents. Observe that

[2]The disjunctive form of subscriptions is not a restriction in our setting; in fact, our algorithms and system can easily accommodate the disjunctive normal form with minor changes.



(a) Localized1 (b) Localized2 (c) Uniform

Fig. 5. Communication cost of various schemes.

the proximity of subscribers with similar subscriptions will significantly influence the communication costs and routing structures. To generate a relevant model that takes into account subscription locality, we did the following: for each keyword, we chose a local neighborhood of $x\%$ of the entire network size, by sampling a root node and selecting the closest $x\%$ nodes from the neighborhood. From this set of nodes, we randomly picked up at most 10 nodes and added the keyword to their subscriptions.

Note that by increasing x, we can tune our subscription model from being highly localized to completely uniform. For our experiments, we varied x and generated three different data-sets $Localized1$, $Localized2$, and $Uniform$, corresponding to x= 2, 15, and 100, respectively. Therefore, in $Localized1$, all the subscribers of a keyword are in close proximity of each other whereas in $Uniform$, they are distributed uniformly (randomly) over the entire network. For each document, we sampled a small subset of nearby nodes and randomly selected a subset of keywords from the subscriptions of this set of nodes and assigned it to the document. We also randomly selected a publisher node for the document. Throughout the experiments, we worked with 1000 keywords and a sample of 10000 documents. On an average, each keyword was assigned to around 5 documents and 10 subscribers. For simplicity, all documents were assumed to be of unit size. We used the overall bandwidth consumption for transmitting the documents as the evaluation metric.

We implemented the clustering algorithms MULT (Algorithm 3) and MULT2 (Algorithm 2) and simulated them in our experiments. We also implemented the optimum graph routing algorithm (OPT) that uses separate routing trees for each document. Here, the routing tree for a document is computed as the (approximate) Steiner tree that connects its publisher and subscriber nodes. We used the algorithm presented in [17] to construct the approximate Steiner trees. Further, to compare single and multiple tree routing, we also implemented two techniques that compute a single routing tree for all documents. Specifically, SINGLE computes a single tree based on metric embedding (Algorithm 1) and SPT is a shortest path tree from a randomly selected root node.

Finally, to compare our algorithms with existing techniques, we implemented the SemCast system according to the description presented in [15].

All of our experiments were carried out on an Intel quad Core 2GHz machine with 2GB RAM and running Ubuntu Linux version 7.04. We used the freely available Boost Library [13] for implementing some of the graph algorithms.

B. Experiments on Sampling

We begin by evaluating the performance of our algorithms with varying sampling factors. In this experiment, we varied the sampling factor from 0.001 to 0.5 and ran our algorithms on a network of 2000 nodes based on the statistics collected from the sampled documents. Finally, we computed the routing cost by disseminating all the documents on these trees. Figure 4 shows the result of this experiment. It can be observed that beyond 3%, there is only a marginal reduction in communication cost due to sampling. Thus, by making only 3% of the entire document set available to the coordinator for computing the routing trees, we can significantly reduce the communication overhead. For the remainder of the experiments, we set the sampling factor to 0.03.

C. Experiments on Communication Cost

We next quantitatively assess the performance benefits of our strategies over traditional approaches. In these experiments, we varied the network size from 500 to 5000 and compared the communication costs of each scheme. The costs of the two clustering schemes were quite similar, so we only show here the results of MULT along with other approaches, and present a more detailed comparison of the two schemes in the next section. Figure 5 shows the result of these experiments; note that the y-axis plots the communication cost using a logarithmic scale.

The first observation is that SEMCAST incurs nearly a factor of 2 more communication cost than MULT. The large communication cost is primarily due to the fact the SEMCAST does not perform filtering of content at the internal broker nodes - if a document matches one of the subscriptions of a channel it ends up being forwarded to all the broker nodes belonging to that channel. We observed that the communication cost can be reduced by 30% using internal filtering on the trees computed by SEMCAST.

It can be observed that SINGLE incurs a factor of up to 10 less communication compared to SPT, and this factor increases consistently with network size. Hence in the case of a single routing tree, subscription-aware route computation has a clear advantage over subscription-oblivious approaches. However, observe that, the optimum graph routing cost (OPT) is nearly 60-70% less compared to the single tree (SINGLE). This margin can be substantially reduced if we allow multiple routing trees in the network. Figure 5(a) shows that MULT results in a significant cost improvement over single tree routing and is within 10% of the optimum.

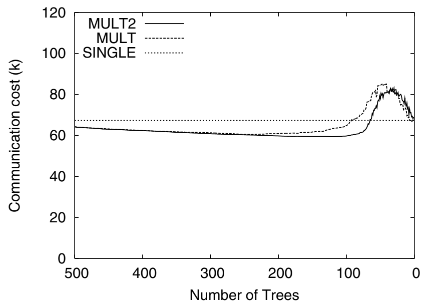


Fig. 6. Variation of cost with number of trees.

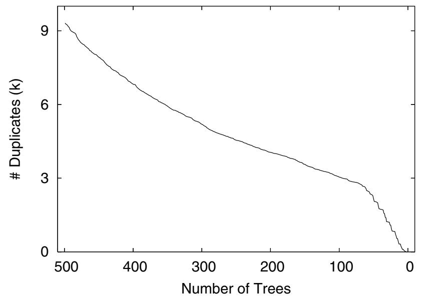


Fig. 7. Variation of number of duplicates with number of trees.

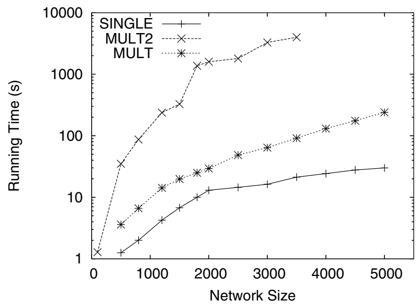


Fig. 8. Comparison of running times for various schemes.

Figure 5 also shows a very interesting trend in cost variation across the different data-sets. Observe that the difference between the costs of single and multiple trees becomes negligible as the data-set becomes more uniform. For Uniform, these costs are almost the same; in fact, the optimal clustering produced by MULT on the Uniform data-set consisted of a single cluster only. The experimental results, in effect, imply that multiple trees offer a significant performance improvement (nearly 50%) over a single tree when the subscription information of the network nodes are localized. However, in case of uniformly distributed subscriptions, single tree routing seems to be sufficient. Note that even in the Uniform data-set, our techniques get roughly the same factor of improvement over SPT as in the localized data-sets.

D. Experiments on Multiple Trees

In this section, we explore some key properties of the multiple tree routing schemes. We present results for a 500 node network over the *Localized1* data-set. Figure 6 shows the variation of the communication cost at each execution step of the clustering algorithms. Observe that in both the clustering techniques, the communication cost starts with a high value and keeps on decreasing for the first few steps. This can be explained as follows: initially each document matches the subscriptions of many clusters and is routed on a large number of trees (one per subscription it matches). This results in many duplicates (i.e. the same document being routed to the same node over multiple trees) and hence accounts for the large communication cost. As the number of trees decreases, the number of duplicates also decreases (see Figure 7), leading to a steady reduction in the communication cost.

However, during the final few steps of clustering, the communication cost was observed to increase. The explanation for this apparent anomaly is as follows: as more and more disparate subscriptions are packed into fewer clusters, the routing within each cluster becomes sub-optimal (see Example I.3). This overhead of sub-optimal routing quickly overshadows the benefit due to a small number of duplicates, and causes the performance to drop to single tree levels. Observe that the optimal clustering with the minimum communication cost for both MULT and MULT2 occurs somewhere between 80 and 100 trees. Note that, this is a significant improvement over some traditional approaches (e.g. [8], [5]) which propose to maintain a separate dissemination tree per broker node. Comparing the two clustering techniques, we observe that MULT2 is marginally better than MULT, performing 3-5% better over all data-sets. Though overall trend remains the same as the network size is varied, we see that MULT2 can produce up to 20% fewer trees.

In our final set of experiments, we plot (in Figure 8) the running times of both clustering algorithms with respect to the network size. Clearly, MULT is several orders of magnitude faster than MULT2 and scales much better with network size.

VIII. Conclusions

In this paper, we have developed algorithms and techniques to achieve scalable and communication-efficient routing of content in a pub/sub system. To minimize communication overhead, our routing algorithm groups subscriptions based on similarity and routes any content matching one or more of these subscriptions in a group over a single dissemination tree (for the group). Towards this end, we designed two novel clustering algorithms for grouping subscriptions that utilize *low-stretch* spanning trees whose cost is within a poly-logarithmic factor of the optimum. Experimental results indicate that our approach achieves significant improvements (by factors ranging from 2 to 10) over traditional methods of content routing.

References

[1] M. Aguilera, R. Strom, et al. Matching events in a content-based subscription system. In *PODC*. 1999.
[2] N. Alon, R. Karp, et al. A graph-theoretic game and its application to the k-server problem. *SIAM J. Comp.*, 24(1):78–100, 1995.
[3] G. Banavar, T. Chandra, et al. An efficient multicast protocol for content-based publish-subscribe systems. In *ICDCS*. 1999.
[4] F. Cao and J. P. Singh. Efficient event routing in content-based publish/subscribe service network. In *INFOCOM*. 2004.
[5] A. Carzaniga, A. Wolf, et al. A routing scheme for content based networking. In *INFOCOM*. 2004.
[6] A. Carzaniga and A. L. Wolf. Forwarding in a content-based network. In *SIGCOMM*. 2003.
[7] R. Chand and P. A. Felber. A scalable protocol for content-based routing in overlay networks. In *NCA*. 2003.
[8] Y. Diao, S. Rizvi, et al. Towards an internet-scale xml dissemination service. In *VLDB*. 2004.
[9] M. Elkin, Y. Emek, et al. Lower-stretch spanning trees. In *STOC*. 2005.
[10] F. Fabret, H. A. Jacobsen, et al. Filtering algorithms and implementation for very fast publish/subscribe systems. In *SIGMOD*. 2001.
[11] W. Fenner, M. Rabinovich, et al. Xtreenet: Scalable overlay networks for xml content dissemination and querying (synopsis). In *WCW*. 2005.
[12] S. Ganguly, S. Bhatnagar, et al. A fast content-based data distribution infrastructure. In *INFOCOM*. 2006.
[13] http://www.boost.org/. Boost c++ source library.
[14] O. Papaemmanouil, Y. Ahmad, et al. Extensible optimization in overlay dissemination trees. In *SIGMOD*. 2006.
[15] O. Papaemmanouil and U. Cetintemel. Semcast: Semantic multicast for content-based data dissemination. In *ICDE*. 2005.
[16] V. Ramasubramanian, R. Peterson, et al. Corona: a high performance publish-subscribe system for the world wide web. In *NSDI*. 2006.
[17] G. Robins and A. Zelikovsky. Improved steiner tree approximation in graphs. In *SODA*. 2000.
[18] E. W. Zegura, K. L. Calvert, et al. How to model an internetwork. In *INFOCOM*. 1996.