Construction of an Efficient Overlay Multicast Infrastructure for Real-time Applications

Suman Banerjee, Christopher Kommareddy, Koushik Kar, Bobby Bhattacharjee, Samir Khuller
University of Maryland, College Park, MD 20742, USA

***Abstract*—We consider an overlay architecture where service providers deploy a set of service nodes (called MSNs) in the network to efficiently implement media-streaming applications. These MSNs are organized into an overlay and act as application-layer multicast forwarding entities for a set of clients.**

We present a decentralized scheme that organizes the MSNs into an appropriate overlay structure that is particularly beneficial for real-time applications. We formulate our optimization criterion as a "degree-constrained minimum average-latency problem" which is known to be NP-Hard. A key feature of this formulation is that it gives a dynamic priority to different MSNs based on the size of its service set.

Our proposed approach iteratively modifies the overlay tree using localized transformations to adapt with changing distribution of MSNs, clients, as well as network conditions. We show that a centralized greedy approach to this problem does not perform quite as well, while our distributed iterative scheme efficiently converges to near-optimal solutions.

I. Introduction

In this paper we consider a two-tier infrastructure to efficiently implement large-scale media-streaming applications on the Internet. This infrastructure, which we call the *Overlay Multicast Network Infrastructure (OMNI),* consists of a set of devices called Multicast Service Nodes (MSNs [13]) distributed in the network and provides efficient data distribution services to a set of end-hosts [1]. An end-host (client) subscribes with a single MSN to receive multicast data service. The MSNs themselves run a distributed protocol to organize themselves into an overlay which forms the multicast data delivery backbone. The data delivery path from the MSN to its clients is independent of the data delivery path used in the overlay backbone, and can be built using network layer multicast application-layer multicast, or a sequence of direct unicasts. The two-tier OMNI architecture is shown in Figure 1.

In this paper, we present a distributed iterative scheme that constructs "good" data distribution paths on the OMNI. Our scheme allows a multicast service provider to deploy a large number of MSNs without explicit concern about optimal placement. Once the capacity constraints of the MSNs are specified, our technique organizes them into an overlay topology, which is continuously adapted with changes in the distribution of the clients as well as changes in network conditions.

[1] Similar models of overlay multicast have been proposed in the literature (e.g. Scattercast [7] and Overlay Multicast Network [13]).

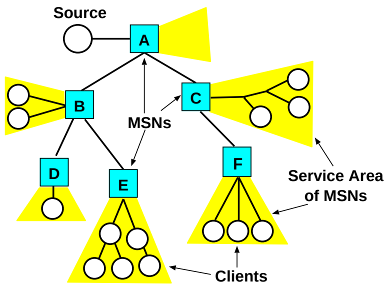

Fig. 1. OMNI Architecture.

Our proposed scheme is most useful for latency-sensitive real-time applications, such as media-streaming. Media streaming applications have experienced immense popularity on the Internet. Unlike static content, real-time data cannot be pre-delivered to the different distribution points in the network. Therefore an efficient data delivery path for real-time content is crucial for such applications. The quality of media playback typically depends on two factors: access loads experienced by the streaming server(s) and jitter experienced by the traffic on the end-to-end path. Our proposed OMNI architecture addresses both these concerns as follows: (1) being based on an overlay architecture, it relieves the access bottleneck at the server(s), and (2) by organizing the overlay to have low-latency overlay paths, it reduces the jitter at the clients.

For large scale data distributions, such as live webcasts, we assume that there is a single source. The source is connected to a single MSN, which we call the root MSN. The problem of efficient OMNI construction is as follows:

> Given a set of MSNs with access bandwidth constraints distributed in the network, construct a multicast data delivery backbone such that the

overlay latency to the client set is minimized.

Since the goal of OMNIs is to minimize the latencies to the entire client set, MSNs that serve a larger client population are, therefore, more important than the ones which serve only a few clients. The relative importance of the corresponding MSNs vary, as clients join and leave the OMNI. This, in turn, affects the structure of the data delivery path of the overlay backbone. Thus, one of the important considerations of the OMNI is its ability to adapt the overlay structure based on the distribution of clients at the different MSNs.

Our overlay construction objective for OMNIs is related to the objective addressed in [14]. In [14] the authors propose a centralized greedy heuristic, called the Compact Tree algorithm, to minimize the maximum latency from the source (also known as the diameter) to an MSN. However the objective of this minimum diameter degree-bounded spanning tree problem does not account for the difference in the relative importance of MSNs depending on the size of the client population that they are serving. In contrast we formulate our objective as the minimum average-latency degree-bounded spanning tree problem which weights the different MSNs by the size of the client population that they serve. We propose an iterative *distributed* solution to this problem, which dynamically adapts the tree structure based on the relative importance of the MSNs. Additionally we show how our solution approach can be easily augmented to define an equivalent distributed solution for the minimum diameter degree-bounded spanning tree problem.

The rest of the paper is structured as follows: In the next section we formalize and differentiate between the definition of these problems. In Section III we describe our solution technique. In Section IV we study the performance of our technique through detailed simulation experiments. In Section V we discuss other application-layer multicast protocols that are related to our work. Finally, we present our conclusions in Section VI.

II. Problem Formulation

In this section we describe the network model and state our solution objectives formally. We also outline the practical requirements that our solution is required to satisfy.

The physical network consists of nodes connected by links. The MSNs are connected to this network at different points through access links.

The multicast overlay network is the network induced by the MSNs on this physical topology. It can be modeled as a complete directed graph, denoted by $G = (V, E)$, where V is the set of vertices and $E = V \times V$ is the set of edges. Each vertex in V represents an MSN. The directed edge from node i to node j in G represents the unicast path from MSN i to MSN j in the physical topology The latency of an edge $\langle i, j \rangle$ corresponds to the unicast path latency from MSN i to MSN j.

The data delivery path on the OMNI will be a directed spanning tree of G rooted at the source MSN, with the edges directed away from the root. Consider a multicast application in which the source injects traffic at the rate of B units per second. We will assume that the the capacity of any incoming or outgoing access link is no less than B. Let the outgoing access link capacity of MSN i be b_i. Then the MSN can send data to at most $d_i = \lfloor b_i/B \rfloor$ other MSNs. This imposes an out-degree bound at MSN i on the overlay tree of the OMNI [2].

The overlay latency $L_{i,j}$ from MSN i to MSN j is the summation of all the unicast latencies along the overlay path from i to j on the tree, T. The latency experienced by a client (attached to MSN i) consists of three parts: (1) the latency from the source to the root MSN, r, (2) the latency from the MSN i to itself, and (3) the overlay latency $L_{r,i}$ on the OMNI from MSN r to MSN i. The arrangement of the MSNs affects only the overlay latency component, and the first two components do not depend on the OMNI overlay structure. Henceforth, for each client we only consider the overlay latency $L_{r,i}$ between the root MSN and MSN i as part of our minimization objective in constructing the OMNI overlay backbone.

We consider two separate objectives. Our first objective is to minimize is the average (or total) overlay latency of all clients. Let c_i be the number of clients that are served by MSN i. Then minimizing the average latency over all clients translates to minimizing the weighted sum of the latencies of all MSNs, where c_i denote the MSN weights.

The second objective is to minimize the maximum overlay latency for all clients. This translates to minimizing the maximum of the overlay latency of all MSNs. Let S denote the set of all MSNs other than the source. Then the two problems described above can be stated as follows:

P1: *Minimum average-latency degree-bounded directed spanning tree problem:* Find a directed spanning tree, T of G rooted at the MSN, r, satisfying the degree-constraint at each node, such that $\sum_{i \in S} c_i L_{r,i}$ is minimized.

P2: *Minimum maximum-latency degree-bounded directed spanning tree problem:* Find a directed spanning tree, T of G rooted at the MSN, r, satisfying the degree-constraint at each node, such that $\max_{i \in S} L_{r,i}$ is minimized.

[2] Internet measurements have shown that links in the core networks are over-provisioned, and therefore are not bottlenecks [3].

The minimum average-latency degree-bounded directed spanning tree problem, as well as the minimum maximum-latency degree-bounded directed spanning tree problem, are NP-hard [4], [14]. For brevity, in the rest of this paper, we will refer to these problems as the *min avg-latency problem* and the *min max-latency problem*, respectively. We focus on the min avg-latency problem because we believe that by weighting the overlay latency costs by the number of clients at each MSN, this problem better captures the relative importance of the MSNs in defining the overlay tree. In this paper we describe an iterative heuristic approach that can be used to solve the min avg-latency problem. In the solution description we also briefly highlight the changes necessary to our distributed solution to solve the min max-latency problem that has been addressed in prior work [14].

The development of the our approach is motivated by the following set of desirable features that make the solution scheme practical.

Decentralization: We require a solution to be to implementable in a distributed manner. It is possible to think of a solution where the information about the client sizes of the MSNs and the unicast path latencies are conveyed to a single central entity, which then finds a "good" tree (using some algorithm), and then directs the MSNs to construct the tree obtained. However, the client population can change dynamically at different MSNs which would require frequent re-computation of the overlay tree. Similarly, changes in network conditions can alter latencies between MSNs which will also incur tree re-computation. Therefore a centralized solution is not practical for even a moderately sized OMNI.

Adaptation: The OMNI overlay should adapt to changes in network conditions and changes in the distribution of clients at the different MSNs.

Feasibility: The OMNI overlay should adapt the tree structure by making incremental changes to the existing tree. However at any point in time the tree should satisfy all the degree constraints at the different MSNs. Any violation of degree constraint would imply an interruption of service for the clients. Therefore, as the tree adapts its structure towards an optimal solution using a sequence of optimization steps, none of the transformations should violate the degree constraints of the MSNs.

Our solution, as described in the next section, satisfies all the properties stated above.

III. Solution

In this section we describe our proposed distributed iterative solution to the problem described in Section II that meets all of the desired objectives. In this solution description, we focus on the min avg-latency problem and only point out relevant modifications needed for the min max-latency problem.

A. State at MSNs

For an MSN i, let $\mathit{Children}(i)$ indicate the set of children of i on the overlay tree and let c_i denote the number of clients being directly served by i. We use the term *aggregate subtree clients* (S_i) at MSN i to denote the entire set of clients served by all MSNs in the subtree rooted at i. The number of such aggregate subtree clients, $s_i = |S_i|$ is given by:

$$s_i = c_i + \sum_{j \in \mathit{Children}(i)} s_j$$

For example in Figure 1, $s_F = 3$, $s_E = 5$, $s_D = 1$, $s_C = 6$, $s_B = 8$, and $s_A = 14$. We also define a term called *aggregate subtree latency* (Λ_i) at any MSN, i, which denotes the summation of the overlay latency of each MSN in the subtree, from MSN i which is weighted by the number of clients at that MSN. This can be expressed as:

$$\Lambda_i = \begin{cases} 0 & \text{if } i \text{ is a leaf MSN} \\ \sum_{j \in \mathit{Children}(i)} s_j l_{i,j} + \Lambda_j & \text{otherwise} \end{cases}$$

where, $l_{i,j}$ is the unicast latency between MSNs i and j. In Figure 1, assuming all edges between MSNs have unit unicast latencies, $\Lambda_F = \Lambda_E = \Lambda_D = 0, \Lambda_C = 3, \Lambda_B = 6$, and $\Lambda_A = 23$. The optimization objective of the min avg-latency problem is to minimize the average subtree latency of the root, $\bar{\Lambda}_r$, (also called the average tree latency) [3].

Each MSN i keeps the following state information:

- *The overlay path from the root to itself:* This is used to detect and avoid loops while performing optimization transformations.
- *The value, s_i, representing the number of aggregate subtree clients.*
- *The aggregate subtree latency:* This is aggregated on the OMNI overlay from the leaves to the root.
- *The unicast latency between itself and its tree neighbors:* Each MSN periodically measures the unicast latency to all its neighbors on the tree.

Each MSN maintains state for all its tree neighbors and all its ancestors in the tree. If the minimum out-degree

[3] The maximum subtree latency, $\lambda_i^{\max}$ at an MSN, i, is the overlay latency from i to another MSN j which has the maximum overlay latency from i among the MSNs in the subtree rooted at i, i.e. $\lambda_i^{\max} = \max\{L_{i,j} | j \in \mathit{Subtree(i)}\}$. The optimization objective of the min max-latency problem is to minimize the maximum subtree latency of the root.

Procedure : *CreateInitialTree*(r, S)
SortedS $\leftarrow$ Sort S in increasing order of dist. from r
{ *Assert:* SortedS$[1] = r$ }
$i \leftarrow 1$
for $j \leftarrow 2$ to N **do**
while *SortedS*$[i]$.*NumChildren* = *SortedS*$[i]$.*DegBd*
$i++$
end while
SortedS$[j]$.*Parent* $\leftarrow$ *SortedS*$[i]$
SortedS$[i]$.*NumChildren*$++$
end for

Fig. 2. Initial tree creation algorithm for the initialization phase. r is the root MSN, S is an array of all the other MSNs and N is the number of MSNs.

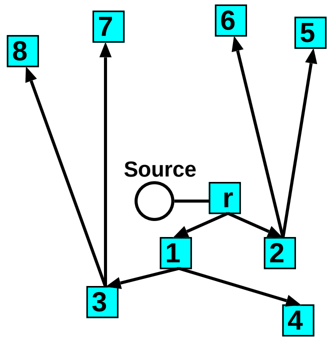

Fig. 3. Initialization of the OMNI using Procedure *CreateInitialTree*. r is the root MSN of the tree. The remaining MSNs are labeled in the increasing order of unicast latencies from r. In this example, we assume that each MSN has a maximum out-degree bound of two.

bound of an MSN is two, then it maintains state for at most $O(\textit{degree} + \log N)$ other MSNs.

We decouple our proposed solution into two parts — an initialization phase followed by successive incremental refinements. In each of these incremental operations no global interactions are necessary. A small number of MSNs interact with each other in each transformation to adapt the tree so that the objective function improves.

B. Initialization

In a typical webcast scenario data distribution is scheduled to commence at a specific time. Prior to this instant the MSNs organize themselves into an initial data delivery tree. Note that the clients of the different MSNs join and leave dynamically. Therefore no information about the client population sizes is available *a priori* at the MSNs during the initialization phase.

Each MSN that intends to join the OMNI measures the unicast latency between itself and the root MSN and sends a *JoinRequest* message to the root MSN. This message contains the tuple $\langle$*LatencyToRoot*, *DegreeBound*$\rangle$. The root MSN gathers *JoinRequests* from all the different MSNs, creates the initial data delivery tree using a simple centralized algorithm, and distributes it to the MSNs.

This centralized initialization procedure is described in pseudo-code in Figure 2. We describe this operation using the example in Figure 3. In this example, all MSNs have a maximum out-degree bound of two. The root, r, sorts the list of MSNs in an increasing order of distance from itself. It then fills up the available degrees of MSNs in this increasing sequence. It starts with itself and chooses the next closest MSNs (1 and 2) to be its children. It next chooses its closest MSN (1) and assigns MSNs 3 and 4 (the next closest MSNs with unassigned parents) as its children. Continuing this process, the tree shown in Figure 3 is constructed.

The centralized algorithm guarantees the following (see proof in the Appendix):

> If the triangle inequality holds on the overlay and if the degree bound of each MSN is at least 2, then overlay latency from the root MSN to *any* other MSN, i, is bounded by $2\ l_{r,i} \log N$, where N is the number of MSNs in the OMNI, and $l_{r,i}$ is the direct unicast latency between the root MSN, r, and MSN i.

The centralized computation of this algorithm is acceptable because it operates off-line before data delivery commences. An optimal solution to the min avg-latency problem is NP-Hard and would typically require $O(N^2)$ latency measurements (i.e. between each pair of MSNs). In contrast, the centralized solution provides a reasonable latency bound using only $O(N)$ latency measurements (one between each MSN and the root MSN). Note that the $\log N$ approximation bound is valid for each MSN. Therefore this initialization procedure is able to guarantee a $\log N$ approximation for both the min avg-latency problem as well as the min max-latency problem.

The initialization procedure, though oblivious of the distribution of the clients at different MSNs, still creates a“good” initial tree. This data delivery tree will be continuously transformed through local operations to dynamically adapt with changing network conditions (i.e. changing latencies between MSNs) and changing distribution of clients at the MSNs. Additionally new MSNs can join and existing MSNs can leave the OMNI even after data delivery commences. Therefore the initialization phase is optional for the MSNs, which can join the OMNI, even after the initialization procedure is done.

C. Local Transformations

We define a local transformation as one which requires interactions between nearby MSNs on the overlay tree. In

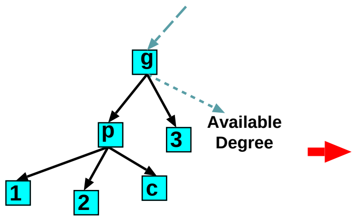

Fig. 4. Child-Promote operation. g is the grand-parent, p is the parent and c is the child. The maximum out-degree of all MSNs is three. MSN c is promoted in this example.

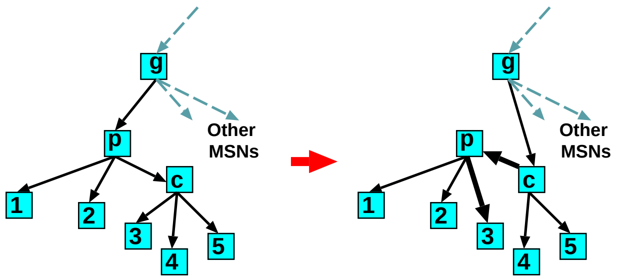

Fig. 5. Parent-Child Swap operation. g is the grand-parent, p is the parent and c is the child. Maximum out-degree is three.

particular these MSNs are within two levels of each other. We define five such local transformation operations that are permissible at any MSN of the tree. Each MSN periodically attempts to perform these operations. This period is called the *transformation period* and is denoted by τ. The operation is performed if it reduces the average-latency of the client population.

Child-Promote: If an MSN g has available degree, then one of its grand-children (e.g. MSN c in Figure 4) is promoted to be a direct child of g if doing so reduces the aggregate subtree latency for the min avg-latency problem. This is true if:

$$(l_{g,c} - l_{g,p} - l_{p,c})s_c < 0$$

For the min max-latency problem, the operation is performed only if it reduces the maximum subtree latency at g which can be verified by testing the same condition as above.

If the triangle inequality holds for the unicast latencies between the MSNs, this condition will always be true. If multiple children of p are eligible to be promoted, a child which maximally reduces the aggregate (maximum) subtree latency for the min avg-latency (min max-latency) problem is chosen.

Parent-Child Swap: In this operation the parent and child are swapped as shown in Figure 5. Note grand-parent, g is the parent of c after the transformation and c is the parent of p. Additionally one child of c is transferred to p. This is done if and only if the out-degree bound of c gets violated by the operation (as in this case). Note that in such a case only one child of c would need to be transferred and p would always have an available degree (since the transformation frees up one of its degrees). The swap operation is performed for the min avg-latency (min max-latency) problem if and only if the aggregate (maximum) subtree latency at g reduces due to the operation. Like the previous case, if multiple children of p are eligible for the swap operation, a child which maximally reduces the aggregate (maximum) subtree latency for the min avg-latency (min max-latency) problem is chosen.

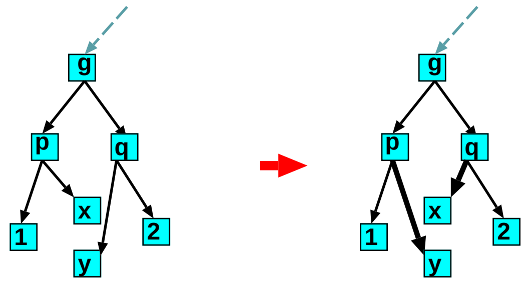

Fig. 6. Iso-level-2 Swap operation. g is the grand-parent, p and q are siblings. x and y are swapped.

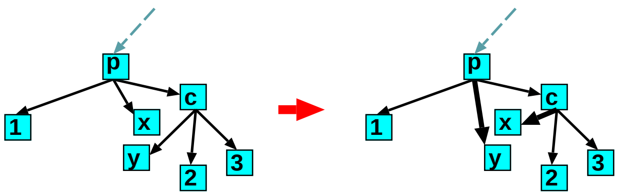

Fig. 7. Aniso-level-1-2 Swap operation. p is the parent of c. x and y are swapped.

Iso-level-2 Swap: We define an iso-level operation as one in which two MSNs at the same level swap their positions on the tree. Iso-level-k denotes a swap where the swapped MSNs have a common ancestor exactly k levels above. Therefore, the iso-level-2 operation defines such a swap for two MSNs that have the same grand-parent. As before, this operation is performed for the min avg-latency (min max-latency) problem between two MSNs x and y if and only if it reduces the aggregate (maximum) subtree latency (e.g. Figure 6).

Iso-level-2 Transfer: This operation is analogous to the previous operation. However, instead of a swap, it performs a transfer. For example, in Figure 6, Iso-level-2 transfer would only shift the position of MSN x from child of p to child of q. MSN y does not shift its position. This operation is only possible if q has available degree.

Aniso-level-1-2 Swap: An aniso-level operation involves two MSN that are not on the same level of the over-

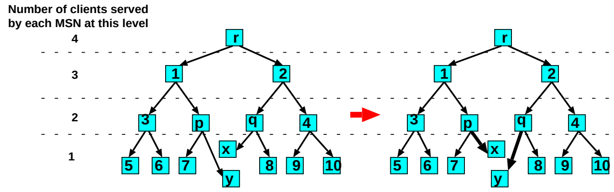

Fig. 8. Example where the five local operations cannot lead to optimality in the min avg-latency problem. All MSNs have maximum out-degree bound of two. r is the root. Arrow lengths indicate the distance between MSNs.

lay tree. An aniso-level-i-j operation involves two MSNs x and y for which the ancestor of x, i levels up, is also the ancestor of y, j levels up. Therefore the defined swap operation involves two MSNs x and y where the parent of x is the same as the grand-parent of y (as shown in Figure 7). The operation is performed if and only if it reduces the aggregate (maximum) subtree latency at p for the min avg-latency (min max-latency) problem.

Following the terminology as described, the *Child-Promote* operation is actually the *Aniso-level-1-2 transfer* operation.

D. Probabilistic Transformation

Each of the defined local operations reduce the aggregate (maximum) subtree latency on the tree for the min avg-latency (min max-latency) problem. Performing these local transformations will guide the objective function towards a local minimum. However, as shown in the example in Figure 8, they alone cannot guarantee that a global minimum will be attained. In the example, the root MSN supports 4 clients. MSNs in level 1 (i.e. 1 and 2) support 3 clients each, MSNs in level 2 support 2 clients each and MSNs in level 3 support a single client each. The arrow lengths indicate the unicast latencies between the MSNs. Initially $l_{p,y} + l_{q,x} < l_{p,x} + l_{q,y}$ and the tree as shown in the initial configuration was formed. The tree in the initial configuration was the optimal tree for our objective function. Let us assume that due to changes in network conditions (i.e., changed unicast latencies) we now have $l_{p,y} + l_{q,x} > l_{p,x} + l_{q,y}$. Therefore the objective function can now be improved by exchanging the positions of MSNs x and y in the tree. However, this is an iso-level-3 operation, and is not one of the local operations. Additionally it is easy to verify that any local operation to the initial tree will increase the objective function. Therefore no sequence of local operation exists that can be applied to the initial tree to reach the global minima.

Therefore we define a probabilistic transformation step that allows MSNs to discover such potential improvements to the objective function and eventually converge to the global minima. In each transformation period, τ, an MSN will choose to perform a probabilistic transformation with a low probability, p_{rand}.

If MSN i chooses to perform a probabilistic transformation in a specific transformation period, it first discovers another MSN, j, from the tree that is not its descendant. This discovery is done by a random-walk on the tree, a technique proposed in Yoid [9]. In this technique, MSN i transmits a *Discover* message with a *time-to-live* (TTL) field to its parent on the tree. The message is randomly forwarded from neighbor to neighbor, without re-tracing its path along the tree and the TTL field is decremented at each hop. The MSN at which the TTL reaches zero is the desired random MSN.

Random Swap: We perform the probabilistic transformation only if i and j are not descendant and ancestor of each other. In the probabilistic transformation, MSNs i and j exchange their positions in the tree. For the min avg-latency (min max-latency) problem, let Δ denote the increase in the aggregate (maximum) subtree latency of MSN k which is the least common ancestor of i and j on the tree (in Figure 8, this is the root MSN, r). k is identified by the *Discover* message as the MSN where the message stops its ascent towards the root and starts to descend. For the min avg-latency problem, Δ can be computed as follows:

$$\Delta = (L'_{k,i} - L_{k,i})s_i + (L'_{k,j} - L_{k,j})s_j$$

where, $L'_{k,i}$ and $L'_{k,j}$ denote the latencies from k to i and j respectively along the overlay if the transformation is performed, and $L_{k,i}$ and $L_{k,j}$ denotes the same prior to the transformation. Each MSN maintains unicast latency estimates of all its neighbors on the tree. The *Discover* message aggregates the value of $L_{k,j}$ on its descent from k to j from these unicast latencies. Similarly, a separate *Tree-Latency* message from k to i computes the value of $L_{k,i}$. (We use a separate message from k to i since we do not assume symmetric latencies between any pair of MSNs.) The L' values is computed from the L values and pair-wise unicast latencies between i, j and their parents. Thus, no global state maintenance is required for this operation.

We use a simulated annealing [2] based technique to probabilistically decide when to perform the swap operation. The swap operation is performed: (1) with a probability of 1 if $\Delta < 0$, and (2) with a probability $e^{-\Delta/T}$ if $\Delta \geq 0$, where T is the "temperature" parameter of the simulated annealing technique. In the min avg-latency (min max-latency) problem The swap operation is performed with a (low) probability even if the aggregate (maximum) subtree latency increases. This is useful in the search for a global optimum in the solution space. Note that the proba-

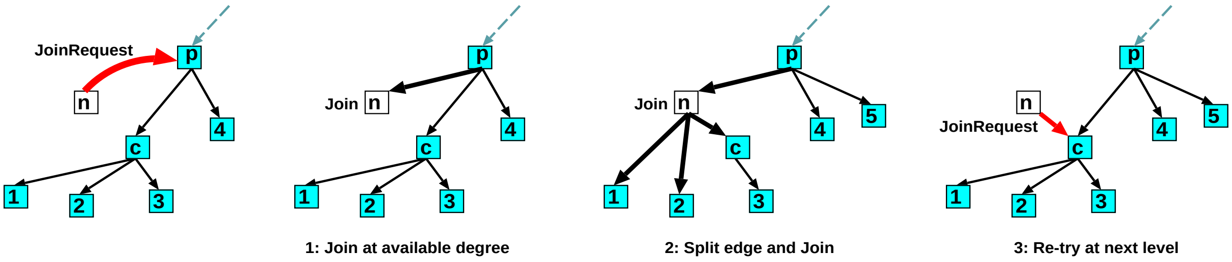

Fig. 9. Join operation for a new MSN. At each level there are three choices available to the joining MSN as shown. For each MSN, the maximum out-degree bound is 3.

bility of the swap gets exponentially smaller with increase in Δ.

E. Join and Leave of MSNs

In our distributed solution, we allow MSNs to arbitrarily join and leave the OMNI overlay. In this section, we describe both these operations in turn.

Join: A new MSN initiates its join procedure by sending the *JoinRequest* message to the root MSN. *JoinRequest* messages received after the initial tree creation phase invokes the distributed join protocol (as shown in Figure 9). At each level of the tree, the new MSN, n, has three options.

1) *Option 1:* If the currently queried MSN, p, has available degree, then n joins as its child. Some of the current children of c (i.e. 1 and 2) may later join as children of n in a later Iso-level-2 transfer operation.
2) *Option 2:* n chooses a child, c, of p and attempts to split the edge between them and join as the parent of c. Additionally some of the current children of c are shifted as children of n.
3) *Option 3:* n re-tries the join process from some MSN, c.

Option 1 has strict precedence over the other two cases. If option 1 fails, then we choose the lowest cost option between 2 and 3. The cost for option 2 can be calculated exactly through local interactions between n, p, c and the children of c. The cost of option 3 requires the knowledge of exactly where in the tree n will join. Instead of this exact computation, we compute the cost of option 3 as the cost incurred if n joins as a child of c. This leads to some inaccuracy which is later handled by the cost-improving local and probabilistic transformations.

Leave: If the leaving MSN is a leaf on the overlay tree, then no further change to the topology is required [4]. Otherwise, one of the children of the departing MSN is promoted up the tree to the position occupied by the departing MSN. We show this with an example in Figure 10. When MSN 3 leaves, one of its children (4 in this case) is promoted. For the min avg-latency (min max-latency) problem the child is chosen such that the aggregate (maximum) subtree latency is reduced the most. The other children of the departing MSN join the subtree rooted at the newly promoted child. For example, 5 attempts to join the subtree rooted at 4. It applies the join procedure described above starting from MSN 4, and is able to join as a child of MSN 7.

[4] The clients of the leaving MSNs need to be re-assigned to some other MSN, but that is an orthogonal issue to OMNI overlay construction.

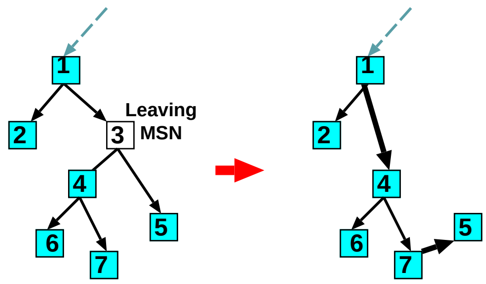

Fig. 10. Leave operation of an MSN. The maximum out-degree of each MSN is two.

Note that MSNs are specially managed infrastructure entities. Therefore it is expected that their failures are rare and most departures from the overlay will be voluntary. In such scenarios the overlay will be appropriately re-structured before the departure of an MSN takes effect.

IV. Simulation Experiments

We have studied the performance of our proposed distributed scheme through detailed simulation experiments. Our network topologies for these experiments were generated using the Transit-Stub graph model of the GT-ITM topology generator [5]. All topologies in these simulations had $10,000$ nodes (representing network routers) with an average node degree between 3 and 4. MSNs were attached to a set of these routers, chosen uniformly at random. As a consequence unicast latencies between different pairs of MSNs varied between 1 and 200 ms. The num-

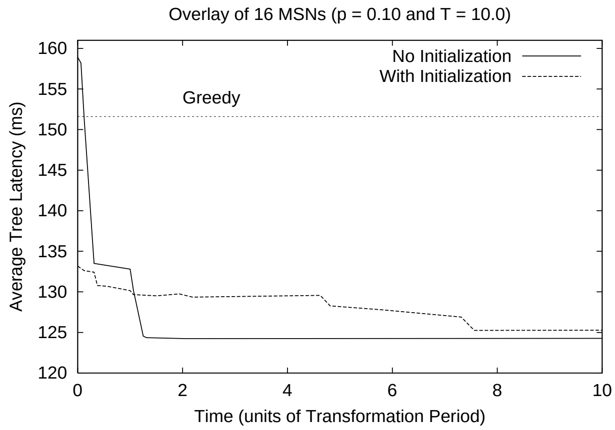

Fig. 11. Effect of the initialization phase (16 MSNs).

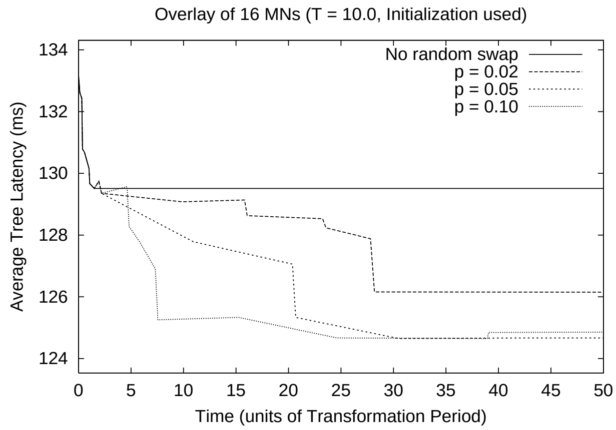

Fig. 12. Varying the probability of performing the random-swap operation for the different MSNs (16 MSNs).

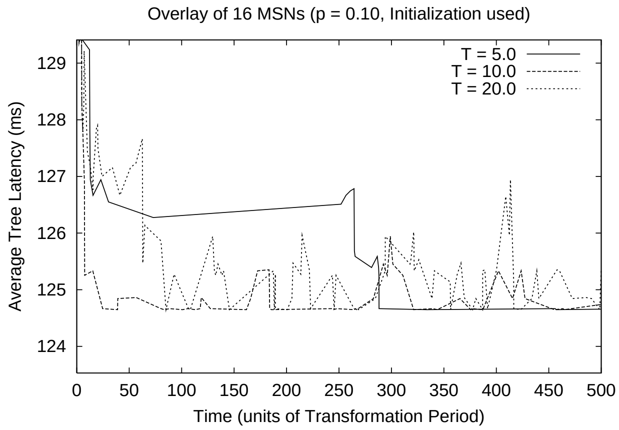

Fig. 13. Varying the temperature parameter for simulated-annealing (16 MSNs).

ber of MSNs was varied between 16 and 512 for different experiments.

In our experiments we compare the performance of our distibuted iterative scheme to these other schemes:

- The optimal solution: We computed the optimal value of the problem by solving an Integer Program (IP) using the CPLEX tool [5]. We describe the formulation of this IP in the Appendix. Computation of the optimal value using an IP requires a search over a $O(M^N)$ solution space, where M is the total number of clients and N is the number of MSNs. We were able to compute the optimal solution for networks with upto 100 clients and 16 MSNs.
- A centralized greedy heuristic solution: This heuristic is a simple variant of the Compact Tree algorithm proposed in [14]. It incrementally builds a spanning tree from the root MSN, r. For each MSN v that is not yet in the partial tree T, we maintain an edge $e(v) = \{u, v\}$ to an MSN u in the tree; u is chosen to minimize a cost metric $\delta(v) = (L_{r,u} + l_{u,v})/c_v$ where, $L_{r,u}$ is the overlay latency from the root of the partial tree to u and c_v is the number of clients being served by v. At each iteration we add one MSN (say v) to the partial tree which has minimum value for $\delta(v)$. Then for each MSN w not in the tree, we update $e(w)$ and $\delta(w)$.
  The centralized greedy heuristic proposed in [14] addresses the min max-latency problem. Our simple modification to that algorithm only changes the cost metric and is the equivalent centralized greedy heuristic for the min avg-latency problem as described in Section II.

[5] Available from http://www.ilog.com.

A. Convergence

We first present convergence properties of our solution for OMNI overlay networks. Figures 11, 12 and 13 show the evolution of the average tree latency, $\bar{\Lambda}_r$, (our minimization objective) over time for different experiment parameters for an example network configuration consisting of 16 MSNs. The MSNs serve between 1 and 5 clients, chosen uniformly at random for each MSN. In these experiments the set of 16 MSNs join the OMNI at time zero. We use our distributed scheme to let these MSNs organize themselves into the appropriate OMNI overlay. The x-axis in these figures are in units of the transformation period parameter, τ, which specifies the average interval between each transformation attempt by the MSNs. The ranges of the axes in these plots are different, since we focus on different time scales to observe the interesting characteristics of these results.

Figure 11 shows the efficacy of the initialization phase. When none of the MSNs make use of the initialization

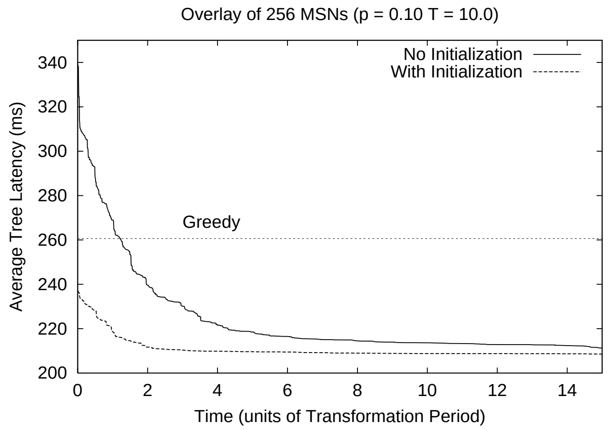

Fig. 14. Effect of the initialization phase (256 MSNs).

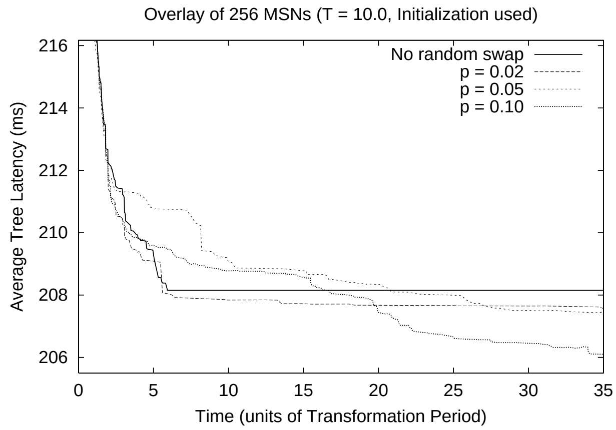

Fig. 15. Varying the probability of performing the random-swap operation for the different MSNs (256 MSNs).

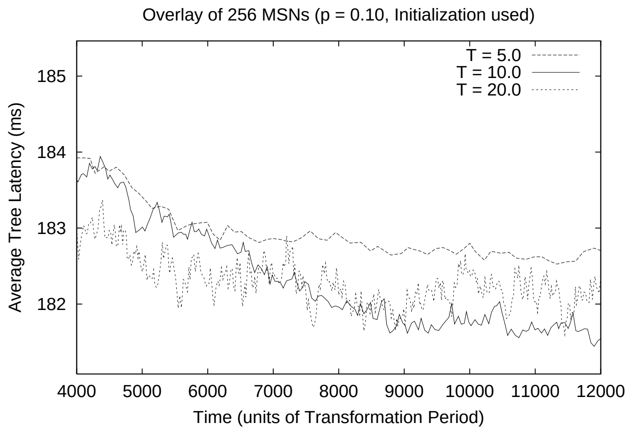

Fig. 16. Varying the temperature parameter for simulated annealing (256 MSNs).

phase, the initial tree has $\bar{\Lambda}_r = 158.92$ ms. In contrast, if the initialization phase is used by all MSNs, the initial tree has $\bar{\Lambda}_r = 133.18$ ms, a 16% reduction in cost. In both cases, however, the overlay quickly converges (within < 8 transformation periods) to a stable value of $\bar{\Lambda}_r \approx 124.5$ ms. The optimal value computed by the IP for this experiment was 113.96 ms. Thus, the cost of our solution is about 9% higher than the optimal. We ran different experiments for different network configurations and found that our distributed scheme converges to within $5 - 9\%$ of the optimum in all cases. A greedy approach to this problem does not work quite as well. The centralized greedy heuristic gives a solution with value 151.59 ms, and is about 21% higher than the converged value of the distributed scheme. In both these cases we had chosen the probability of a random-swap, p_{rand}, at the MSNs to be 0.1 and the T parameter of simulated-annealing to be 10.

In Figure 12 we show how the choice of p_{rand} affects the results. The initialization phase is used by MSNs for all the results shown in this figure. The local transformations occur quite rapidly and quickly reduces the cost of the tree for all the different cases. The $p_{rand} = 0$ case has no probabilistic transformations and is only able to reach a stable value of 129.51 ms. Clearly, once the objective reaches a local minimum it is unable to find a better solution that will take it towards a global minimum. As p_{rand} increases, the search for a global minimum becomes more aggressive and the objective function reaches the lower stable value rapidly. Figure 13 shows the corresponding plots for varying the T parameter. A higher T value in the simulated-annealing process implies that a random swap that leads to cost increment is permitted with a higher probability. For the moderate and high value of T (10 and 20), the schemes are more aggressive and hence the value of $\bar{\Lambda}_r$ experiences more oscillations. In the process both these schemes are aggressively able to find better solutions to the objective function. The oscillations are restricted to within 2% of the converged value.

Figures 14, 15, and 16 show the corresponding plots for experiments with 256 MSNs. Note that for the 256 MSN experiments, the best solution found by different choice of parameters has $\bar{\Lambda}_r = 181.53$ ms. Our distributed solution converges to this value after 7607 transformation period (τ) units. However, it converges to within 15% of the best solution within 5 transformation periods. Figure 16 shows the effect of the temperature parameter for the convergence. As before the oscillations are higher for higher temperatures, but are restricted to less than 1% of the converged value (the y-axis is magnified to illustrate the oscillations in this plot). This experiment also indicates that a greedy approach does not work well for this problem. The solution found by the greedy heuristic for this network configuration is 43% higher than the one found by our proposed technique.

Number of MSNs	Distributed Iterative Scheme	Centralized Greedy Scheme	Greedy/Iterative Ratio
16	**146.81**	174.32	1.17
32	**167.41**	231.64	1.34
64	**182.60**	258.88	1.40
128	**194.49**	291.44	1.49
256	**191.51**	289.67	1.51
512	**171.77**	262.94	1.53

TABLE I

COMPARISON OF THE BEST SOLUTION (IN MS) OF THE AVERAGE TREE LATENCY OBTAINED BY OUR PROPOSED DISTRIBUTED ITERATIVE SCHEME AND THE CENTRALIZED GREEDY HEURISTIC WITH VARYING OMNI SIZES, AVERAGED OVER 10 RUNS EACH.

We present a comparison of our scheme with the greedy heuristic in Table I. We observe that the performance of our proposed scheme gets progressively better than the greedy heuristic with increasing size of the OMNI overlay.

B. Adaptability

We next present results of the the adaptability of our distributed scheme for MSN joins and leaves, changes in network conditions and changing distribution of client populations.

MSNs join and leave: We show how the distributed scheme adapts the OMNI as different MSNs join and leave the overlay. Figure 17 plots the average tree latency for a join-leave experiment involving 248 MSNs. In this experiment, 128 MSNs join the OMNI during the initialization phase. Every 1500 transformation periods (marked by the vertical lines in the figure), a set of MSNs join or leave. For example, at time 6000, 64 MSNs join the OMNI and at time 7500, 64 MSNs leave the OMNI. These bulk changes to the OMNI are equivalent to a widespread network outage, e.g. a network partition. The other changes to the OMNI are much smaller, e.g. 8-32 simultaneous changes as shown in the figure. In each case, we let the OMNI converge before the next set of changes is effected. In all these changes the OMNI reaches to within 6% of its converged value of $\bar{\Lambda}_r$ within 5 transformation periods.

In Figure 18 we show the distribution of the number of transformations that happen in the first 10 transformation periods after a set of changes. (We only plot these distributions for 5 sets of changes — initial join of 128 MSNs, 8 MSNs join at time 1500, 64 MSNs join at time 6000, 64 MSNs leave at time 7500, and 8 MSNs leave at time 12000.) The bulk of the necessary transformations to converge to the best solution occur within the first 5 transformation periods after the change. Of these a vast majority (more than 97%) are due to local transformations.

These results suggest that the transformation period at the MSNs can be set to a relatively large value (e.g. 1 minute) and the OMNI overlay would still converge within a short time. It can also be set adaptively to a low value when the OMNI is experiencing a lot of changes for faster convergence and a higher value when it is relatively stable.

Changing client distributions and network conditions: A key aspect of the proposed distributed scheme is its ability to adapt to changing distribution of clients at the different MSNs. In Figure 19, we show a run from a sample experiment involving 16 MSNs. In this experiment, we allow a set of MSNs to join the overlay. Subsequently we varied the number of clients served by MSN x over time and observed its effects on the tree and the overlay latency to MSN x. The figure shows the time evolution of the relevant subtree fragment of the overlay.

In its initial configuration, the overlay latency from MSN 0 to MSN x is 59 ms. As the number of clients increases to 7, the importance of MSN x increases. It eventually changes its parent to MSN 4 (Panel 1), so that its overlay latency reduces to 54 ms. As the number of clients increases to 9, it becomes a direct child of the root MSN (Panel 2) with an even lower overlay latency of 51 ms. Subsequently the number of clients of MSN x decreases. This causes x to migrate down the tree, while other MSNs with larger client sizes move up. This example demonstrates how the scheme prioritizes the MSNs based on the number of clients that they serve.

We also performed similar experiments to study the effects of changing unicast latencies on the overlay structure. If the unicast latency on a tree edge between parent MSN x and one of its children, MSN y, goes up, the distributed scheme simply adapts the overlay by finding a better point of attachment for MSN y. Therefore, in one of our experiments, we picked an MSN directly connected to the root and increased its unicast latencies to all other MSNs (including the root MSN). A high latency edge close to the root affects a large number of clients. Therefore our distributed scheme adapted the overlay to reduce the average tree latency by moving this MSN to a leaf position in the tree, so that it cannot affect a large number of clients.

V. RELATED WORK

A number of other projects (e.g. Narada [8], NICE [1], Yoid [9], Gossamer [7],Overcast [10],ALMI [11], Scribe [6], Bayeux [15] multicast-CAN [12]) have explored implementing multicast at the application layer. However, in these protocols the end-hosts are considered to be equivalent peers and are organized into an appropriate overlay structure for multicast data delivery. In

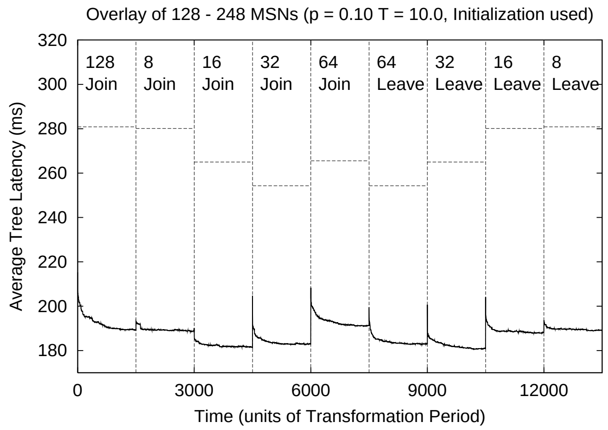

Fig. 17. Join leave experiments with 248 MSNs. The horizontal lines mark the solution obtained using the greedy heuristic.

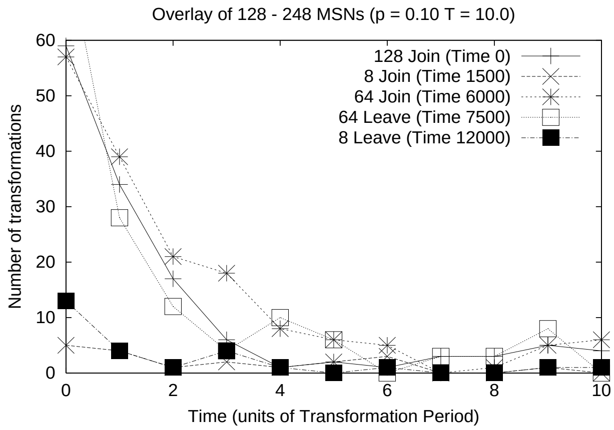

Fig. 18. Distribution of number of transformations in the first 10 transformation periods after a set of changes happen in the join leave experiment with 248 MSNs.

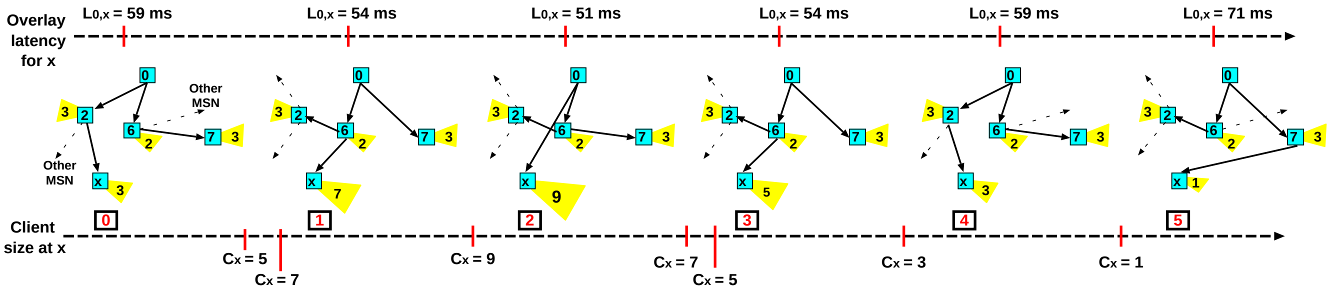

Fig. 19. Dynamics of the OMNI as number of clients change at MSNs (16 MSNs). MSN 0 is the root. MSNs 0, 2, and 6 had out-degree bound of 2 each and MSNs 7 and x had out-degree bound of 3 each. We varied the number of clients being served by MSN x. The relevant unicast latencies between MSNs are as follows: $l_{0,2} = 29$ ms, $l_{0,6} = 25$ ms, $l_{0,7} = 42$ ms, $l_{0,x} = 51$ ms, $l_{2,x} = 30$ ms, $l_{6,2} = 4$ ms, $l_{6,7} = 18$ ms, $l_{6,x} = 29$ ms, $l_{7,x} = 29$ ms. c_x indicates the number of clients at MSN x which changes with time. The time axis is not drawn to scale.

contrast, our work in this paper describes the OMNI architecture which is defined as a two-tier overlay multicast data delivery architecture.

An architecture similar to OMNI has also been proposed in [13] and their approach of overlay construction is related to ours. In [14] and [13] the authors proposed centralized heuristics to two related problems — minimum diameter degree-limited spanning tree and limited diameter residual-balanced spanning tree. The minimum diameter degree-limited spanning tree problem is same as the min max-latency problem. The focus of our paper is the min avg-latency problem, which better captures the relative importance of different MSNs based on the number of clients that are attached to them. In contrast to the centralized greedy solution proposed in [14], we propose an iterative distributed solution to the min avg-latency problem and show how it can be adapted to solve the min max-latency problem as well. Scattercast [7] defines another overlay-based multicast data delivery infrastructure, where a set of ScatterCast Proxies (SCXs) have responsibilities equivalent to the MSNs in the OMNI architecture. The SCXs organize themselves into a data delivery tree using the Gossamer protocol [7], which as mentioned before, does not organize the tree based on the relative importance of the SCXs. Clients register with these SCXs to receive multicast data.

VI. Conclusions

We have presented an iterative solution to the min avg-latency problem in the context of the OMNI architecture. Our solution is completely decentralized and each operation of our scheme requires interaction between only the affected MSNs. This scheme continuously attempts to improve the quality of the overlay tree with respect to our objective function. At each such operation, our scheme guarantees that the feasibility requirements, with respect to the MSN out-degree bounds, are met. Finally, our solution is adaptive and appropriately transforms the tree with join and leave operations of MSNs, changes in network conditions and distribution of clients at different MSNs.

REFERENCES

[1] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable application layer multicast. In *Proc. ACM Sigcomm*, Aug. 2002.

[2] D. Bertsekas. *Network Optimization: Continuous and Discrete Models*. Ahtena Scientific, 1998.

[3] S. Bhattacharyya, C. Diot, J. Jetcheva, and N. Taft. Pop-Level Access-Link-Level Traffic Dynamics in a Tier-1 POP. In *ACM Sigcomm Internet Measurement Workshop*, Nov. 2001.

[4] M. Blum, P. Chalasani, D. Coppersmith, B. Pulleyblank, P. Raghavan, and M. Sudan. The minimum latency problem. In *Proc. ACM Symposium on Theory of Computing*, May 1994.

[5] K. Calvert, E. Zegura, and S. Bhattacharjee. How to Model an Internetwork. In *Proc. IEEE Infocom*, 1996.

[6] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron. SCRIBE: A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in communications (JSAC)*, 2002. To appear.

[7] Y. Chawathe. Scattercast: An Architecture for Internet Broadcast Distribution as an Infrastructure Service. *Ph.D. Thesis, University of California, Berkeley*, Dec. 2000.

[8] Y.-H. Chu, S. G. Rao, and H. Zhang. A Case for End System Multicast. In *Proc. ACM Sigmetrics*, June 2000.

[9] P. Francis. Yoid: Extending the Multicast Internet Architecture, 1999. White paper http://www.aciri.org/yoid/.

[10] J. Jannotti, D. Gifford, K. Johnson, M. Kaashoek, and J. O'Toole. Overcast: Reliable Multicasting with an Overlay Network. In *Proc. 4th Symposium on Operating Systems Design and Implementation*, Oct. 2000.

[11] D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel. ALMI: An Application Level Multicast Infrastructure. In *Proc. 3rd Usenix Symposium on Internet Technologies & Systems*, March 2001.

[12] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Application-level multicast using content-addressable networks. In *Proc. 3rd International Workshop on Networked Group Communication*, Nov. 2001.

[13] S. Shi and J. Turner. Routing in overlay multicast networks. In *Proceedings of Infocom*, June 2002.

[14] S. Shi, J. Turner, and M. Waldvogel. Dimensioning server access bandwidth and multicast routing in overlay networks. In *Proceedings of NOSSDAV*, June 2001.

[15] S. Q. Zhuang, B. Y. Zhao, A. D. Joseph, R. Katz, and J. Kubiatowicz. Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination. In *11th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 2001)*, 2001.

APPENDIX

I: PROOF OF APPROXIMATION RATIO

Here we show that our initialization procedure (Section III-B) ensures that the overlay latency of any MSN is at most $2\log_2 N$ times the direct unicast latency of the MSN from the root MSN.

We assume that unicast latencies follow the triangle inequality. We also assume that unicast path latencies are symmetric, i.e., for any $\langle i,j\rangle \in E$, $l_{i,j} = l_{j,i}$.

Consider any MSN i in the OMNI constructed by our initialization procedure. Note that the MSNs were added in the increasing order of their unicast latencies from the root MSN, r. Therefore, for any MSN j that lies in the overlay path from r to i, $l_{r,j} \le l_{r,i}$. Thus for any two nodes j and k on the overlay path from r to i, $l_{j,k} \le l_{j,r} + l_{r,k} = l_{r,j} + l_{r,k} \le 2l_{r,i}$ (using symmetry and the triangle inequality). Let $E_i \subseteq E$ be the set of edges in the overlay path from r to i. Since the minimum out-degree of any MSN is two, it follows that $|E_i| \le \log_2 N$. Let $E_i \subseteq E$ be the set of edges on the overlay path from r to i. Thus $L_{r,i} = \sum_{(j,k)\in E_i} l_{j,k} \le 2l_{r,i}|E_i| \le 2l_{r,i}\log_2 N$.

II: INTEGER-PROGRAMMING FORMULATION

Here we present a linear integer programming formulation for the avg-latency problem, which can be used to solve the problem optimally using CPLEX. Developing a nonlinear integer programming formulation for this problem is not difficult. However, CPLEX is typically much more efficient in solving linear integer programs. In the formulation described below, the number of variables and constraints are also linear in the size of the OMNI.

For each edge $\langle i,j\rangle \in E$ in graph G, define two variables: a binary variable $x_{i,j}$, and a non-negative real (or integer) variable $f_{i,j}$, where $x_{i,j}$ denotes whether or note the edge $\langle i,j\rangle$ is included in the tree and $f_{i,j}$ denotes the number of clients which are served through edge $\langle i,j\rangle$.

Then the avg-latency problem can be formulated as:

$$\text{minimize} \quad \frac{1}{N}\sum_{\langle i,j\rangle\in E} l_{i,j} f_{i,j}$$

subject to

$$\begin{aligned}
\sum_{k\in V\setminus\{i\}} f_{k,i} - \sum_{k\in V\setminus\{i\}} f_{i,k} &= c_i \quad \forall i \in V\setminus\{r\} &(1)\\
0 \le f_{i,j} &\le C x_{i,j} \quad \forall\langle i,j\rangle \in E &(2)\\
\sum_{\langle i,j\rangle\in E} x_{i,j} &\le N-1 &(3)\\
x_{i,j} &\in \{0,1\} \quad \forall\langle i,j\rangle \in E &(4)
\end{aligned}$$

In Constraint 3 and in the objective function, N is the total number of MSNs. In Constraint 2, C is the total number of clients served by the OMNI. The objective function, as well as Constraint 1, follow from the definition of the variables $f_{i,j}$. Constraint 2 ensure that the variable $f_{i,j}$ is zero if $x_{i,j}$ is zero. Constraint 3 is necessary to enforce the tree structure of the OMNI overlay. All the contraints together ensure that the solution is a spanning tree rooted at r.