A Graph Theoretic Approach to Bounding Delay in Proxy-Assisted, End-System Multicast

Naceur M. Malouch[†] Zhen Liu[*] Dan Rubenstein[‡] Sambit Sahu[*]

[†] Projet Mistral
INRIA
Sophia Antipolis, France
Naceur.Malouch@sophia.inria.fr

[*] IBM T.J. Watson Research Center
Hawthorne, NY
{zhenl,sambits}@us.ibm.com

[‡] Dept. of Electrical Engineering
Columbia University
New York, NY
danr@ee.columbia.edu

***Abstract*— End-system multicast provides a low-cost solution to scalably broadcast information to groups of users. However, last-mile bandwidth limitations constrain tree fanouts leading to high end-to-end delivery delays. These delays can be reduced if the network provides forwarding proxies with high fanout capabilities at an additional cost. We use simple graph theoretic network models to explore the problem of building hybrid proxy/end-system application layer multicast trees that meet fixed end-to-end delay bounds. Our goal is to meet a fixed delay bound while minimizing costs associated with the utilization of proxies. We provide an algorithm and formally prove its optimality in a fully-connected overlay network with uniform-length edges. We then adapt this algorithm into a heuristic and evaluate the heuristic for simulated transit-stub networks with variable-delay edges. We compare our heuristic in a proxy-free environment to previously developed heuristics and show that our heuristic typically yields further reductions in the maximum session end-to-end delay.**

I. Introduction

MULTICAST is a delivery service that can offer tremendous savings in bandwidth for applications that require delivery of the same data to multiple receiver destinations. However, numerous architectural and economic complications have prevented a ubiquitous deployment within the IP network layer [9]. This has led to recent efforts that explore implementing multicast within the IP application layer upon *network overlays*: virtual networks that directly connect application-layer-supporting network components to one another by tunneling transmissions across the underlying, unicast-only network.

Previous work considers two variants of this application layer multicast. One variant, known as *End-System Multicast (ESM)*, forms the overlay out of the very network end-systems that wish to receive the broadcast [8], [7], [6]. These end-systems connect together over unicast channels to form a multicast tree, rooted at the transmission source in which the end-systems are the nodes and the unicast channels are the edges. ESM is a cheap alternative because there is no need to pay the network or service provider for additional multicast support - the users controlling the end-systems provide all additional functionality necessary. However, applicability of ESM to delay-sensitive applications is constrained for two reasons. First, the transmission bandwidth available to end-systems is often limited by the last-mile connectivity technology such as modem, DSL, and cable, limiting the number of copies of a transmission that an end-system can simultaneously forward. Hence, interior nodes of end-system multicast trees tend to have lower fanout resulting in trees of greater depth and delivery delay. Second, transfer delays across this "last mile" are often a significant fraction of a connection's overall end-to-end delay, ranging between 20ms and 150ms depending upon the last mile technology. These large delays further magnify the delay penalties formed from the longer chains of end-systems.

An alternative approach that we will call *Proxy-Based Multicast (PBM)* [14], [1], [13] uses *proxies* as multicast forwarding points. In this paper, a proxy is a high-bandwidth multicast forwarding device that is provided by the network or service provider at a cost. It can be a router or an end-system that can be strategically placed within the network, attached directly to high-speed lines. In comparison to end-system multicast, because proxy transmission quantities are not constrained by last-mile bandwidth limitations, trees formed from proxies are flatter and wider with lower-delay edges.

In this paper, we focus on the needs of distributed applications such as teleconferencing, distributed gaming, chat rooms, and small-scale live concerts or sporting events where the number of receivers is in the tens or hundreds. For these kinds of applications, low-latency delivery is of paramount importance, as is keeping session costs to a minimum. We consider applications designed to cope with delays up to some Δ. Transmission below this delay can be achieved upon a unicast-only network layer by multicasting at the application layer through proxies. However, we assume that each proxy charges per copy of the transmission that it forwards. Hence, our goal is to restrict the number of transmissions that emanate from proxies by using end-systems to perform the multicast wherever possible and still meet the end-to-end delay bound requirements of the application for all session participants.

This paper has two main contributions:

- It provides a more formal evaluation of the use of end-system multicast for delay-constrained multicast applications.
- It is the first work that we are aware of that explores the problem of minimizing proxy costs in hybrid proxy/end-system environments.

This material was supported in part by the National Science Foundation under CAREER Award No. ANI-0133829. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

N. Malouch performed this work as a visitor at IBM Labs, Hawthorne, NY, Summer 2001.

In this paper, we do not produce protocols that are ready for deployment. Indeed, we do not focus on several issues that crop up on practice, such as dealing with joining and leaving participants in the session, and building a distributed version of the protocol. Our goal instead is to

- provide a well-formalized optimization problem.
- understand the complexities involved in solving the desired optimization problem when the constraints are static and known.
- compare our solutions to existing solutions that might be applied in this area.

We develop a simple graphical model to explore algorithms and heuristics in a theoretical context in which nodes represent multicast-capable agents (proxies or end-systems) and edges represent routes between these agents. Any "acceptable" multicast tree built on top of this graph satisfies three requirements: a) it must connect all end-system nodes (but need not connect all proxies) to the source, b) the delay along the path from the source to any end-system must be below the specified delay bound, and c) the number of children of a node must fall beneath the bound imposed by the node's bandwidth constraint. While our focus is on small to medium-sized sessions, we believe our solution could easily be adapted to support larger size applications (thousands of participants) in networks that deploy larger capacity proxies.

We provide an optimal algorithm for the case in which the delays between all pairs of nodes in the graph are identical, but where fanouts from the various nodes can differ due to the variety in access bandwidths available to the nodes. This uniform-delay assumption is appropriate for networks where the most significant delays are due to high transfer delays of identical last-mile technologies, as an approximation for delay in systems where precise node-to-node delays are unavailable, or where all node guarantees only that they will forward a packet to the next hop on the overlay within a fixed time bound, τ. Next, we consider environments in which delays between nodes need not be uniform. Here, the optimization problem is NP-hard. We address the challenge of finding an efficient, low-complexity solution by extending our optimal algorithm to a heuristic that quickly identifies a "good", but not necessarily optimal tree. The heuristic contains a tunable parameter that allows us to adjust the relative importance given to bandwidth constraints of nodes versus delay constraints of edges. We evaluate the performance of the heuristic through simulation on randomly generated transit-stub topologies and consider cases where proxy placement is restricted to within the backbone, restricted within the stub networks, restricted to access points, or is unrestricted. We evaluate our heuristic in two ways. First, we compare the maximum end-system delay of proxy-free trees built by our heuristic to those built by the heuristic developed in [7]. We demonstrate in theory that our heuristic provides a 25% reduction in delay. Next, we demonstrate the expected cost (where cost equals the number of edges extending from proxies) of trees built by the heuristic to achieve tighter delay bounds.

The rest of the paper is organized as follows. Section II discusses related work. In Section III, we formalize our network model. Section IV presents theoretical results pertaining to uniform-distance networks. Section V discusses the development of the heuristic, and Section VI presents our simulation results evaluating the heuristic. We discuss limitations of our work and future directions in Section VII and conclude in Section VIII.

II. Related Work

Studies to date in the area of application layer multicast have been mostly experimental in nature for both end-system multicast [8], [7], [6], [17] and Proxy-Based Multicast [14], [1], [13]. The experimental work that relates most closely to our theoretical work here is the recent work of Chu et al [7] that uses a heuristic called *Bandwidth-Latency* to build the multicast overlay tree. This heuristic, described in more detail in [21], selects paths by choosing those with the greatest available bandwidth (i.e., maximum possible fanout). Edge delays in the overlay are used only when edges are classified as having identical available bandwidths. We will demonstrate that our heuristic captures the behavior of Bandwidth-Latency when our tunable parameter is set to 1, and that delays can be further reduced by using alternate tunings.

Previous work that has investigated the building of delay-bounded multicast trees has often been in the context of network layer multicast where node fanouts are restricted only by their degree within the underlying graph topology and where inclusion of router nodes in the tree is optional. There, the general optimization problems are variants of NP-complete Steiner-Tree problems and are also NP-complete. Several works have developed heuristic approximations [15], [16], [24], [19], [18] or ratio-bounded approximations [5] to these optimization problems. However, these heuristics are inapplicable to the problems addressed in this paper since there is no straightforward way to account for degree constraints of nodes in a tree that are less its degree in the underlying connectivity graph.

Bounded-fanout multicast has been explored in the context of building (non-source-specific) minimum spanning trees that minimize aggregate edge costs instead of minimizing the delay from a specific source [3], [2]. Computation of delay-bounded paths that minimize a monotone or additive metric, neither of which covers the case of fanout constraints is described in [12]. [20] presents a survey of previous work in the area of QoS multicast routing. There is no discussion in the survey of work that addresses the problem we consider here.

III. Architecture and Model

In this section, we present a more formal definition of our network model and formally pose the optimization problem. However, we first begin by giving a high level description of the problem setting. We consider a set of end-system nodes in which one node in particular wishes to multicast information to the other participating nodes. This is accomplished by constructing a *multicast overlay* where end-system nodes forward the transmission to (several) other participating end-system nodes via unicast connections. These nodes (including the source) have limited bandwidth capabilities, such that they

must form trees in which each node is only required to forward data to a small set of other nodes.

The sessions that we consider here require that the data emanating from the sender must reach all session participants within some fixed amount of time. For cases where transmission delays between end-systems are difficult to predict accurately, or where the delay results from transmission over last-mile technologies, it is desirable to bound the number of hops in the tree that must be traversed to deliver data to an end-systems.

To assist these sessions toward meeting delay bounds, the network provides *forwarding proxies*. These are nodes with access to high bandwidth levels. The multicast session can draw upon these proxies to forward data within the delay constraints to a much larger set of end-systems. However, the network charges a price for each proxies' services that is an increasing function of the number of copies that the proxy is asked to forward.

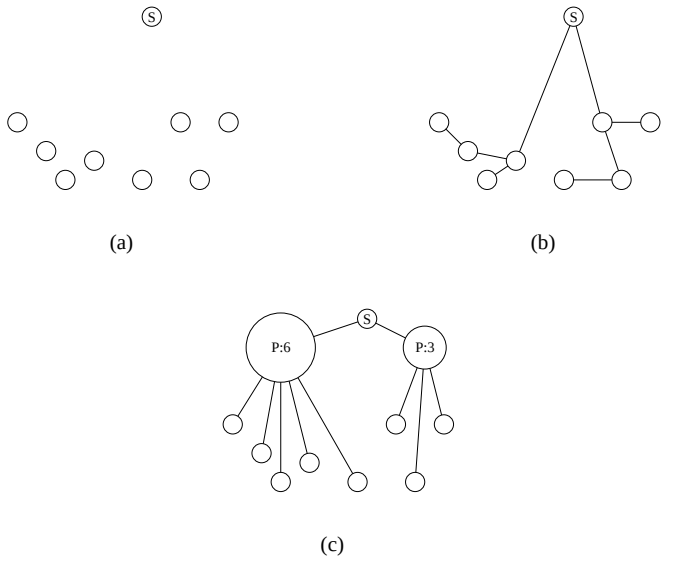


Fig. 1. Reducing Delay with Forwarding Proxies

For example, Figure 1(a) depicts a set of end-system nodes with a particular node labeled S as the data source. S wishes transmit data with low delay to a set of end-system nodes, but the bandwidth limitations imposed on each node (including S) restricts it to simultaneously forwarding at most two copies of the transmission. Hence, a minimal-depth multicast overlay tree in this example has depth 3: the tree depicted in Figure 1(b) is an example of one such tree. Figure 1(c) depicts the same set of end-system nodes in a network where two high bandwidth proxies are also made available. The one on the left can simultaneously forward data to 6 end-systems, and the one on the right can simultaneously forward data to 3 end-systems. By utilizing the proxies to respectively forward data directly to 5 and 3 end-system participants, it is possible to build a multicast overlay tree with depth 2. There is no tree that can be built with depth 2 in which the number of transmissions that emanate from the proxies is smaller. Hence, the tree depicted here is a minimum cost tree when a restriction is applied that trees must have depth no more than two.

We now state our model of the network in a more formal manner. Let $G = (s, N, P, E)$ be a network consisting of a source node s, a set of end-system nodes N, a (possibly empty) set of proxy nodes P, and a set of edges E such that an edge $e = \langle n_1, n_2 \rangle \in E$ exists between each pair of nodes $n_1, n_2 \in \{s\} \cup N \cup P$ (i.e., the overlay network is fully connected). Because overlay networks communicate across tunnels implemented at the transport layer, it is possible for an overlay node to communicate directly with any other overlay node in the network. Let $d(\langle n_1, n_2 \rangle)$ represent the end-to-end delay from n_1 to n_2. In addition, our assumption that bandwidth rates are constrained by the last-mile hop translates to the delays along the edges and the bandwidth availabilities between pairs of nodes being independent of the respective delays and bandwidth availabilities at other nodes. The fact that the actual paths represented by these network edges share links in common is of no consequence since these links do not impose bandwidth constraints in our model.

Definition 1—Fanout Constraints: A **fanout constraint function (FCF)**, $f()$, is a function that maps each node $n \in \{s\} \cup N \cup P$ to a non-negative integer. $f(n) = i$ implies that node n has sufficient bandwidth capabilities to forward session data to at most i other nodes.

We assume that given the transmission rate of the supported session, a node n can determine $f(n)$. For instance, if a node n's bandwidth capability is r and a session is to be transmitted at rate ρ, then $f(n) = \max\{\lfloor (r - \rho)/\rho \rfloor, 0\}$.[1] We assume proxies have access to larger amounts of bandwidth than end-systems, such that $f(p) > f(n)$ for most proxies p and end-systems n. In this paper, we will also use FCFs to artificially limit potential fanout from proxies, i.e., we will construct FCFs $f_1()$ where $f_1(n) = f(n)$ for $n \in \{s\} \cup N$ and $f_1(p) \leq f(p)$ for $p \in P$.

Upon G, we wish to build a tree $T = \{s, N, P_T, E_T\}$ formed from nodes $\{s\} \cup N \cup P_T$ where $\emptyset \subseteq P_T \subseteq P$ and $E_T \subset E$ are the edges. We define several functions that describe several relevant tree properties:

- $\mathbf{\Pi_n(T)}$: the parent of node n in tree T.
- $\mathbf{c_n(T)}$: the number of child nodes of n in tree T, i.e., $c_n(T) = |\{n' : \Pi_{n'}(T) = n\}|$.
- $\mathbf{D_n(T)}$: the delay from s to node n in tree T where $D_s(T) = 0$ and $D_n(T) \equiv D_{\Pi_n(T)}(T) + d(\langle \Pi_n(T), n \rangle)$.
- $\mathbf{D_{max}(T, S)} = \max_{n \in S} D_n(T)$, i.e., the maximum depth in T of any nodes in $S \subseteq T$. For conciseness, we define $D_{\max}(T) \equiv D_{\max}(T, T)$.

Definition 2—Legal Connectivity: We say a tree T is **legally connected with respect to FCF f()** if s is the root of T, $N \subseteq T$, and $c_n(T) \leq f(n)$ for all $n \in T$, i.e., all nodes of N belong to T and no node's maximum degree constraint under $f()$ is violated.

Tree Cost: Let $A(T)$ be the cost for the session to utilize tree T. In this paper, we restrict our attention to cost functions of the form $A(T) = \sum_{n \in P} g(c_n(T))$, where g is an non-

[1] Here, we subtract ρ from the numerator to allow sufficient bandwidth for the node to receive session data.

decreasing, concave or linear function. Under such a cost function, the increase in cost to add an additional transmission from a proxy does not increase. This covers cost functions of the form $g(i) = 0$ for $i = 0$ and $g(i) = \mathcal{C}_1 + \mathcal{C}_2 i$ otherwise, i.e., the proxy provider charges $\mathcal{C}_1$ for each proxy provided to the session plus $\mathcal{C}_2$ for each copy of the session transmitted from that proxy.

Optimization Problem: The formal description of our optimization problem is as follows: Let Δ be an application delay bound. Let $f()$ be the FCF imposed by the application on nodes $\{s\} \cup N \cup P \in G$ and let $A()$ be the proxy cost function. We wish to find a tree, $T_{\min} = (s, N, P_{T_{\min}}, E_{T_{\min}})$ where

- $T_{\min}$ is legally connected w.r.t. $f()$.
- $D_{\max}(T_{\min}, N) \leq \Delta$.
- $A(T_{\min}) \leq A(T)$ for all other trees T that are legally connected w.r.t. $f()$ and where $D_{\max}(T, N) \leq \Delta$.

IV. Fully Connected, Uniform Edge Overlays

In this section, we present an algorithm (called FindMinCost) for use in a network where the end-to-end delay between all pairs of nodes is uniform (WLOG we assume $\forall n_1, n_2 \in \{s\} \cup N \cup P, d(\langle n_1, n_2 \rangle) = 1$). We prove that, given a maximum delay, Δ, the algorithm finds a multicast tree that minimizes the cost to connect the source to all session receivers along paths with a delay less than Δ. This algorithm is developed in three steps. In the first step, we present an algorithm (called MinDepth) that, for a given FCF, $f()$, computes a minimum-depth tree rooted at s. This initial algorithm does not distinguish between proxy nodes and receiver nodes, and hence the tree does not necessarily have minimal cost. Next, we apply the theory of majorization to construct an algorithm (called FindBestProxyTree) that computes a minimum-depth tree that does not exceed a fixed cost, C. FindBestProxyTree is implemented by selecting an appropriate FCF and then applying MinDepth. Last, we construct FindMinCost by choosing various values for C until a minimum cost for which a tree exists with maximum depth no larger than Δ is found.

A. Minimizing Fanout-constrained Tree Depth

We begin by presenting Algorithm MinDepth which computes a minimum depth tree w.r.t. FCF $f()$ on G. The algorithm essentially puts nodes with highest potential fanout (i.e., largest $f(n)$) closer to the source:

Algorithm 1: MinDepth(G, $f()$)

1) Let the source node be n_0 and order the nodes in G as $n_1, n_2, \cdots, n_{|N|-1}$ such that $f(n_i) \geq f(n_j)$ for all $1 \leq i < j < |N|$.
2) $i = 1, j = 0$
3) While $i < |N|$
4) Set $\Pi_{n_i}(T_{\min}) = n_j$
5) $i++$
6) If $c_{n_j}(T_{\min}) = f(n_j)$ then $j++$
7) end loop
8) return $T_{\min}$

Lemma 1: Algorithm MinDepth generates a minimum-depth legally connected tree w.r.t. $f()$.

Proof: Let L_T^i be the set of nodes within tree T that reside at depth less than or equal to i. We begin by proving the following claim:

Claim 1: $|L_T^i| \leq 1 + \sum_{n \in L_T^{i-1}} f(n) - |L_T^{i-1}| = 1 + \sum_{n \in L_T^{i-1}} (f(n) - 1)$.

Proof: Every node in the subtree formed from the nodes in L_T^{i-1} has one parent with the exception of the root, which has no parents. The sum of fanouts of nodes in L_T^{i-1} is $\sum_{n \in L_T^{i-1}} f(n)$, and with $|L_T^{i-1}| - 1$ edges "used" to connect a node to its parent , there remain at most $1 + \sum_{n \in L_T^{i-1}} f(n) - |L_T^{i-1}|$ edges that can connect nodes of depth i to nodes of depth $i-1$ in T. ■

We next show by induction on i that for any legally connected tree T w.r.t. $f()$, $|L_{T_{\min}}^i| \geq |L_T^i|$ for all $i < D_{\max}(T_{\min})$. For $i = 0$, the result clearly holds, since $|L_{T_{\min}}^0| = |L_T^0| = 1$, i.e., just the root node. Assume that $|L_{T_{\min}}^{i-1}| \geq |L_T^{i-1}|$, and let $S(j)$ be the set of j nodes first attached to the tree by MinDepth. Since MinDepth places nodes of largest fanout at the mininum depths of the tree, it follows that $\sum_{n \in S(j)} f(n) \geq \sum_{n \in S'} f(n)$ for any other set of nodes S' where $|S'| = |S(j)| = j$. By the construction of $T_{\min}$ using Algorithm MinDepth, no nodes in $L_{T_{\min}}^{i-1}$ are leaves, and it must be the case that $f(n) - 1 \geq 0$ for all $n \in L_{T_{\min}}^{i-1}$. Choosing $j = |L_T^{i-1}|$, since $|L_{T_{\min}}^{i-1}| \geq |L_T^{i-1}|$, we have $S(j) \subseteq L_{T_{\min}}^{i-1}$ and

$$\begin{aligned}
&\sum_{n \in L_{T_{\min}}^{i-1}} (f(n) - 1) \\
&= \sum_{n \in S(j)} (f(n) - 1) + \sum_{n \in L_{T_{\min}}^{i-1} \setminus S(j)} (f(n) - 1) \\
&\geq \sum_{n \in S(j)} (f(n) - 1) \geq \sum_{n \in L_T^{i-1}} (f(n) - 1). \qquad (1)
\end{aligned}$$

Furthermore, since MinDepth does not add nodes at depth i until $c_n(T_{\min}) = f(n)$ for all nodes n where $D_n(T_{\min}) < i-1$, we have that

$$\begin{aligned}
|L_{T_{\min}}^i| &= 1 + \sum_{n \in L_{T_{\min}}^{i-1}} f(n) - |L_{T_{\min}}^{i-1}| \\
&= 1 + \sum_{n \in L_{T_{\min}}^{i-1}} (f(n) - 1) \\
&\geq 1 + \sum_{n \in L_T^{i-1}} (f(n) - 1) \qquad (2) \\
&\geq |L_T^i|, \qquad (3)
\end{aligned}$$

where (2) holds due to (1) and (3) holds due to Claim 1, completing the proof by induction (for all $i < D_{\max}(T_{\min})$). The final result follows from the fact that for any $i < D_{\max}(T_{\min})$, $|L_T^i| \leq |L_{T_{\min}}^i| \leq |N|$. Thus, not all nodes of T can lie at a depth less that $D_{\max}(T_{\min})$. ■

B. Applying Majorization to FCFs

We next develop an algorithm that inserts proxy nodes within a tree that minimizes tree depth while keeping costs below a

given bound, C. Before introducing the algorithm, we introduce a majorization technique and prove an important property of majorization that is critical in demonstrating the correctness of our algorithm.

Definition 3: Let $f_1()$ and $f_2()$ be two different FCFs in G. We say that $f_1()$ **majorizes** $f_2()$ and write $f_1() \succ f_2()$ if there exist two orderings of the nodes in G, $\alpha_1, \cdots, \alpha_m$ and $\beta_1, \cdots, \beta_m$ such that all the following hold:

- $f_1(\alpha_i) \geq f_1(\alpha_j)$ and $f_2(\beta_i) \geq f_2(\beta_j)$ for $i < j$.
- For all $j > 0, \sum_{i=1}^{j} f_1(\alpha_i) \geq \sum_{i=1}^{j} f_2(\beta_i)$.
- $\sum_{i=1}^{m} f_1(\alpha_i) = \sum_{i=1}^{m} f_2(\beta_i)$.

Lemma 2: Let $f_1()$ and $f_2()$ be two different maximum fanout assignments in which $f_1() \succ f_2()$. If T_0 is a legally connected tree in G with respect to $f_2()$, then there exists a tree T_1 that is legally connected with respect to $f_1()$ where $D_{\max}(T_1) \leq D_{\max}(T_0)$.

Proof: Let $\alpha_1, \cdots, \alpha_m$ be an ordering of the nodes in G in which $f_1(\alpha_{i+1}) \leq f_1(\alpha_i)$ and $\beta_1, \cdots, \beta_m$ be an alternate ordering in which $f_2(\beta_{i+1}) \leq f_2(\beta_i)$. We construct T_1 as follows: first, create a tree T_2 that is legally connected w.r.t. $f_2()$ using Algorithm 1. By Lemma 1, $D_{\max}(T_2) \leq D_{\max}(T_0)$. Next, form a tree T_3 in which node α_i connects to node α_j iff β_i connects to β_j in tree T_2. Since the trees have isomorphic connectivity structure, $D_{\max}(T_3) = D_{\max}(T_2)$. Some nodes in T_3 may violate fanout constraints of $f_1()$ such that T_3 is not legally connected w.r.t. $f_1()$. We convert T_3 into a legally connected tree w.r.t. $f_1()$, T_1 by detaching subtrees rooted at nodes n where $c_n(T_3) > f_1(n)$ and re-attaching them to nodes n' where $D_{n'}(T_3) \leq D_n(T_3)$ and $c_{n'}(T_3) < f_1(n')$. The fact that a sufficient number of available edges exist at lower depths of the tree follows from the condition that $f_1() \succ f_2()$ and hence for all $j > 0, \sum_{i=1}^{j} f_1(\alpha_i) \geq \sum_{i=1}^{j} f_2(\beta_i)$ and that in trees constructed by Algorithm MinDepth, the depth of nodes of larger fanout is no more than those of lesser depth. The transition from T_1 from T_3 involved only the shifting of subtrees closer to the source. Hence, $D_{\max}(T_1) \leq D_{\max}(T_3) = D_{\max}(T_2) \leq D_{\max}(T_0)$. ■

C. Minimizing Proxy Involvement

We now design an algorithm that calls Algorithm MinDepth to generate the minimum-depth tree with cost no more than a given C. We do this by restricting the number of children that proxies are permitted beyond the restrictions imposed by their fanout constraints by varying the FCF used when computing the minimum-depth tree. Lemma 2 uniquely determines the FCF to use within the algorithm:

Algorithm 2: FindBestProxyTree(G, C, $f()$)

1) Order proxy nodes $a_1, \cdots, a_m$ such that $f(a_i) \geq f(a_{i+1})$.
2) Construct $f_1()$ such that $f_1(n) = f(n)$ for all $n \in \{s\} \cup N$ and recursively compute $f_1(a_i) = \max_{0 \leq k \leq f(a_i)} \{k : \sum_{j=1}^{i-1} g(f_1(a_j)) + g(k) \leq C\}$.
3) return MinDepth(G, $f_1()$)

Lemma 3: Let T_0 be the tree generated by Algorithm 2. Then $A(T_0) \leq C$. Furthermore, if a tree T exists in which $A(T) = C$ and $D_{\max}(T) \leq d$, then $D_{\max}(T_0) \leq d$.[2]

Here, we present a proof for the case where $g(i) = i$, i.e., session cost for a tree T equals sum of the number of children of proxies.

Proof: The fact that $A(T_0) \leq C$ follows from the construction of the FCF, $f_1()$. Note that $f_1()$ is constructed such that $\sum_{a \in A} f_1(a) = C$. Furthermore, $f_1()$ is also constructed such that any other $f_2()$ where $\sum_{a \in A} f_2(a) = C$ satisfies $f_1() \succ f_2()$. Let T be some tree where $A(T) = C$, and let the fanout of nodes in T be described by fanout function $f_2()$ (i.e., $\forall n \in \{s\} \cup N \cup P, f_2(n) = c_n(T)$). Since $f_1() \succ f_2()$, by Lemma 2, $D_{\max}(T_0) \leq D_{\max}(T)$. Thus, if $D_{\max}(T) \leq d$, then $D_{\max}(T_0) \leq d$.

■

The final algorithm developed in this section calls FindBestProxyTree with different values of C until the smallest C is identified within which a tree of depth less than Δ can be constructed.

Algorithm 3: FindMinCost(G, $f()$, Δ)

1) Set $C_{\min} = 0, C_{\max} = \sum_a f(a)$
2) While $C_{\max} - C_{\min} > 1$
3) $C = C_{\min} + \lfloor (C_{\max} - C_{\min})/2 \rfloor$
4) T = FindBestProxyTree(G, C)
5) if $D_{\max}(T) > \Delta$
6) $C_{\min} = C$
7) else $C_{\max} = C$
8) end if
9) end loop
10) return FindBestProxyTree(G, $C_{\max}$)

Lemma 4: For any pair of FCFs, $f_1(), f_2()$, if for all $n \in G, f_1(n) \leq f_2(n)$, then the depth of the minimum-depth tree that is legally connected w.r.t. $f_2()$ is less than the depth of the minimum-depth tree that is legally connected w.r.t. $f_1()$.

Proof: Any tree that is legally connected w.r.t. $f_1()$ is also legally connected w.r.t. $f_2()$. ■

Theorem 1: Algorithm FindMinCost() finds the minimum cost legally connected tree T w.r.t. FCF $f()$ where $D_{\max}(T) \leq dbound$ and does so in time $O(nlog^2 n)$, where $n = |\{s\} \cup N \cup P|$.

Proof: Correctness of the algorithm follows trivially from Lemmas 3 and 4, plus the observation that $C_{\min}$ is never set to a cost for which a legally connected tree w.r.t. $f_1()$ exists that has depth of Δ or less, and that $C_{\max}$ is only set to costs for which such a tree does exist. Algorithm MinDepth runs in time $O(n \log n)$ to sort the nodes as a function of FCF $f()$. Building the tree takes $O(n)$ time. FindBestProxyTree takes $O(n)$ time to generate $f_1()$ prior to calling MinDepth. Finally, each iteration in Algorithm FindMinCost halves the distance between $C_{\min}$ and $C_{\max}$. Noting that proxy costs cannot exceed $|P| \cdot \max_{p \in P} f(p) \cdot g(1)$ which is $O(n)$, there can be at most $O(\log n)$ iterations. ■

[2]Note that it is possible that $A(T_0) < C$ if not all outgoing proxy edges permitted under $f_1()$ are used, and hence, we cannot claim that a tree of cost C exists.

V. Heuristics for Non-Uniform-Edge Graphs

In this section, we focus on the development of a heuristic that seeks to minimize the cost of a delay-bounded, fanout-constrained multicast with maximum delay Δ in a network where end-to-end delays between nodes can vary. It can be shown that solving this problem exactly is NP-hard via a reduction to Hamiltonian path. Details of this reduction are provided in AppendixA.

A. Design from Theoretical Observations

We draw two observations from our theoretical results in the previous section:

- It is beneficial to have nodes with high fanout closer to the root of the tree.
- One approach to finding the minimum cost delay-bounded tree is to construct an artificial bound on cost by limiting aggregate fanout permitted over all proxies, and then varying this bound to identify a minimum cost tree that meets the delay bound.

Obviously, our theoretical results do not account for the variation in end-to-end delays between nodes. Clearly, it is beneficial to keep delays of hops near the root of the tree small since edges near the root affect a larger fraction of receivers than do their descendants. A dilemma arises when there are a class of nodes with high permitted fanout connected to edges with high delay and another class of nodes with small permitted fanout connected to edges with small delay. Should we move nodes with high fanout nearer to the root, or edges with small delay? Our approach is to develop a heuristic with a tunable parameter, α that can be varied between 0 and 1 whose value determines the relative importance of edge delays and permitted node fanouts when making this selection.

We now describe the process used by heuristic FindTree(Δ, C) in its attempt to build a tree with maximum delay bound less than Δ and maximum cost less than C. During the running of the heuristic, a proxy budget B is maintained that limits the set of edges that can be used from proxies. Initially, this budget is set to C, and every time a new edge is used in the tree extending from a proxy, the cost of adding that edge is deducted from B.

During its execution, the heuristic also maintains three sets of nodes:

- S_a is the set of nodes already attached to the tree and able to accept more children. Initially, $S_a = \{s\}$.
- S_o is the set of nodes to be attached. Initially, $S_o = N \cup P$.
- S_f is a set of nodes that have been attached to the tree but can no longer accept additional children because of fanout restrictions. Initially, $S_f = \emptyset$.

Throughout the duration of the running of the heuristic, S_a, S_o, S_f remain mutually exclusive sets with $S_a \cup S_o \cup S_f = \{s\} \cup N \cup P$. While $S_o \cap N \neq \emptyset$, the heuristic repeats the following procedure. For each node $n \in S_o$, the minimum distance, $\delta(s,n)$ from s to n along a path of nodes whose hop prior to n is some $n_1 \in S_a$ is computed. This ensures that were n to attach to n_1, this would not violate the fanout constraint of n_1. A set S is formed of the nodes $n \in S_o$ where all of the following hold:

- $\delta(s,n) \leq \Delta$
- Adding n does not incur costs that violate the remaining proxy budget, B (this is of concern when $n \in P$).

If $S = \emptyset$, then the heuristic returns a null tree (indicating failure to find an acceptable tree). Otherwise, the node $n \in S$ is selected that minimizes

$$\mathcal{H}_\alpha(n) \quad = \quad \alpha \frac{f(n)}{f_{\max}} + (1-\alpha)\frac{\delta_{\min}}{\delta(s,n)}. \tag{4}$$

where $f_{\max}$ and $\delta_{\min}$ are normalization constants, such that for all n, $\frac{f(n)}{f_{\max}}$ and $\frac{\delta_{\min}}{\delta(s,n)}$ lie between 0 and 1. These constants are calculated as

- $f_{\max} = \max_{n \in \{s\} \cup N \cup P} f(n)$.
- $\delta_{\min} = \min_{n \in \{s\} \cup N \cup P} d(s,n)$

where $d(s,n)$ is simply the shortest path from s to n (ignoring budgets and fanout constraints, e.g., just applying Dijkstra's algorithm).

The node n is then attached to the tree through n_1, and is moved from S_o into S_a. Nodes n and n_1 are then moved from S_a to S_f if their respective numbers of children equal their respective fanout constraints.

B. Additional Modifications

Evaluation with this preliminary heuristic revealed two problems:

1) The proxy budget, B would on occasion be depleted by attaching unneeded chains of high fanout proxy nodes with low delay edges that never had any end-system descendents. These proxy nodes would serve no use in the session since they would not lie on the any of the end-systems' transmission paths, but would still prevent the heuristic from subsequently adding additional proxies that might in fact serve some use.
2) Proxies would often aggressively be attached near the source, depleting available edges of nodes near the source. This would sometimes push end-systems away from the source, unnecessarily increasing transmission delays.

These problems were rectified in a second version of the heuristic, FindTree2(Δ, C), that was identical to the first heuristic except that a 2-phase process to attach proxies was used. In this 2-phase process, nodes are attached as in the previous version of the heuristic, but an attached proxy, $p \in P$ would only be counted against proxy budget or against the fanout constraint of an ancestor node in the tree when some descendent of p (attached later on by the heuristic) is an end-system. An example of this 2-phase process is depicted in Figure 2. The node marked with an S is the source, nodes marked with E are end-systems and nodes marked with P are proxies. The number of "counted" children of a node is indicated to its immediate left. In Figure 2(a), proxies have been added but are not ancestors to any end-systems and hence do not "count" as descendents. In Figure 2(b), and end-system is added, incrementing each node's child count up by 1 all the way up the chain to the source. In Figure 2(c), an additional end-system node is attached to the same node as before, increasing its parent node's child count. However, that node had "counted" previously, its parent does

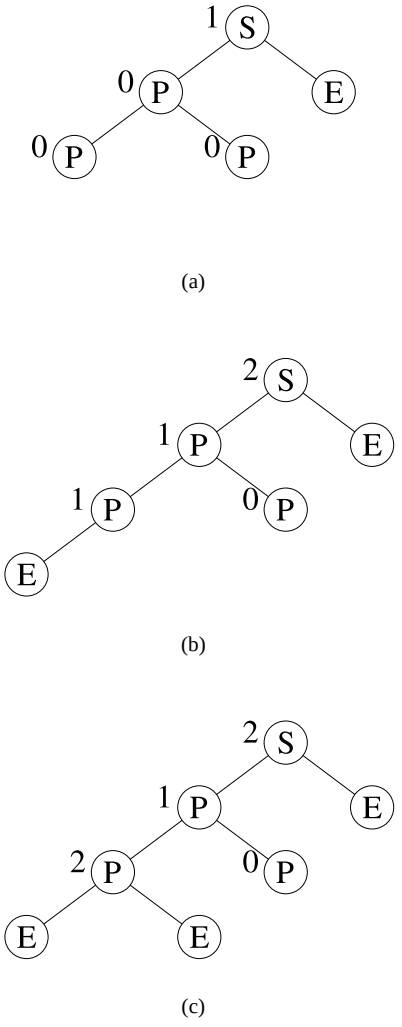


Fig. 2. Sample 2-phase proxy attachment

not increment its own child count. At the completion of the running of the heuristic, children that are not "counted" do not have any end-system descendents, and hence can be dropped from the tree.

When $\alpha = 1$, the heuristic gives priority to nodes with larger fanout. Here, the heuristic most closely resembles the algorithm from Section IV where edge delays are used only to break ties. Also, the heuristic is identical to the heuristic used within [7]. When $\alpha = 0$, the heuristic gives priority to nodes with minimal delay from the source. Values of α between 0 and 1 give priority to nodes with both high fanout and low delay, with the emphasis placed on fanout increasing with increasing α.

VI. Heuristic Evaluation

In this section, we evaluate the heuristic in transit-stub networks [23] created using the gt-itm software package [4]. While debate continues about the accuracy of topologies generated to emulate large-scale networks [10], our understanding is that the package provides an accurate model of the medium-sized network topologies we experiment upon within this paper.

A. Experimental Setup

1) Underlying Network Topology: For all experiments, we generate two instances of the underlying network layer topology. Both instances contain 30 nodes within the transit domain, 10 of which are randomly selected to connect to 10 separate stub domains. Each stub domain contains 20 nodes including an *edge router* that connects the stub domain to the transit backbone. This gives a total of 30 transit nodes, 10 edge routers and 190 (non-edge) stub nodes. Edges are constructed for the transit domain according to the distribution defined within the Waxman model [22] with $\alpha_W = 0.3$ and $\beta_W = 0.3$ (we use the "W" subscript to distinguish these parameters used within the Waxman model from the α parameter used within the heuristic). The two underlying network topology instances differ in how α_W and β_W are set when generating edges within the stub domain. In one instance, we generate *sparse* stub domains by setting $\alpha_W = 0.3$ and $\beta_W = 0.3$. In the *dense* instance, we set $\alpha_W = 0.6$ and $\beta_W = 0.7$. The delay assigned to each edge is proportional to its length.

2) Overlay Generation: For each experimental run, a fixed number of end-systems are connected to randomly chosen stub nodes (nodes are chosen with replacement such that a single stub node can connect to multiple end-systems), and 10 proxy nodes are selected. We consider four ways of assigning proxies to nodes in the network:

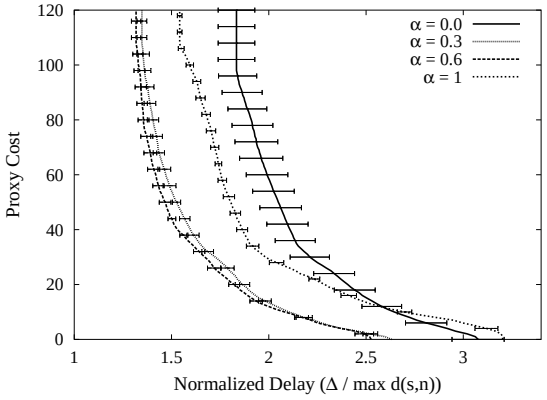
- Proxy placement is restricted to the transit backbone.
- Proxy placement is restricted to stub nodes.
- Proxy placement is restricted to edge nodes (i.e., the bridges between transit and stub)
- Proxy placement is unrestricted, with each proxy having equal likelihood of being attached to a transit, stub or edge node.

The delay between a node and the proxy connected to that node is set to 0 (i.e., the proxy is co-located with the node). Delay from a stub node to a connecting end-system is set to 0.3 times the average delay between the source and stubs that contain end-systems. The fanout from a proxy node is chosen from a uniform distribution between 5 and 15. End-system fanout is uniformly chosen between 1 and 3.

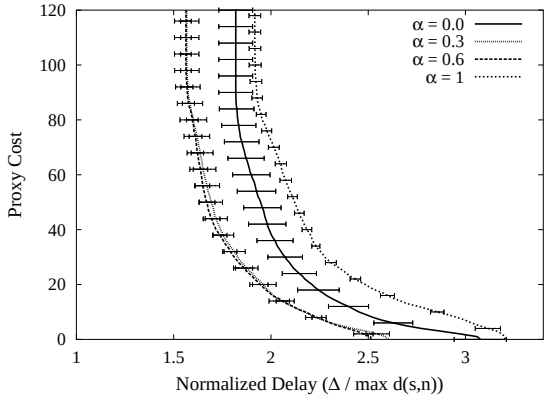
B. Performance Evaluation

Figure 3 plots the cost of the tree for a session with 100 end-systems in which end-systems meet a given delay bound. The cost function used within the simulation is the sum of the fanouts of edges extending from proxies within the tree, (i.e., the cost function is $g(i) = i$). The different sub-figures plot results for the various proxy placement restrictions. In each sub-figure, the value of the x-axis is $\Delta / \max_{n \in N} d(s, n)$, i.e., the delay bound normalized to the largest end-system end-to-end (unicast) delay from the source (in the underlying network). The y-axis gives the cost. Each curve plots, for a given α, the cost of the tree computed by the heuristic to meet the delay bound for all receivers given on the x-axis. Each point plotted is the average of 98 simulation runs. The 95% confidence intervals shown are generated using seven sample points, where each sample point is the average of 14 runs.[3] Values of α whose

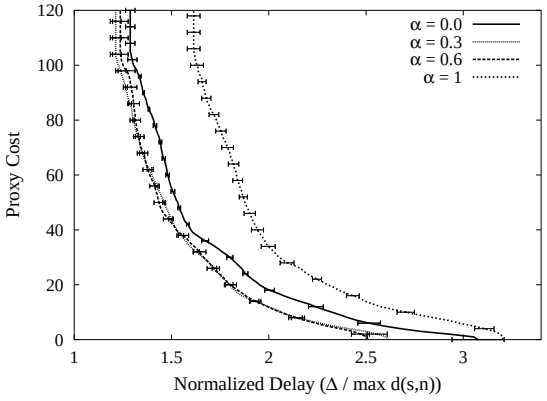
[3]Each simulation run determines the minimum delay obtainable for all end-systems given a fixed cost budget. Our averages are then computed over the set



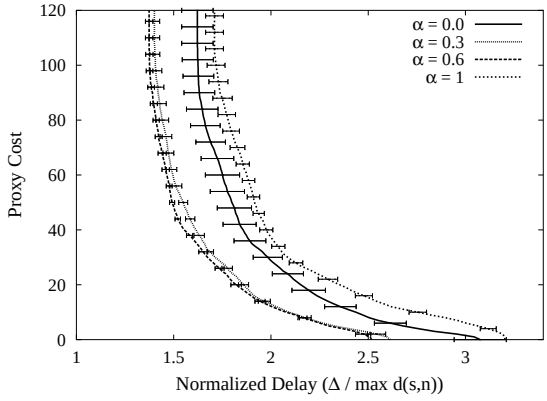
(a) Proxies on transits



(b) Proxies on stubs



(c) Proxies at edges



(d) Proxies everywhere

Fig. 3. Heuristic Performance

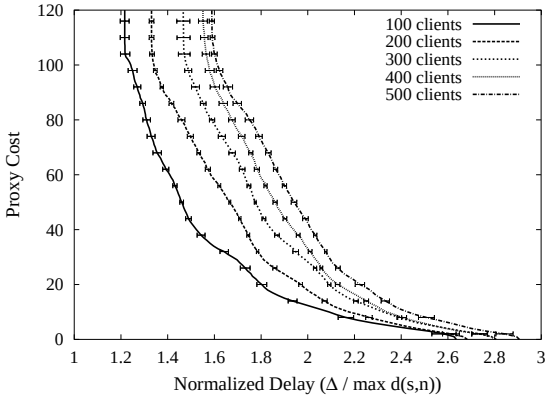
curves lie closer to the bottom and left-most edges of the graph are preferable, since these points represent lower cost trees that yield lower delays.

We see that $\alpha = 0$ is preferable to $\alpha = 1$ when proxies are restricted to either the stub or edge points but not when proxies are restricted within the transit portion of the network. Intuitively, this is because proxies in the transit portion of the network are typically closer to one another. Hence, the high fanout nodes that are first added into the tree when $\alpha = 1$ are closer together. In contrast, when proxies lie at the edges of the network, the distances (and hence delays) traversed to connect these high fanout nodes together becomes excessive. However, these observations are in some sense moot, since in all four scenarios, $\alpha = 0.6$ and $\alpha = 0.3$ produce trees whose cost is consistently lower to achieve any fixed bound on the delay. This demonstrates that the "best" trees result from giving a more equal consideration to both the transmission delay and bandwidth constraints. The heuristic used in [7] is equivalent to our heuristic with $\alpha = 1$. The point at which the curves touch the x-axis indicates the minimum delay achievable for all receivers in a session that receives no proxy support. We see that a value of $\alpha = 0.6$ or $\alpha = 0.3$ yields almost a 25% reduction in this delay in comparison to the case where $\alpha = 1$.

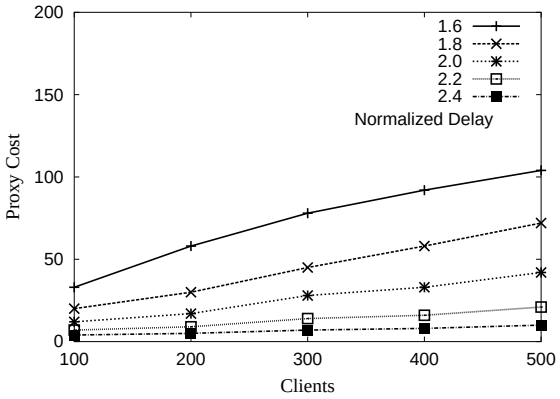
By comparing the plots across the three figures, we observe that proxies reduce costs for a fixed delay more significantly when placed at edges or in the transit portion of the network than in comparison to the edges. We have also explored how varying the number of end-systems and proxies impacts delay and cost. We see that increasing the number of end-systems grows costs, but at a very slow rate. Increasing the number of proxies decreases costs, but at a rate even slower than the increase in costs when additional end-systems are added to the session. We see the same trends irregardless of whether the stub networks are sparse or dense.

Next, we explore how proxy costs of delay-bounded trees constructed by the heuristic vary as a function of the number of receivers in the multicast session. Figure 4 plots these results along two axes. In Figure 4(a), we plot the proxy cost as a function of the delay bound using the heuristic with $\alpha = 0.3$. Each curve plots the cost for a fixed number of receivers participating in the session. This figure demonstrates that there is an increasing cost in building a tree that meets a given delay bound as the number of receivers increases, but that this cost grows slowly. Figure 4(b) further illustrates this observation,

of delays achieved for each cost value. As a result, our confidence intervals lie horizontally for each cost, instead of vertically for each delay.



(a) Proxy Cost vs. Delay



(b) Proxy Cost vs. Number of Receivers

Fig. 4. Proxy costs as a function of number of receivers and desired delay bound

plotting the cost as a function of the number of clients in the session, where each curve represents a different bound on the maximum delay to any receiver in the session. We observe an approximate linear growth in the cost as the number of clients is increased. However, we note that because it would appear as though the curves would cross the line above the y-axis, we suspect that the cost per receiver decreases as the number of receivers increase, and that the cost approaches some asymptotic value as the number of receivers grows large. In addition, the cost per receiver increases at a much higher rate as the desired delay bound is tightened. Hence, proxy-assisted multicast costs less per client as the number of clients in a session increases and as desired delay bounds are relaxed.

VII. Discussion

There are two basic directions for future work. One involves further improvement to the performance of the heuristic. First, it may be possible to optimize searches for minimum cost within the heuristic by using a binary jump process similar to what is employed within the exact algorithm. Second, we currently fix α during the construction of a tree within the heuristic. It may make sense to vary α during this process, e.g., starting with a very high α and decreasing α during the building of the tree.

The second direction involves the consideration of proxy selection in networks that account for additional network dynamics such as those handled by Chu et al in the proxy-free environment [8], [7]. These include:

- The overlay connectivity graph may not be fully connected due to communication and state maintenance overheads associated with maintaining connectivity between pairs of nodes.
- End-system members might join or leave a session in progress, requiring dynamic restructuring of the multicast overlay tree.
- Delays and bandwidth constraints can vary as a result of congestion from other sessions competing for the same network bandwidth.
- Bandwidth can be constrained at points other than last-mile hops, such that the bandwidth constraints on a pair of overlay edges need not be disjoint.
- We must consider development of an algorithm that operates in a distributed fashion.

Our results presented here give some guidelines that will help us to address some of the challenges:

- If full connectivity cannot be maintained, we suspect that "better" overlays will result when nodes with larger fanout constraints communicate directly with one another and directly with the session source.
- Proxies can be temporarily employed during periods of change of group membership. After membership appears more stable, a new tree can be formed with cost re-evaluated. Proxies can also serve as a temporary means of assisting an end-system whose bandwidth suddenly decreases. We assume of course, that a proxy would not exit a session early unless it was no longer needed or some network fault occurred.

Finally, there are more complex economic issues that can be considered. For instance, we do not consider the case where different service providers charge different rates for their proxy services. Hence, the "best" proxy to minimize the cost need not be the greatest fanout, least delay proxy, but might also be the cheapest.

VIII. Conclusion

In this paper, we evaluated algorithms and heuristics to minimize the cost of employing proxies as a means of meeting delay constraints for hybrid end-system/proxy application layer multicast sessions. We present an algorithm that provably minimizes costs for the case where delays between application layer multicast points is uniform. For the case where delays between these points vary, solving the problem optimally is NP-hard. We instead resort to a tunable heuristic, and demonstrate that certain tunings outperform previously developed heuristics. Our results demonstrate that these hybrid approaches are a promising means for enabling low-cost, delay bounded multicast services upon a unicast-only network layer.

REFERENCES

[1] Akamai Corporation. Internet Bottlenecks: The Case for Edge Delivery Services, 2000. Akamai whitepaper.

[2] F. Bauer and A. Varma. Degree-constrained Multicasting in Point-to-point Networks. In *Proceedings of IEEE INFOCOM'95*, Boston, MA, March 1995.

[3] Fred Bauer and Anujan Varma. Distributed Degree-Constrained Multicasting in Point-to-Point Networks. Technical Report UCSC-CRL-95-09, UCSC, 1995.

[4] Ken Calvert and Ellen Zegura. Gt internetwork topology models (gt-itm). *http://www.cc.gatech.edu/fac/Ellen.Zegura/graphs.html*, 1997.

[5] M. Charikar, C. Chekuri, T. Cheung, Z. Dai, A. Goel, S. Guha, and M. Li. Approximation Algorithms for Directed Steiner Problems. In *ACM-SIAM Symposium on Discrete Algorithms*, San Francisco, CA, January 1998.

[6] Y. Chawathe, S. McCanne, and E. Brewer. RMX: Reliable Multicast for Heterogeneous Networks. In *Proceedings of IEEE INFOCOM'00*, Tel-Aviv, Israel, March 2000.

[7] Y. Chu, S. Rao, S. Seshan, and H. Zhang. Enabling Conferencing Applications on the Internet Using an Overlay Multicast Architecture. In *Proceedings of ACM SIGCOMM'01*, San Diego, CA, August 2001.

[8] Y. Chu, S. Rao, and H. Zhang. A Case for End System Multicast. In *Proceedings of ACM SIGMETRICS'00*, Santa Clara, CA, May 2000.

[9] C. Diot, B. Levine, B. Lyles, H. Kassem, and D. Balensiefen. Deployment Issues for the IP Multicast Service and Architecture. *IEEE Network Magazine*, Jan/Feb 2000.

[10] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On Power-Law Relationships of the Internet Topology. In *Proceedings of SIGCOMM'99*, Cambridge, MA, September 1999.

[11] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, New York, NY, 1979.

[12] A. Goel, K.G. Ramakrishnan, D. Kataria, and L. Logothetis. Efficient Computation of Delay-sensitive Routes from One Source to All Destinations. In *Proceedings of IEEE INFOCOM'01*, Anchorage, AK, April 2001.

[13] Inktomi Corporation. The Inktomi Overlay Solution for Streaming Media Broadcasts. Inktomi whitepaper.

[14] J. Jannotti, D. Gifford, K. Johnson, M. Kaashoek, and J. O'Toole. Overcast: Reliable Multicasting with an Overlay Network. In *Proceedings of USENIX*, San Diego, CA, October 2000.

[15] Vachaspathi P. Kompella, Joseph C. Pasquale, and George C. Polyzos. Multicast Routing for Multimedia Communication. *IEEE/ACM Transactions on Networking*, 1(3):286–292, 1993.

[16] Mehrdad Parsa, Qing Zhu, and J. J. Garcia-Luna-Aceves. An Iterative Algorithm for Delay-constrained Minimum-cost Multicasting. *IEEE/ACM Transactions on Networking*, 6(4):461–474, 1998.

[17] D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel. ALMI: An Application Level Multicast Infrastructure. In *3rd Usenix Symposium on Internet Technologies and systems (USITS)*, March 2001.

[18] S. Ramanathan. Multicast Tree Generation in Networks with Asymmetric Links. *IEEE/ACM Transactions on Networking*, 4(4), 1996.

[19] S. Vutukury and J.J. Garcia-Luna-Aceves. A Practical Framework for Minimum-Delay Routing in Computer Networks. *Journal of High Speed Networks*, 8(4), 1999.

[20] B. Wang and J. Hou. Multicast Routing and its QoS Extension. *IEEE Network*, Jan/Feb 2000.

[21] Z. Wang and J. Crowcroft. Bandwidth-delay Based Routing Algorithms. In *IEEE Globecom'95*, November 1995.

[22] Bernard M. Waxman. Routing of multipoint connections. *IEEE Journal on Selected Areas in Cummunications, 6(9): 1617-1622*, 1988.

[23] Ellen Zegura, Ken Calvert, and Samrat Bhattacharjee. How to model an internetwork. *Proceedings of IEEE Infocom'96*, 1996.

[24] Q. Zhu, M. Parsa, and J.J. Garcia-Luna-Aceves. A Source-based Algorithm for Delay Constrained Minimum Cost Multicasting. In *Proceedings of IEEE INFOCOM'95*, Boston, MA, March 1995.

APPENDIX

A. The Need for Heuristics

In Section V, we claim that the problem of finding a tree whose longest path from a given source node s to any node is minimized, with the restriction that within the tree, each node n has a bound on its fanout, $f()$, is NP-hard. We did not find a proof of this fact within the literature. In particular, there is no obvious reduction from a Steiner Tree problem. Also, we seek a shortest paths tree, and not a minimum spanning tree as in [3], [2]. We provide a sketch here of the reduction from Hamiltonian Path. While we doubt the results presented here are novel, we include them for completeness.

We begin by considering a graph $G = (N, E)$ that is not fully connected in which all edges have unit length. If we set $f(n) = 1$ for all $n \in G$ (bounding the permitted fanout that can be used to construct the tree, not bounding its fanout in the underlying graph, G), then a shortest-path tree would have depth $|N| - 1$ iff a Hamiltonian Path existed within the graph, and finding a Hamiltonian path is known to be NP-complete [11].[4] Thus, finding a shortest-path tree in G is NP-hard. To show the problem is NP-hard for graphs in which the tree can have arbitrary (but bounded) fanout $f(n) < N$, we construct a new graph G' that is identical to G except we add some additional nodes and edges. For each node n with $f(n) > 1$, create $f_1(n) = f(n) - 1$ additional nodes and create edges in G' that attach these new nodes to (and only to) n. Finding a shortest-path tree in G' that includes all nodes in G' will require n to attach to the new $f(n) - 1$ in the tree that only it attaches to in the underlying graph G'. If a tree can be constructed that does not violate fanout constraints in G', then we have constructed a Hamiltonian Path for the arbitrary graph, G. Thus, the problem of constructing a minimum-depth tree with an arbitrary FCF is NP-hard.

Last, to show that the above problem is NP-hard in a fully-connected graph G with variable-length edges, we construct G as follows: Choose an arbitrary graph with unit edge-lengths of 1. Wherever an edge does not exist, add an edge of length $\Delta + 1$, where Δ is the desired delay bound. Clearly, these new edges cannot be used in a tree that solves our problem, so solving the problem reduces to solving the problem on an arbitrary graph with unit length edges. Thus, solving the problem on a variable-length edge fully-connected graph is NP-hard.

[4] Thanks to Micah Adler for pointing us to this reduction.