Placing Multicast Proxies for Internet Live Media Streaming

Jun Guo, Sanjay Jha

School of Computer Science and Engineering

The University of New South Wales, Australia

Email: {jguo, sjha}@cse.unsw.edu.au

Abstract— **Placing multicast proxies on the Internet can largely reduce the delivery delay of Internet live media streaming using overlay multicast mechanisms. In this paper, we propose a new approach to address the issue of proxy placement in an overlay multicast network. The task is to decide an optimal placement of multicast proxies in the overlay multicast network so as to minimize the average end-to-end delay of the overlay skeleton tree subject to out-degree balancing constraint and maximum delivery delay bound. We present two heuristic methods for this proxy placement problem. Experimental results demonstrate that proxy placement due to our proposed approach can greatly improve the overlay multicast routing performance. In comparison with an existing approach commonly used for replica placement in content distribution networks, our proposed approach significantly improves the end-to-end latency performance of overlay multicast networks, and is not much sensitive to network dynamics.**

I. INTRODUCTION

Live media streaming on the Internet is becoming increasingly popular as more and more media companies wish to leverage Internet broadcast channels to reach global audience. Since sessions for sought-after events may incur tens of thousands of simultaneous users [1], it is desirable to implement Internet live media streaming cost-effectively. While IP multicast [2] is doubtlessly the ideal solution for supporting large-scale Internet live media streaming, and extensive studies on IP multicast have been conducted over the past two decades, enabling IP multicast across the global Internet has not been successful due to its various deployment issues [3]. Until the global deployment of IP multicast could be eventually realized, the Internet community has to resort to interim multicast solutions [4]. Among them, end-system multicast [5]–[7] and proxy-assisted multicast [8]–[10] are two representative overlay multicast solutions, both of which emulate the forwarding mechanism of IP multicast at the application layer.

In end-system multicast, hosts wishing to participate in a live streaming session self-organize into an overlay tree from the source of the session. Multicast-related functionality is completely implemented by end systems via unicast connections. Since broadband access technologies of today's Internet provide users with very limited outgoing bandwidth, the fanout capability of end systems is significantly restricted. As a result, overlay trees in end-system multicast can be rather tall, so that hosts deep in the tree (hence far from the source) are likely to suffer considerable lag [11]. Moreover, due to the presence of group dynamics, reconstruction of overlay trees can be very frequent in end-system multicast, which yields glitch and incurs heavy maintenance overhead. Although studies of live streaming workloads on the Internet [12] indicated that it is possible to support large-scale live media streaming using end-system multicast, the observations are drawn from experiments with low-quality video streams. It is unlikely that existing broadband access technologies can support efficient end-system multicast for high-quality video streaming.

As a middle-ground solution between end-system multicast and IP multicast, proxy-assisted multicast is designed to offer better performance than end-system multicast and require less infrastructure support than IP multicast [10]. This approach relies on a set of dedicated multicast proxies with large processing power and high fanout capability to construct a backbone service domain for the overlay multicast network. These application layer servers create overlay trees among themselves via unicast connections. They are typically placed at co-location facilities with high-speed connection to the Internet, so that they can readily utilize the over-provisioned links in the core networks without experiencing significant variations in end-to-end delay. They are fairly static and thus eliminate large maintenance overhead due to tree reconstruction. Communication between end systems and multicast proxies is flexible to use end-system multicast, IP multicast, or any other technique. In comparison with overlay trees formed entirely by end systems, overlay trees based on multicast proxies are much flatter. It was shown in [10] that such a two-tier overlay multicast architecture can achieve comparable performance to IP multicast.

Studies of proxy-assisted multicast have primarily focused on routing algorithms with the assumption that locations of multicast proxies are predetermined [8], [9], [13]. Less attention has been given to the development of proxy placement algorithms to further improve the performance of overlay multicast networks [14], [15].

The proxy placement problem addressed in [14] was designed for overlay multicast within a single autonomous system (AS). The nature of their problem formulation requires the detailed knowledge of the substrate network topology, which does not scale to inter-domain proxy placement. An inter-domain proxy placement problem was later considered in [15]. Their problem formulation aims to minimize the weighted average delay between each node in the overlay network and its closest multicast proxy, which resembles the K-median

problem commonly used for replica placement in content distribution networks (CDNs) [16]. However, connecting each node in the overlay network to its closest multicast proxy does not necessarily yield good latency performance from the *source-to-end* point of view in overlay multicast networks [14].

In this paper, we propose a new inter-domain proxy placement problem for overlay multicast networks to support large-scale Internet live media streaming. Our proposed approach is going to help a content service provider to automate and optimize the proxy placement decision across the Internet. Similar to Qiu *et al.* [16], we allow the content service provider to use the technique of [17] for strategically clustering its Internet clients, and map each cluster to one *client site* for potential proxy placement. We assume that in each client site there exists a co-location center. To deploy a set of multicast proxies and to operate the overlay multicast network, the content service provider enters into contracts with the selected co-location centers for access bandwidth usage and service level agreements [18]. Once the multicast proxies are deployed, live contents can be distributed in real-time from the origin server to niche audience across the globe using the overlay multicast network infrastructure.

Our proposed approach is routing-aware in a sense that it explicitly measures the quality of a proxy placement solution in terms of the long-term average latency performance of an *overlay skeleton tree*. Given the set of N client sites and the set of K ($K < N$) multicast proxies, an overlay skeleton tree is defined in such a way that it spans all N client sites via the set of K multicast proxies. We identify two factors to support such a routing-aware approach. Firstly, studies of live streaming workloads on the Internet have found that it is not uncommon that popular streams can reach a wide audience spanning hundreds of AS domains [1]. Thus, it is reasonable to expect that sessions for sought-after events may incur large amount of user traffic from all client sites and bring all multicast proxies into play for good quality live streaming. Secondly, for sessions composed of a subset of client sites, we can readily utilize subtrees of the overlay skeleton tree for the purpose of data delivery. Thus, if the end-to-end latency performance of the overlay skeleton tree is optimized, its subtrees are likely to yield satisfactory latency performance.

We summarize our contributions in this paper as below:

1) We propose a new approach which we call RAPP to address the issue of proxy placement in an overlay multicast network. RAPP is a recursive acronym which stands for RAPP is an Approach for Proxy Placement. The proposed approach allows a content service provider to automate and optimize the proxy placement decision across the Internet. In comparison with the K-median problem presented in [15], solutions to the RAPP problem can significantly improve the end-to-end latency performance of overlay skeleton trees. The performance gain is not much sensitive to network dynamics.

2) We present a simple heuristic method called STAMP for the RAPP problem. STAMP stands for Simple Technique for Allocation of Multicast Proxies. It obtains

good quality solutions to the RAPP problem yet with lower computational complexity $O(N^2)$ than $O(KN^2)$ of the greedy algorithm proposed in [15] for the K-median problem.

3) We design a weight-coded genetic algorithm (GA) for the RAPP problem to obtain closer-to-optimal solutions for fairly large size problem instances with reasonable computational complexity.

4) We provide an integer linear programming (ILP) formulation of the RAPP problem. The ILP model can be used to compute optimal solutions for small size problem instances. We believe that it paves the way for further development of approximation approaches to solve the RAPP problem [19].

The remainder of this paper is organized as follows. Section II deals with the problem formulation. In Section III, we describe our solution methods. Simulation experiments are reported in Section IV. Finally, we provide concluding remarks in Section V.

II. PROBLEM FORMULATION

We model the overlay multicast network as a complete directed graph $G = (V, E)$, where V is the set of N nodes and $E = V \times V$ is the set of edges. Each node in V represents a client site. Let node r be the client site where the origin server is situated. K ($K < N$) multicast proxies are to be placed among N client sites, which constitute the set $P \subset V$. As in [14], we place the root proxy in node r by default. For brevity, we call client sites with proxies co-located as *proxy nodes* and those without proxies co-located as *non-proxy nodes*.

The weight c_i of node i is given by the long-term average number of clients concentrated at the corresponding client site. The directed edge $\langle i, j \rangle$ in E from node i to node j represents the unicast path of latency $l_{i,j}$ from node i to node j. In practice, unicast latency quantities between any two nodes can be estimated from long-term average latency of end-to-end Internet paths using online measurement techniques [20], [21]. By $\{l_{i,j}\}$, we denote the matrix of unicast latency quantities between each pair of nodes in G.

An overlay skeleton tree can be represented by a directed spanning tree T of G rooted at node r. A directed edge from node i to node j, $i \in V$, $j \in V - \{r\}$, can be included in T if and only if node i is in P. Consequently, the set of internal nodes of T is composed of proxy nodes only, and the set of leaf nodes of T is exclusively composed of non-proxy nodes.

The objective of the RAPP problem is to place the set of multicast proxies on the set of client sites such that the average end-to-end delay of T over all client sites is minimized. Since client sites with higher weights are more important than those with lower weights, we translate the optimality criterion of the RAPP problem to minimizing the normalized weighted sum of latencies from the root proxy to each client site along T. Additionally, we introduce a maximum delivery delay bound for the RAPP problem. This offers the content service provider an additional degree of freedom for controlling the maximum end-to-end delay of T within a certain threshold.

Moreover, we further introduce a load balancing constraint for the RAPP problem, since it is undesirable that any multicast proxy is excessively loaded in the overlay skeleton tree. For this purpose, it suffices by enforcing that the out-degree is balanced among all internal nodes of T.

For each sink node in the set $V - \{r\}$, we define $R_{r,v}$ as the set of directed edges that form the overlay routing path from the root proxy to node v. Let $L_{r,v}$ denote the latency of the overlay routing path from the root proxy to node v. Let $\bar{L}$ denote the average end-to-end delay. Let $L_{\max}$ denote the maximum end-to-end delay. Let $L_{\max}^{\mathrm{B}}$ denote the specified bound on $L_{\max}$. Given the unicast latency matrix $\{l_{i,j}\}$, we readily have

$$L_{r,v} = \sum_{\langle i,j \rangle \in R_{r,v}} l_{i,j} \qquad (1)$$

and

$$\bar{L} = \sum_{v \in V-\{r\}} c_v L_{r,v} \Big/ \sum_{v \in V-\{r\}} c_v \qquad (2)$$

and

$$L_{\max} = \max_{v \in V-\{r\}} L_{r,v} . \qquad (3)$$

Let $d(i)$ denote the out-degree of an internal node i, $i \in P$. The following proposition states a property on the sum $\sum_{i \in P} d(i)$ of out-degrees in an overlay skeleton tree.

Proposition 1: The sum of out-degrees in an overlay skeleton tree is $N - 1$.

Proof: It follows from Corollary 1.5.3 of [22] that a spanning tree with N nodes has exactly $N - 1$ edges. By the definition of an overlay skeleton tree, between each pair of nodes i and j, the edge is either directed from node i to node j, or directed from node j to node i. In either case, it contributes one count towards the sum of out-degrees. ∎

We define an out-degree balancing index F, given by

$$F = \max_{i \in P} d(i) - \min_{i \in P} d(i) . \qquad (4)$$

A smaller value of F indicates a more balanced out-degree distribution within the set of internal nodes. Let F^{LB} denote the lower bound (LB) on F. Clearly, $F^{\mathrm{LB}} = 0$ if

$$N - 1 = mK, \ m = 1, 2, \ldots \qquad (5)$$

and we require each internal node to have an out-degree of exactly m. In situations where (5) does not hold, $F^{\mathrm{LB}} = 1$. In such cases, letting $m = \lfloor \frac{N-1}{K} \rfloor$ and $n = N - 1 - mK$, we require n internal nodes to have an out-degree of exactly $m + 1$, and the remaining $K - n$ internal nodes to have an out-degree of exactly m.

Definition 1: **The RAPP problem**

Given a complete directed graph $G = (V, E)$ of N nodes, find a constrained directed spanning tree T of G rooted at node r, such that $\bar{L}$ is minimized, and T is subject to constraints on: 1) K, $K < N$, nodes (including node r) are selected as internal nodes, all other $N - K$ nodes are for leaf nodes; 2) $L_{\max}$ is bounded by $L_{\max}^{\mathrm{B}}$; 3) F is restricted to F^{LB}.

Such a heavily constrained spanning tree problem is NP-hard, which can be established by the theorem presented in the Appendix. We thus resort to heuristic methods to find near-optimal solutions for this challenging problem.

III. SOLUTIONS

In this section, we present two heuristic methods for the RAPP problem. The first method is a simple technique for allocation of multicast proxies which we call STAMP for brevity in this paper. The second method is a meta-heuristic approach based on a weight-coded GA. We also provide an ILP formulation of the RAPP problem.

A. STAMP

Given the unicast latency matrix $\{l_{i,j}\}$, we compute s_v for each node v in the set $V - \{r\}$ by

$$s_v = c_v l_{r,v} + \sum_{j \in V-\{r,v\}} c_j (l_{r,v} + l_{v,j}) . \qquad (6)$$

We then identify $K - 1$ nodes in $V - \{r\}$ with the smallest values on s_v. These $K - 1$ nodes together with node r constitute the set P of internal nodes of the overlay skeleton tree T. Intuitively, this method aims to find nodes that (should they be chosen as internal nodes) would more likely result in smaller latency of the overlay routing path between each leaf node and the root node.

For each computation of (6), STAMP requires $2N - 4$ additions and $N - 2$ multiplications. This results in $O(N^2)$ complexity for computing s_v for all $N-1$ nodes in the set $V - \{r\}$. Then, in order to identify $K - 1$ nodes with the smallest values on s_v, it either requires a total of $(K-1)(2N-K-2)/2$ comparisons of complexity $O(KN)$, or suffices by applying a quick sort of complexity $O(N \log N)$ on s_v if $K > \log N$. Therefore, STAMP is $O(N^2)$ in complexity.

B. Weight-coded GA

GAs are population-based stochastic search and optimization approaches inspired by the mechanism of natural selection which obeys the rule of "survival of the fittest" [23]. As shown in Fig. 1, in a typical implementation of GAs, a population of chromosomes is processed. Each chromosome represents a candidate solution to the problem. Starting from an initial population of randomly created chromosomes, GAs perform multi-directional stochastic search through a genetic evolution process without the need for any problem information except the objective function values. It is hoped that after a certain number of generations, the best chromosome represents a good quality solution that is reasonably close to the optimal solution.

GAs have been extensively used for solving various real-world complex optimization problems due to their broad applicability, ease of use and global perspective. In particular, they have found successful applications in a number of tree-based network optimization problems (see e.g. [24] and references therein). In all cases, it was observed that GAs can achieve near-optimal solutions for fairly large size instances with reasonable computational effort. We choose to implement a weight-coded GA [25] for the RAPP problem, since it uses a node-based encoding approach for chromosome representation
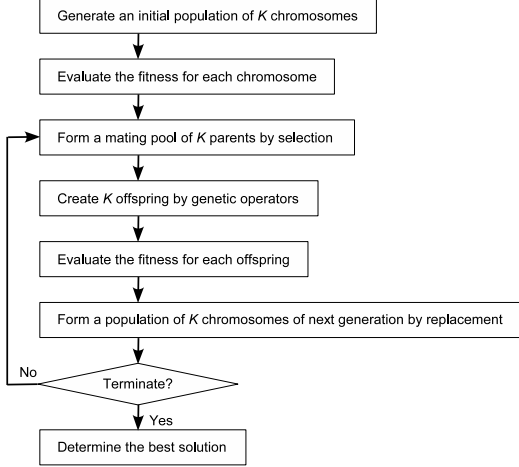
Fig. 1. Flowchart of GAs.

of candidate solutions to the problem, rather than an edge-based encoding approach which is inefficient in the context of a complete graph.

1) Tree computation heuristic: Implementing a weight-coded GA for the RAPP problem requires a heuristic method (preferably with low complexity) for computing the overlay skeleton tree. For this purpose, we have modified the centralized greedy algorithm presented in [7] to obtain a low complexity tree computation algorithm tailored to the RAPP problem. Given the proxy placement solution computed from STAMP, the greedy algorithm proceeds in two stages. The first stage deals with the construction of the subtree of the overlay skeleton tree connecting proxy nodes only. The second stage manages the connection between non-proxy nodes and proxy nodes to complete the overlay skeleton tree.

An important objective of the greedy algorithm is to guarantee that the out-degree balancing index F of T is restricted to F^{LB}. As we have discussed in Section II, if $F^{\text{LB}} = 0$, we require each proxy node to have an out-degree of exactly m, where $m = \frac{N-1}{K}$. This case can be trivially resolved. For each node v that has just been added to the partial tree of T, we check the cumulated out-degree $d(i)$ of node i which connects node v. If $d(i) = m$, we mark node i so that node i will not be considered by any of the remaining unconnected nodes in the set $V - \{r\}$. On the other hand, if $F^{\text{LB}} = 1$, we require n proxy nodes to have an out-degree of exactly $m + 1$, and the remaining $K - n$ proxy nodes to have an out-degree of exactly m, where in this case $m = \lfloor \frac{N-1}{K} \rfloor$ and $n = N - 1 - mK$. To achieve this, we make sure that once the n-th proxy node whose out-degree reaches $m + 1$ has been marked, we further check for each unmarked proxy node if its out-degree has reached m. If so, we mark such nodes accordingly, again to make sure that they will not be considered by any of the remaining unconnected nodes in the set $V - \{r\}$.

The tree construction process starts by arranging all $K - 1$ proxy nodes in the set $P - \{r\}$ in a non-decreasing order

according to their s_v values computed from (6). Then, we add these nodes one-by-one in that order to T. Let P' denote the set of all proxy nodes in the partial tree of T constructed so far. Let $\bar{P}$ denote the set of all proxy nodes that have been marked. Starting from the initial tree with P' including node r only, for each unconnected proxy node v in the ordered set $P - \{r\}$, we find an unmarked proxy node u in the partial tree of T constructed so far such that $c_v(L_{r,u} + l_{u,v})$ is the smallest. After node v is added to T, we update P', and we apply the out-degree balancing module to check if node u needs to be marked into the set $\bar{P}$.

Once the subtree of T containing all proxy nodes is formed, we then connect non-proxy nodes one-by-one to T. Let W denote the set of all non-proxy nodes not yet in T, and initialize W to $V - P$. At any iteration, for each node v in the set W, we find an unmarked proxy node u in T which minimizes $\delta_v = c_v(L_{r,u} + l_{u,v})$. We then identify node v in W with the largest value of such δ_v, and add it to T by creating a directed edge from node u to node v. After node v is added to T, we update W, and we again apply the out-degree balancing module to check if node u needs to be marked into the set $\bar{P}$.

Clearly, this greedy algorithm ensures that T satisfies all constraints of the RAPP problem except the bound $L_{\text{max}}^{\text{B}}$ on L_{max}. It can be shown that this greedy algorithm is $O(KN^2)$ in complexity.

2) GA implementation: By weighted coding [25], we express a chromosome of GA in the form of a real-valued vector $\mathbf{w}$ of size N. Each gene w_i, $i = 1, 2, \ldots, N$, of the chromosome $\mathbf{w}$ holds a biasing weight in the range $(0, 1)$. The string of N biasing weights is used to produce an altered version of the unicast latency matrix $\{l_{i,j}\}$ by

$$l'_{i,j} = l_{i,j} \cdot w_i \cdot w_j \tag{7}$$

for all $\langle i, j \rangle \in E$. STAMP and the tree computation heuristic are applied to identify a candidate solution to the RAPP problem represented by $\mathbf{w}$ using the altered matrix $\{l'_{i,j}\}$, but the corresponding $\bar{L}$ and L_{max} values of this solution are computed using the original matrix $\{l_{i,j}\}$. This way, we can effectively operate GA in the heavily constrained solution space of the RAPP problem by simply exploring the space of biasing weights and handling any constraint violation on L_{max} in each generation of the evolution process.

For the purpose of constraint handling, we use an efficient approach proposed in [26] to guide the weight-coded GA towards the constrained optimum without the need for any penalty parameter in the fitness function. Let $L_{\text{max}}(\mathbf{w})$ and $\bar{L}(\mathbf{w})$ respectively denote the L_{max} and $\bar{L}$ values of the candidate solution represented by $\mathbf{w}$. Let $\bar{L}_{\text{max}}$ be the $\bar{L}$ value of the worst feasible chromosome in the population. Let $L_{\text{max}}^{\text{UB}}$ be an upper bound (UB) on L_{max}. Let $\bar{L}^{\text{UB}}$ be an upper bound on $\bar{L}$. Note that both $L_{\text{max}}^{\text{UB}}$ and $\bar{L}^{\text{UB}}$ can be safely set to $(N - 1) \cdot \max_{\langle i,j \rangle \in E} l_{i,j}$ for the RAPP problem.

If $\mathbf{w}$ is a feasible chromosome, i.e. $L_{\text{max}}(\mathbf{w}) \leq L_{\text{max}}^{\text{B}}$, we evaluate its fitness $f(\mathbf{w})$ by

$$f(\mathbf{w}) = \left[L_{\text{max}}^{\text{UB}} - L_{\text{max}}^{\text{B}} \right] + \left[\bar{L}^{\text{UB}} - \bar{L}(\mathbf{w}) \right]. \tag{8}$$

On the other hand, if $\mathbf{w}$ is an infeasible chromosome, i.e. $L_{\max}(\mathbf{w}) > L_{\max}^{\mathrm{B}}$, we evaluate its fitness $f(\mathbf{w})$ by

$$f(\mathbf{w}) = \left[L_{\max}^{\mathrm{UB}} - L_{\max}(\mathbf{w}) \right] + \left[\bar{L}^{\mathrm{UB}} - \bar{L}_{\max} \right]. \quad (9)$$

With the fitness function defined in this form, we ensure that the higher fitness value a chromosome has, the better candidate solution it represents. Moreover, we also guarantee that 1) the fitness value of a feasible chromosome is higher than that of an infeasible chromosome, 2) the comparison between two feasible chromosomes is purely based on the objective function value $\bar{L}$, and 3) the comparison between two infeasible chromosomes uses the information of constraint violation on $L_{\max}$ alone.

The evolution process is initiated with a randomly generated population of size M, except for one in which all biasing weights are set to one. This chromosome represents the approximate solution obtained from STAMP and the tree computation heuristic directly using the original matrix $\{l_{i,j}\}$. Seeding the population in this knowledge-augmented way can substantially improve the search efficiency of GA [25]. Starting from this initial population, in each generation of the evolution process, we form a mating pool by selecting M good chromosomes, some of which may be duplicate, from the current population. We use the tournament selection operator for this purpose [26]. For two chromosomes chosen for competition in a binary tournament, the winner goes to the one with a higher fitness value. Moreover, we set the rule such that any chromosome is made to participate in exactly two tournaments. As a result, any chromosome in the current population will have at most two copies as the parents in the new population. This is particularly important in our context to prevent the good quality approximate solution from becoming a predominant chromosome which would likely lead to premature convergence of GA.

For each pair of coupled chromosomes in the mating pool, a uniform crossover operator is used to promote random information exchange between the two parents. One such operation at rate r_c generates a pair of new chromosomes, each of which inherits some parts of genetic materials from both parents. To introduce greater variability into the offspring, for each gene w_i, $i = 1, 2, \ldots, N$, of an offspring $\mathbf{w}$, we further apply a uniform mutation operator with a small probability r_m. If w_i is chosen for mutation, it is altered with a biasing weight randomly generated from the range $(0, 1)$.

We evaluate the fitness of each offspring using either (8) or (9) depending on if it represents a valid candidate solution. The parents and the offspring are then combined into one transitional population (of size $\geq M$ and $\leq 2M$). The M chromosomes with highest fitness values are chosen and placed in the new population of the next generation.

The evolution process is terminated after a predefined number of generations have been completed. Alternatively, we presume convergence if the best chromosomes are not improved over a certain succession of generations. In either case, we decide the best candidate solution with the highest fitness value from the last population.

For each generation of the weight-coded GA, it is clear that the complexity of selecting M parents from the current population of M chromosomes is no more than $O(M)$. It then requires at most M crossover operations each of which has complexity $O(N)$, and is followed by MN mutation operations of overall complexity $O(MN)$ in the worst case. Since the evaluation of the fitness for each of the M offsprings requires exactly one computation of STAMP of complexity $O(N^2)$ and one computation of the tree computation heuristic of complexity $O(KN^2)$, the overall computational complexity of one generation of GA is no more than $O(MKN^2)$.

C. ILP formulation

Here we provide an ILP formulation for the RAPP problem. Let the 0-1 variables x_i, $i \in V$, indicate if node i is selected as an internal node. Let the 0-1 variables $p_{i,j}^v$, $v \in V - \{r\}$, $\langle i, j \rangle \in E$, indicate if the directed edge $\langle i, j \rangle$ is included in the overlay routing path from node r to node v. Let the 0-1 variables $t_{i,j}$, $\langle i, j \rangle \in E$, indicate if the directed edge $\langle i, j \rangle$ is included in the overlay skeleton tree. Let the non-negative variables $d_{\max}$ and $d_{\min}$ respectively identify the maximum out-degree and the minimum out-degree among the internal nodes. For the latter, it is useful to define a non-negative variable y_i for each $i \in V$.

The RAPP problem can now be formulated as:

$$\text{Minimize} \quad \left(\sum_{v \in V - \{r\}} c_v \sum_{\langle i,j \rangle \in E} p_{i,j}^v l_{i,j} \right) \Big/ \sum_{v \in V - \{r\}} c_v \quad (10)$$

subject to

$$x_r = 1 \quad (11)$$

$$\sum_{v \in V - \{r\}} x_v = K - 1 \quad (12)$$

$$\sum_{j:\langle i,j \rangle \in E} p_{i,j}^v - \sum_{j:\langle j,i \rangle \in E} p_{j,i}^v = \begin{cases} 1, & \text{if } i = r \\ 0, & \text{if } i \in V - \{r, v\} \\ -1, & \text{if } i = v \end{cases}$$
$$\forall v \in V - \{r\} \quad (13)$$

$$\sum_{v \in V - \{r\}} p_{i,j}^v \leq (N - 1) \cdot t_{i,j}, \ \forall \langle i, j \rangle \in E \quad (14)$$

$$\sum_{\langle i,j \rangle \in E} t_{i,j} = N - 1 \quad (15)$$

$$\sum_{j:\langle i,j \rangle \in E} t_{i,j} \leq (N - 1) \cdot x_i, \ \forall i \in V \quad (16)$$

$$\sum_{\langle i,j \rangle \in E} p_{i,j}^v l_{i,j} \leq L_{\max}^{\mathrm{B}}, \ \forall v \in V - \{r\} \quad (17)$$

$$d_{\max} \geq \sum_{j:\langle i,j \rangle \in E} t_{i,j}, \ \forall i \in V \quad (18)$$

$$y_i = (N - 1) \cdot (1 - x_i) + \sum_{j:\langle i,j \rangle \in E} t_{i,j}, \ \forall i \in V \quad (19)$$

$$d_{\min} \leq y_i, \ \forall i \in V \quad (20)$$

$$d_{\max} - d_{\min} \leq F^{\mathrm{LB}} \quad (21)$$

The objective function presented in (10) is equivalent to (2) by the definition of $p_{i,j}^v$. Equations (11) and (12) restrict that K nodes (including node r by default) are selected as internal nodes. Equations (13) and (14) ensure that the solution is a directed spanning tree rooted at node r. More explicitly, they enforce one single overlay routing path for each root-sink pair. Equation (15) restricts that the sum of out-degrees counts $N - 1$, which is a necessary property of an overlay skeleton tree established in Proposition 1. Equation (16) ensures that the directed edge $\langle i, j \rangle$ can be included in the overlay skeleton tree if and only if node i is an internal node. Equation (17) restricts that the maximum end-to-end delay is bounded by $L_{\max}^{\mathrm{B}}$. Equations (18) to (21) enforce the balance of out-degree among nodes selected as internal nodes. All equations jointly ensure that the solution is a directed spanning tree rooted at node r and satisfies all constraints of the RAPP problem.

IV. SIMULATION EXPERIMENTS

We have examined our proposed algorithms for the RAPP problem through detailed simulation experiments. The various network topologies used in our experiments were obtained from the GT-ITM topology generator [27]. In particular, the small-size topologies ($N = 20$) were generated using the flat random graph model with an average node degree between 3 and 5. Unicast latency between different pairs of nodes in these flat random graphs varied from 1 to 50 msec. The two large-size topologies ($N = 100, 300$) were generated from the transit-stub graph model with an average node degree between 3 and 4. Unicast latency between different pairs of nodes in these transit-stub graphs ranged from 1 to 350 msec. It was reported in [27] that the transit-stub model was designed to generate a domain structure resembling that of the Internet.

Based on these network topologies, we designed the following experiments to study the performance of the proposed algorithms for the RAPP problem. For the small-size topologies, we considered the various instances in which we set each different node to be the root node. In all such instances, the number of multicast proxies was fixed to $K = 3$ and $K = 5$. Due to the small size of these topologies, we were able to compute the optimal solutions by solving the ILP model of the RAPP problem. For the purpose of performance comparison, we also computed the optimal proxy placement solutions due to the K-median approach using the ILP model presented in [15]. The CPLEX tool [28] was used to solve the ILP models in our experiments.

ILP is, however, known to have an exponential computational complexity and is unlikely to be solved for problems with a large number of integer variables. Consequently, we were unable to obtain ILP results for the large-size topologies. Instead, for the large-size topologies, we evaluated the performance of STAMP against that of the weight-coded GA, since the capability of GA in large-size problem instances has been widely established in the literature. Since the K-median problem is also NP-hard, we used an efficient greedy algorithm called greedy-proxy (GP) presented in [15]. There, it was shown that GP can obtain good quality approximate solutions

to the K-median problem with complexity $O(KN^2)$. Due to the limited space, we did not present results for the large-size topologies with respect to different root nodes. Instead, we randomly chose one node as the root node and varied K from 10 to 30 for each such topology. In all experiments, we conducted the weight-coded GA with ten independent runs each of which starts from a different initial population. The crossover rate r_c was varied from 0.5 to 1.0 at the step size of 0.1. The mutation probability r_m was set to $1/N$. Each run repeated the evolution process with up to 1000 generations. The population size M was set to 100.

A. Results on flat random graphs

For each instance, we first utilized the ILP model of RAPP to identify the minimal value on $L_{\max}$ achievable by RAPP. We set this value as $L_{\max}^{\mathrm{B}}$. Then, we proceeded to compute the minimal value on $\bar{L}$. We also used the ILP model of RAPP to compute the best overlay skeleton tree solution based on the given proxy placement solution obtained from STAMP, K-median and GP. This is done by fixing the indicator variables x_i defined for proxy placement in the ILP model of RAPP to the corresponding proxy nodes found by each other algorithm.

We observe from Fig. 2 that, in almost all instances, both STAMP and GP hit the optimal solutions identified by their corresponding ILP models. For $K = 3$, the performance ratio of RAPP to K-median is between 0.7 and 1 for $\bar{L}$ and between 0.5 and 1 for $L_{\max}$. For $K = 5$, the ratio is between 0.5 and 0.8 for $\bar{L}$ and between 0.5 and 0.7 for $L_{\max}$. It is interesting to see that in several instances where GP obtains only suboptimal solutions to the K-median problem, such solutions indeed yield better performance from the perspective of the latency performance of the overlay skeleton tree. This again justifies that proxy placement in the context of overlay multicast networks can not be modeled by the K-median approach in general.

B. Results on transit-stub graphs

Similarly, for each instance in these large-size topologies, we first found the minimal reachable value on $L_{\max}$ within the predefined number of generations. For doing this, we modified the fitness function of the weight-coded GA to account for $L_{\max}$ directly. We set this value as $L_{\max}^{\mathrm{B}}$. Then, we proceeded to find the minimal value on $\bar{L}$ within the predefined number of generations. We also used the weight-coded GA to find the best overlay skeleton tree solution based on the given proxy placement solution obtained from STAMP and GP.

Figure 3 shows that the performance ratio of STAMP to GP is between 0.6 and 0.8 for $\bar{L}$ and between 0.6 and 0.9 for $L_{\max}$ in the case of $N = 100$. For $N = 300$, the ratio is between 0.6 and 0.7 for both $\bar{L}$ and $L_{\max}$. The performance ratio of STAMP to RAPP in terms of $\bar{L}$ is only between 1.0 and 1.1 for both $N = 100$ and $N = 300$. It is also interesting to observe that the $L_{\max}$ results obtained by STAMP and RAPP are identical in all the various instances. This is because in these cases, both solutions from STAMP and RAPP have indeed reached the lower bound on $L_{\max}$, which is the unicast latency
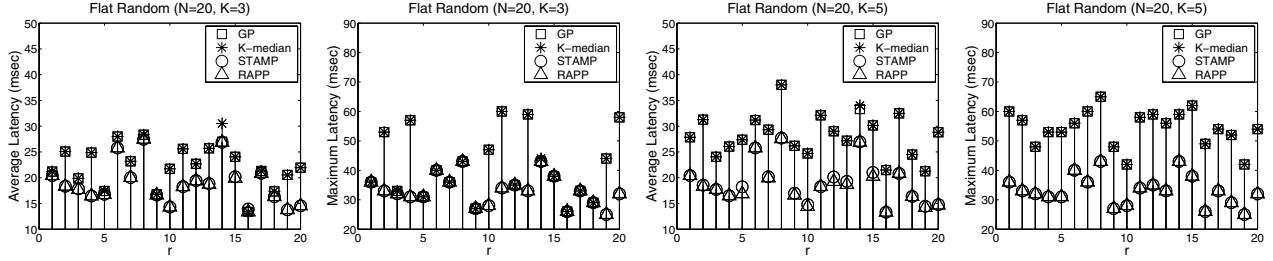
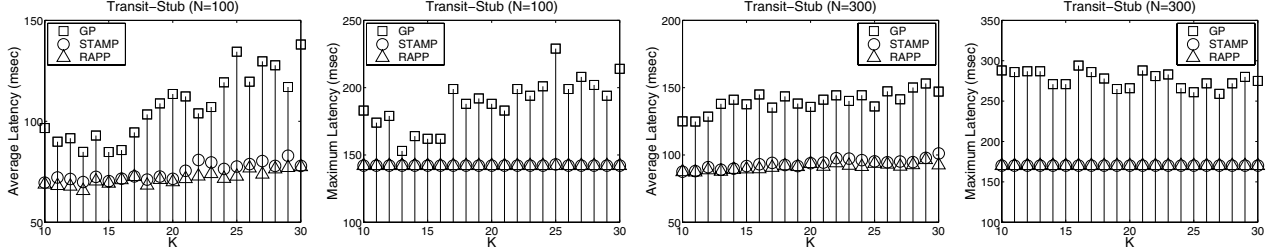Fig. 2. Quality of proxy placement with tree computation using ILP.

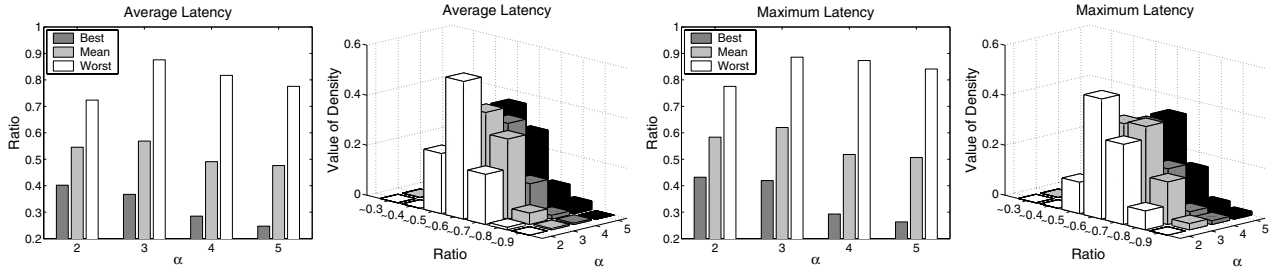Fig. 3. Quality of proxy placement with tree computation using weight-coded GA.

Fig. 4. Impact of network dynamics on proxy placement. $\sim x$ denotes the interval $[x - 0.1, \, x)$.

from the root node to the farthest sink node in the physical topology. This again confirms the capability of STAMP as well as the weight-coded GA.

C. Impact of network dynamics

To study the impact of network dynamics on the performance of STAMP and GP, we randomly altered each element of the unicast latency matrix in the range between $l_{i,j}/\alpha$ and $\alpha \cdot l_{i,j}$. The parameter α was varied from 2 up to 5. We used the transit-stub graphs with $N = 100$ and set $K = 20$. For each setting of α, we conducted 200 independent runs. Figure 4 shows the best, mean, and worst performance ratio of STAMP to GP under each setting of α. We also plotted the density histogram of the ratio in Fig. 4 for both $\bar{L}$ and $L_{\max}$.

We observe that the performance ratio of STAMP to GP is around 0.5 in both $\bar{L}$ and $L_{\max}$, even when α is as high as 5. Although the ratio in the worst scenario is as high as 0.88, such a case is indeed rare as can be confirmed from the density histogram of the ratio. Besides, Internet measurements [29] have shown that the vast majority of connections in today's Internet do not actually experience significant variations in end-to-end delay during their lifetime.

Since STAMP consistently obtains proxy placement solutions leading to better latency performance of the overlay skeleton tree even in the presence of considerable network dynamics, we draw the conclusion that STAMP is more applicable than GP to address inter-domain proxy placement in overlay multicast networks. This is especially true in consideration of the lower complexity $O(N^2)$ of STAMP than $O(KN^2)$ of GP. Consequently, STAMP can obtain much better proxy placement solutions yet at the cost of smaller amount of CPU time as demonstrated in Fig. 5.

V. CONCLUSION

Proxy placement can have a large impact on the latency performance of overlay multicast networks, especially for emerging applications like large-scale Internet live media streaming. The existing proposal tackled the problem of inter-domain multicast proxy placement by adopting the K-median approach. However, our experiments in this paper have demonstrated that the K-median approach does not yield good quality proxy placement solutions from the source-to-end latency performance point of view in overlay multicast networks. Strongly motivated, we have proposed in this paper
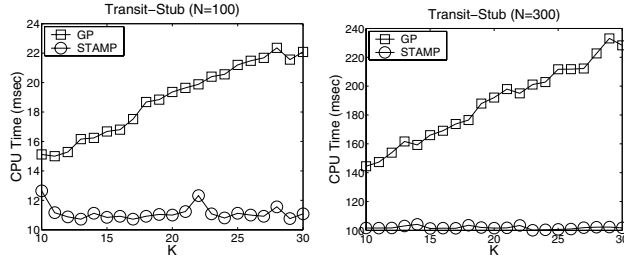
Fig. 5. CPU time comparison on a 3.2GHz Xeon machine between one computation of GP and one computation of STAMP.

a new approach and formulated the RAPP problem to address inter-domain proxy placement, aiming to support large-scale Internet live media streaming more efficiently using overlay multicast mechanisms. We have presented two efficient heuristic methods for the RAPP problem, which have been shown by simulation experiments to be scalable for large-size overlay multicast networks and yield near-optimal latency performance of overlay skeleton trees. Simulation experiments have confirmed that our proposed algorithms can significantly improve the latency performance of overlay skeleton trees in comparison with the K-median approach. In addition, the performance gain is not much sensitive to network dynamics.

APPENDIX

Theorem 1: The RAPP problem is NP-hard.

Proof: It is easy to see that the decision version Π of the RAPP problem is in the class of NP. This is because a non-deterministic algorithm needs only to guess a directed spanning tree rooted at node r and check in polynomial time if it satisfies all constraints of the RAPP problem. Having $M = N - 1$, $c_i = c_j$, $\forall i, j \in V - \{r\}$, $i \neq j$, and allowing $L_{\max}^{\mathrm{B}} = L_{\max}^{\mathrm{UB}}$, this restricts Π to a variant of the traveling salesman problem respecting mean arrival time of the tour starting from the root node ([30], page 211), which is NP-complete according to [31]. ∎

REFERENCES

[1] K. Sripanidkulchai, B. Maggs, and H. Zhang, "An analysis of live streaming workloads on the Internet," in *Proc. ACM IMC 04*, Taormina, Sicily, Italy, Oct. 2004, pp. 41–54.

[2] S. E. Deering and D. R. Cheriton, "Multicast routing in datagram internetworks and extended LANs," *ACM Trans. Comput. Syst.*, vol. 8, no. 2, pp. 85–110, May 1990.

[3] C. Diot, B. N. Levine, B. Lyles, H. Kassem, and D. Balensiefen, "Deployment issues for the IP multicast service and architecture," *IEEE Network*, vol. 14, no. 1, pp. 78–88, Jan./Feb. 2000.

[4] IRTF SAM RG, http://www.samrg.org, 2007.

[5] Y.-H. Chu, S. G. Rao, and H. Zhang, "A case for end system multicast," in *Proc. ACM SIGMETRICS 00*, Santa Clara, CA, USA, Jun. 2000, pp. 1–12.

[6] M. Kwon and S. Fahmy, "Topology-aware overlay networks for group communication," in *Proc. ACM NOSSDAV 02*, Miami, FL, USA, May 2002, pp. 127–136.

[7] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," in *Proc. ACM SIGCOMM 02*, Pittsburgh, PA, USA, Aug. 2002, pp. 205–217.

[8] S. Y. Shi and J. S. Turner, "Routing in overlay multicast networks," in *Proc. IEEE INFOCOM 02*, vol. 3, New York, NY, USA, Jun. 2002, pp. 1200–1208.

[9] S. Banerjee, C. Kommareddy, K. Kar, B. Bhattacharjee, and S. Khuller, "Construction of an efficient overlay multicast infrastructure for real-time applications," in *Proc. IEEE INFOCOM 03*, vol. 2, San Francisco, CA, USA, Mar. 2003, pp. 1521–1531.

[10] L. Lao, J.-H. Cui, and M. Gerla, "TOMA: a viable solution for large-scale multicast service support," in *Proc. IFIP NETWORKING 05*, Waterloo, Ontario, Canada, May 2005, pp. 906–917.

[11] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross, "Insights into PPLive: A measurement study of a large-scale P2P IPTV system," in *Proc. Workshop on Internet Protocol TV (IPTV) services over World Wide Web in conjunction with WWW 2006*, Edinburgh, Scotland, May 2006 2006.

[12] K. Sripanidkulchai, A. Ganjam, B. Maggs, and H. Zhang, "The feasibility of supporting large-scale live streaming applications with dynamic application end-points," in *Proc. ACM SIGCOMM 04*, Portland, OR, USA, Aug. 2004, pp. 107–120.

[13] N. M. Malouch, Z. Liu, D. Rubenstein, and S. Sahu, "A graph theoretic approach to bounding delay in proxy-assisted, end-system multicast," in *Proc. IEEE IWQoS 02*, May 2002, pp. 106–115.

[14] X. Wu, B. S. Sharif, and O. R. Hinton, "An improved resource allocation scheme for plane cover multiple access using genetic algorithm," *IEEE Trans. Evol. Comput.*, vol. 9, no. 1, pp. 74–81, Feb. 2005.

[15] L. Lao, J.-H. Cui, and M. Gerla, "Multicast service overlay design," in *Proc. SPECTS 05*, Philadelphia, PA, USA, Jul. 2005.

[16] L. Qiu, V. N. Padmanabhan, and G. M. Voelker, "On the placement of web server replicas," in *Proc. IEEE INFOCOM 01*, vol. 3, Anchorage, AK, USA, Apr. 2001, pp. 1587–1596.

[17] B. Krishnamurthy and J. Wang, "On network-aware clustering of web clients," in *Proc. ACM SIGCOMM 00*, Stockholm, Sweden, Aug. 2000, pp. 97–110.

[18] J. Martin and A. Nilsson, "On service level agreements for IP networks," in *Proc. IEEE INFOCOM 02*, vol. 2, Jun. 2002, pp. 855–863.

[19] F. Safaei, I. Ouveysi, M. Zukerman, and R. Pattie, "Carrier-scale programmable networks: wholesaler platform and resource optimization," *IEEE J. Select. Areas Commun.*, vol. 19, no. 3, pp. 566–573, Mar. 2001.

[20] K. P. Gummadi, S. Saroiu, and S. D. Gribble, "King: Estimating latency between arbitrary internet end hosts," in *Proc. ACM IMW 02*, Marseille, France, Nov. 2002, pp. 5–18.

[21] H. V. Madhyastha, T. Anderson, A. Krishnamurthy, N. Spring, and A. Venkataramani, "A structural approach to latency prediction," in *Proc. ACM IMC 06*, Rio de Janeiro, Brazil, Oct. 2006, pp. 99–104.

[22] R. Diestel, *Graph Theory*, 3rd ed. New York: Springer-Verlag, 2005.

[23] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley, 1989.

[24] S.-M. Soak, D. W. Corne, and B.-H. Ahn, "The edge-window-decoder representation for tree-based problems," *IEEE Trans. Evol. Comput.*, vol. 10, no. 2, pp. 124–144, Apr. 2006.

[25] G. R. Raidl and B. A. Julstrom, "A weighted coding in a genetic algorithm for the degree-constrained minimum spanning tree problem," in *Proc. ACM Symp. Applied Computing*, Como, Italy, Mar. 2000, pp. 440–445.

[26] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer Methods in Applied Mechanics and Engineering*, vol. 186, no. 2–4, pp. 311–338, Jun. 2000.

[27] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *Proc. IEEE INFOCOM 96*, vol. 2, San Francisco, CA, USA, Mar. 1996, pp. 594–602.

[28] ILOG, http://www.ilog.com/, 2007.

[29] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, and D. Towsley, "Inferring TCP connection characteristics through passive measurements," in *Proc. IEEE INFOCOM 04*, vol. 3, Mar. 2004, pp. 1582–1592.

[30] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness.* San Francisco: W. H. Freeman, 1979.

[31] S. Sahni and T. Gonzalez, "P-complete approximation problems," *J. ACM*, vol. 23, no. 3, pp. 555–565, Jul. 1976.