

Max-Min Overlay Multicast: Rate Allocation and Tree Construction

Yi Cui, Yuan Xue and Klara Nahrstedt

Department of Computer Science, University of Illinois at Urbana-Champaign
{yicui, xue, klara}@cs.uiuc.edu

Abstract—Although initially proposed as the deployable alternative to IP multicast, overlay multicast actually offers us great flexibilities on QoS-aware resource allocation for network applications. For example, in overlay multicast streaming, (1) the streaming rate of each client can be diversified to better accommodate network heterogeneity, through various end-to-end streaming adaptation techniques; and (2) one can freely organize the overlay session by rearranging the multicast tree, for the purpose of better resource utilization and fairness among all clients. The goal of this paper, is to find the max-min rate allocation in overlay multicast, which is pareto-optimal in terms of network resource utilization, and max-min fair. We approach this goal in two steps. First, we present a distributed algorithm, which is able to return the max-min rate allocation for any given overlay multicast tree. Second, we study the problem of finding the optimal tree, whose max-min rate allocation is optimal among all trees. After proving its NP-hardness, we propose a heuristic algorithm of overlay multicast tree construction. A variation of the heuristic is also designed to handle the dynamic client join/departure. Both of them have approximation bound of 1/2 to the optimal value. Experimental results show that they achieve high average throughput, almost saturate link utilization, and consistent min-favorability.

I. INTRODUCTION

Although initially proposed as a deployable alternative to IP multicast, overlay multicast[1] actually revolutionizes the way multicast-based applications can be built. In IP multicast, except for nodes at the edge, the network is composed of routers, whose task is no more than forwarding packets. In contrast, each node in the overlay network is an intelligent end host. This offers us great flexibilities on QoS-aware resource allocation in the overlay network, in order to better utilize network resource, meanwhile achieving certain fairness.

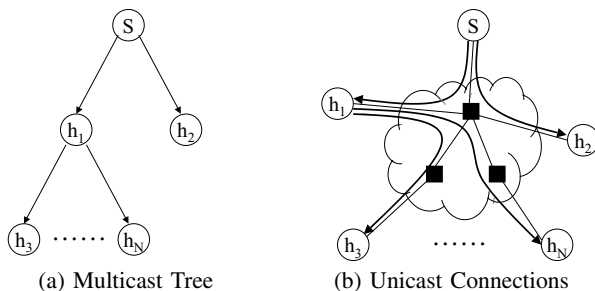


Fig. 1. Overlay-based Streaming Multicast

We illustrate such flexibilities via an example of overlay-based streaming multicast. As shown in Fig. 1, N end hosts

form an overlay multicast tree T rooted at server S . Each tree edge connecting two end hosts corresponds to their unicast path underneath.

First, the streaming rate of each client can be diversified, which better accommodates network heterogeneity. For example, h_2 can have a lower rate than h_1 , if h_2 's access link has lower capacity. This is also achievable in IP multicast through layered streaming[2][3]. In overlay multicast, however, besides the layered approach[4], all end-to-end streaming adaptation techniques (frame dropping, transcoding[5], etc.) can be applied, as all data relays happen via unicast. Such a flexibility enables the rate control in a much finer granularity than the layered approach.

Second, one can freely organize the overlay multicast session by rearranging the overlay multicast tree. For example, we can modify the tree T in Fig. 1 (a) by letting h_1 and h_2 switch their positions, if h_2 's access link has higher capacity than h_1 . In this way, h_3 can have a better chance to receive the stream at a higher rate from its parent.

The purpose of this work, is to study how we can utilize the above flexibilities to achieve optimal network resource utilization, while maintaining certain fairness among all end hosts. Formally, given an overlay multicast session, our goal is to find its max-min rate allocation, which is pareto-optimal in terms of network resource utilization, and max-min fair. We choose this model mainly because it is best suitable for heterogeneous network environment, which we envision to be the case for overlay network. When one faces the choice to increase the streaming rate of either but only one of two end hosts, choosing the low-end host can greatly enhance its streaming quality, hence its user's satisfaction. Otherwise, increasing the already high streaming rate for the high-end host can also enhance its user's satisfaction but the improvement will not be equally significant. Therefore, choosing max-min fairness can help maximize the overall satisfaction of all users.

However, achieving max-min resource allocation in overlay multicast is not a trivial task. The solution space to this problem is illustrated in Fig. 2. The "Max-min" axis represents the degree of max-min optimality of a rate allocation¹. Along the "Rate Allocation" axis, we can see that for any given overlay multicast tree, there exists a unique max-min rate

¹Two rate allocation vectors can be comparable, whose details can be found in Sec. III-A.

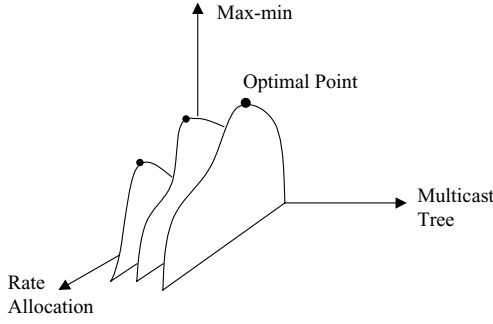


Fig. 2. Solution Space to Max-min Rate Allocation

allocation, which has the highest degree of optimality. Along the “Multicast Tree” axis, we can see there exists a unique optimal tree among all trees, whose max-min rate allocation has the highest degree of optimality, which is the unique optimal point of the entire space.

We explore this solution space in two steps. First, we present a max-min rate allocation algorithm for any given overlay multicast tree. The algorithm incurs very limited network measurement and messaging overhead, converges in only one iteration, and guarantees to return max-min rate allocation. Second, we study the problem of finding the optimal tree and prove it to be NP-hard. We then propose a heuristic algorithm of tree construction. A variation of the heuristic is also designed to handle the dynamic client join/departure. Both of them have approximation bound of 1/2, i.e., they are guaranteed to construct a multicast tree, whose max-min rate allocation has the minimum receiving rate at least 1/2 of the same value returned by the optimal tree. Experimentally, our algorithms greatly outperform this bound. Experimental results also show that under heterogeneous network environments, different multicast session scales, and dynamic client joins/leaves, the algorithm consistently achieves high average throughput, almost saturate link utilization, and consistent min-favorability.

The rest of the paper is organized as follows. Sec. II presents the network model. Sec. III presents the problem formulation. Sec. IV presents the algorithms. Sec. V presents experimental results. Finally, we review related works in Sec. VI and conclude in Sec. VII.

II. OVERLAY NETWORK MODEL

We consider an overlay network consisting of server S and N clients, denoted as $\mathcal{H} = \{h_1, \dots, h_N\}$. We then define the rate vector $\mathbf{x} = (x_1, \dots, x_N)$, where x_i is the streaming rate of $h_i (i = 1, \dots, N)$. For two end hosts h_k and h_i , if h_k relays the stream to h_i , then h_k is h_i 's parent, h_i is h_k 's child, denoted as $h_k \rightarrow h_i$. For example, in Fig. 1, $h_1 \rightarrow h_3$.

In this paper, we assume that the bottleneck of a unicast path only happens at either access link of its two end hosts, as shown in Fig. 3. Our assumption is based on the observation that the backbone links of today's Internet are usually over-provisioned[6][7]. In this way, we can view the cloud in Fig. 3

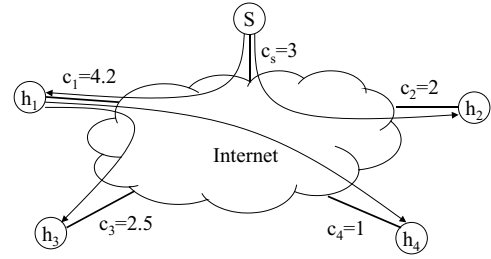


Fig. 3. Network Model

as a single node. The network topology is thus simplified as a *star topology*.

Now we define the capacity vector $\mathbf{c} = (c_S, c_1, \dots, c_N)$, where c_S is the access link capacity of the server S , c_i is the access link capacity of $h_i (i = 1, \dots, N)$. For any host, the sum of rates of all streams to and from it cannot exceed its access link capacity. Formally, it must satisfy that

$$\sum_{S \rightarrow h_i} x_i \leq c_S \quad (1)$$

$$x_k + \sum_{h_k \rightarrow h_i} x_i \leq c_k \quad (2)$$

which we refer to as *network constraint*. It can also be formalized in the following way.

$$\mathbf{A} \cdot \mathbf{x} \leq \mathbf{c} \quad (3)$$

\mathbf{A} is a $(N + 1) \times N$ matrix, such that

$$A_{li} = \begin{cases} 1 & \text{if } l = i \text{ or } h_l \rightarrow h_i \\ 0 & \text{otherwise} \end{cases}$$

Moreover, if $h_k \rightarrow h_i$, then $x_k \geq x_i$, i.e., as the child of h_k , h_i 's streaming rate cannot exceed its parent. We refer to such a constraint as *data constraint*. Note that if $S \rightarrow h_i$, this rule does not apply to x_i , since S is the root of the tree. We define a $N \times N$ matrix \mathbf{B} to formalize the data constraint.

$$B_{ik} = \begin{cases} -1 & \text{if } h_k \rightarrow h_i \\ 1 & \text{if } i = k \text{ and } h_i \text{'s parent is not } S \\ 0 & \text{otherwise} \end{cases}$$

Then it must satisfy that

$$\mathbf{B} \cdot \mathbf{x} \leq \mathbf{0} \quad (4)$$

We collect above notations into Tab. II. In the example by Fig. 3, there are 4 clients ($N = 4$). Hence, Inequality (3) becomes

$$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \leq \begin{pmatrix} 3 \\ 4.2 \\ 2 \\ 2.5 \\ 1 \end{pmatrix}$$

Inequality (4) becomes

$$\begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \leq \mathbf{0}$$

Notation	Definition
S	Server
$\mathcal{H} = \{h_i, i = 1, \dots, N\}$	End Hosts
$\mathbf{x} = (x_i, i = 1, \dots, N)$	Streaming Rate of h_i
$\mathbf{c} = (c_S, c_i, i = 1, \dots, N)$	Capacity of S and h_i 's Access Link
$S \rightarrow h_i$	S is the Parent of h_i
$h_k \rightarrow h_i$	h_k is the Parent of h_i
$\mathbf{A}_{(N+1) \times N}$	Network Constraint Matrix
$\mathbf{B}_{N \times N}$	Data Constraint Matrix

TABLE I
NOTATIONS IN SEC. II

Throughout this paper, we will use this example to illustrate our algorithms.

III. PROBLEM FORMULATION

A. Goal: Optimal and Fair Resource Allocation

In this work, we consider the problem of fair resource allocation in an overlay network while resource utilization is simultaneously maximized. We first introduce max-min, a commonly accepted fairness model. Consider the rate vector $\mathbf{x} = (x_i, i = 1 \dots N)$. We say \mathbf{x} is *ordered* if it satisfies that $x_i \leq x_{i+1} (1 \leq i < N)^2$. We have the following definitions and proposition, whose details can be found in [3].

Definition 1 (Min-favorability): Let $\mathbf{x} = (x_1, \dots, x_N)$ and $\mathbf{x}' = (x'_1, \dots, x'_N)$ be two ordered vectors. We say $\mathbf{x} \geq_m \mathbf{x}'$ (\mathbf{x} is *min-favorable* to \mathbf{x}') if no i exists such that $x'_i > x_i$, or for any i where $x'_i > x_i$, there is some $j < i$ where $x'_j < x_j$.

Min-favorability is similar to alphabetizing two text strings of the same length. Let x_i represent the i th character of the first string, and x'_i the i th character of the second string. Then $\mathbf{x} \geq_m \mathbf{x}'$ iff $\mathbf{x} = \mathbf{x}'$ or an alphabetization places \mathbf{x}' before \mathbf{x} .

Definition 2 (Max-min): An ordered vector \mathbf{x}^* is a max-min rate vector, if for any ordered vector \mathbf{x} , $\mathbf{x}^* \geq_m \mathbf{x}$. We denote a max-min rate vector as $\mathbf{x}^* = \mathbf{max}_m \{\mathbf{x}\}$.

The definition of max-min rate vector states that \mathbf{x}^* is min-favorable than any other rate vectors. This means that a rate allocation vector \mathbf{x}^* is max-min fair if the streaming rate x_i can not be improved without decreasing the rate of any other flow x_j , where $j < i$. Roughly, the definition states that the max-min fair allocation gives the most poorly treated user (i.e., the user who receives the lowest rate) the largest possible share, while not wasting any network resources. Further, we show that max-min rate allocation is Pareto optimal in resource utilization.

Definition 3: For two rate vectors \mathbf{x} and \mathbf{x}' , $\mathbf{x} \geq \mathbf{x}'$, if and only if for all $i = 1 \dots N$, $x_i \geq x'_i$.

²If not otherwise specified, a vector is assumed to be ordered thereafter in this paper.

Definition 4 (Pareto optimality). A rate vector $\mathbf{x}^* = (x_i^*, i = 1 \dots N)$ is *Pareto optimal*, if $\forall \mathbf{x}$, $\mathbf{x} \geq \mathbf{x}^*$, then $\mathbf{x} = \mathbf{x}^*$.

Proposition 1. A max-min rate vector \mathbf{x}^* is Pareto optimal.

Proof: If \mathbf{x}^* is the max-min rate vector, then for any ordered vector \mathbf{x} , $\mathbf{x}^* \geq_m \mathbf{x}$. This means that there exists no vector $\mathbf{x} > \mathbf{x}^*$, which establishes the result. ■

To this end, we formulate the optimal rate allocation of an overlay network as follows, where Inequality (6) and (7) are the network and data constraints of the overlay network.

$$\mathbf{max}_m \quad \mathbf{x} \quad (5)$$

$$\mathbf{subject\ to} \quad \mathbf{A} \cdot \mathbf{x} \leq \mathbf{c} \quad (6)$$

$$\mathbf{B} \cdot \mathbf{x} \leq \mathbf{0} \quad (7)$$

B. Solution Space

To achieve optimal and fair rate allocation, there are two dimensions in the solution space along which we can explore: rate allocation and overlay multicast tree construction. First, if the overlay multicast tree is given, max-min allocation goal can be achieved through rate allocation among different unicast flows. Second, different overlay multicast trees will result in different max-min rate allocations. Further, there exists one overlay multicast tree, whose max-min rate allocation is min-favorable to the max-min rate allocations of all the other trees.

1) *Rate Allocation:* First, when an overlay multicast tree T is given, the network and data constraint matrices $\mathbf{A}(T)$ and $\mathbf{B}(T)$ are determined. Thus the problem can be stated as follows.

$$\mathbf{R}(T) : \quad \mathbf{max}_m \quad \mathbf{x} \quad (8)$$

$$\mathbf{subject\ to} \quad \mathbf{A}(T) \cdot \mathbf{x} \leq \mathbf{c}$$

$$\mathbf{B}(T) \cdot \mathbf{x} \leq \mathbf{0}$$

$$\mathbf{over} \quad \mathbf{x} \geq \mathbf{0}$$

2) *Overlay Multicast Tree Construction:* Further, let \mathcal{T} be the set containing all multicast tree configurations. By Cayley's theorem [8], $|\mathcal{T}| = (N+1)^{N-1}$. For any tree $T \in \mathcal{T}$, solving $\mathbf{R}(T)$ will give a max-min rate allocation $\mathbf{x}^*(T)$. Among these max-min rate allocations, there exists one tree $T^\diamond \in \mathcal{T}$, whose max-min rate allocation $\mathbf{x}^*(T^\diamond)$ is the most min-favorable among all trees in \mathcal{T} . This problem can be formulated as follows.

$$\mathbf{T} : \quad \mathbf{max}_m \quad \mathbf{x} \quad (9)$$

$$\mathbf{subject\ to} \quad \mathbf{A}(T) \cdot \mathbf{x} \leq \mathbf{c}$$

$$\mathbf{B}(T) \cdot \mathbf{x} \leq \mathbf{0}$$

$$\mathbf{over} \quad \mathbf{x} \geq \mathbf{0}$$

$$T \in \mathcal{T}$$

Theorem 1: Problem \mathbf{T} is NP-hard. Proof is provided in the Appendix 1.

IV. RESOURCE ALLOCATION ALGORITHMS

A. A Max-Min Rate Allocation Algorithm

We present a distributed algorithm to Problem $\mathbf{R}(T)$, shown in Tab. II. The algorithm is distributed, in that it only depends on exchange of *Rate Report* and *Rate Update* messages between parent and children nodes in the tree. The algorithm starts from the leaf nodes, which report their initial streaming rates to their parents. Upon collecting all *rate reports* from its children, the parent node then calculates its own streaming rate, and sends the rate report further up, until the server S is reached. S then adjusts the streaming rates of its children according to the network constraint, and notifies them. Upon receiving the *rate update*, each child adjusts its streaming rate, then sends the rate update further to its children. This process is repeated until it reaches all leaf nodes. The algorithm only needs one such iteration to return the max-min rate allocation \mathbf{x}^* .

End Host h_k (h_k has M children h_1, \dots, h_M)	
<u>Initialization</u>	
1	if h_k has no children ($M = 0$)
2	$x_k \leftarrow c_k$
3	Report Rate x_k to its Parent
<u>On Receiving Rate Reports x_i from $h_i (i = 1, \dots, M)$</u>	
1	Sort h_1, \dots, h_M such that $x_1 \leq \dots \leq x_M$
2	$c \leftarrow c_k$
3	$i \leftarrow 1$
4	while $i \leq M$ and $\frac{c}{M+2-i} > x_i$
5	$c \leftarrow c - x_i$
6	$i \leftarrow i + 1$
7	$c \leftarrow \frac{c}{M+2-i}$
8	while $i \leq M$
9	$x_i \leftarrow c$
10	$i \leftarrow i + 1$
11	$x_k \leftarrow c$
12	Report Rate x_k to h_k 's Parent
<u>On Receiving Rate Update x_k from its Parent</u>	
1	for $h_i (i = 1, \dots, M)$
2	$x_i \leftarrow \min(x_i, x_k)$
3	Update Rate x_i to h_i
Server S (S has M children h_1, \dots, h_M)	
<u>On Receiving Rate Reports x_i from $h_i (i = 1, \dots, M)$</u>	
1	Sort h_1, \dots, h_M such that $x_1 \leq \dots \leq x_M$
2	$c \leftarrow c_k$
3	$i \leftarrow 1$
4	while $i \leq M$ and $\frac{c}{M+1-i} > x_i$
5	$c \leftarrow c - x_i$
6	$i \leftarrow i + 1$
7	$c \leftarrow \frac{c}{M+1-i}$
8	while $i \leq M$
9	$x_i \leftarrow c$
10	$i \leftarrow i + 1$
11	for $h_i (i = 1, \dots, M)$
12	Update Rate x_i to h_i

TABLE II

MAX-MIN RATE ALLOCATION ALGORITHM

We illustrate our algorithm using the sample topology in Fig. 3. At the beginning, the leaf nodes h_3 and h_4 initialize

their receiving rates as their own access link capacities, i.e., $x_3 = c_3 = 2.5$ and $x_4 = c_4 = 1$, and send the rate report messages to h_1 (Fig. 4 (a)). Upon receiving messages from h_3 and h_4 , h_1 first initializes the available bandwidth on its access link as $c = c_1 = 4.2$. Since there will be 3 streams going through h_1 's access link, h_1 equally divides c into 3 portions, $\frac{c}{3} = \frac{4.2}{3}$, and see if it is a feasible allocation for all the 3 streams. It turns out that $x_4 = 1 < \frac{4.2}{3}$, which indicates that h_4 's access link is the bottleneck. In this case, h_1 determines h_4 's rate as 1, and allocates the corresponding bandwidth from its own link. Now, h_1 's available bandwidth subsides to $c = c_1 - x_4 = 3.2$. Then h_1 equally divides c into 2 portions, $\frac{c}{2} = 1.6$, and see if it is a feasible allocation for the rest streams. Since $x_3 > 1.6$, h_1 allocates $\frac{c}{2}$ amount of bandwidth to h_3 , i.e., $x_3 = 1.6$, and declares the residual bandwidth as its own streaming rate, i.e., $x_1 = 1.6$. Note that this is the maximum rate h_1 can possibly have, since h_1 's link is already fully utilized at this point. Increasing x_1 will inevitably reduce x_3 or x_4 , which is already smaller than or equal to x_1 .

Now h_1 sends rate report message to its parent S , together with its sibling node h_2 (Fig. 4 (b)). h_2 initializes $x_2 = c_2 = 2$, since it is also a leaf node. Similar to h_1 , S first initializes the available bandwidth on its link as $c = c_S = 3$. Since S will stream to both of its children through its own link, it equally divides c into 2 portions, $\frac{c}{2} = 1.5$, and see if it is a feasible allocation for h_1 and h_2 . Since both x_1 and x_2 are greater than 1.5, S adjusts them such that $x_1 = x_2 = 1.5$. Finally, S notifies h_1 and h_2 about the updated rates (Fig. 4 (c)).

Upon receiving the rate update messages, h_1 and h_2 adjusts their own rates accordingly. h_1 further sends the message down to its children h_3 and h_4 (Fig. 4 (d)) to ensure that the data constraint is not violated. As a result, h_3 adjust its own rate from 1.6 to 1.5, so that x_3 will be no greater than x_1 , the rate of h_3 's parent. The algorithm finishes when all leaf nodes are done with the stream rate adjustment. The final rate vector is $\mathbf{x} = (1, 1.5, 1.5, 1.5)$.

Theorem 2: The Algorithm in Tab. II returns the max-min rate allocation.

Proof is provided in the Appendix 2. The salient features of our algorithm is as follows.

- 1) The message exchange (rate report/update) happens exactly once between each parent/child pair. Therefore, the messaging overhead is $2N$, N being the number of clients.
- 2) Since the message exchange can happen in parallel among sibling nodes, the algorithm takes time of $O(2t \cdot \text{height}(T))$, t being the longest messaging delay among all unicast connections in the tree.
- 3) Each end host only needs to measure the available bandwidth of its own access link.

B. A Multicast Tree Construction Heuristic

Given the NP-hardness of finding the optimal multicast tree (Problem \mathbf{T}), we propose a tree construction heuristic as shown in Tab. III.

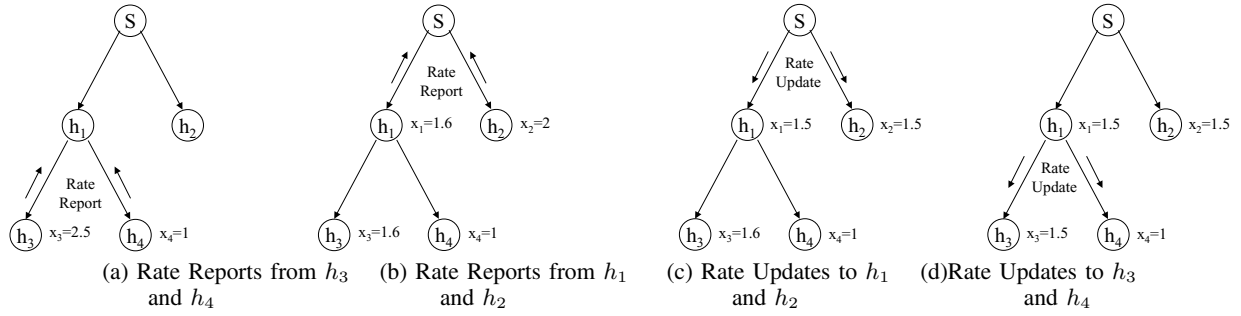


Fig. 4. Sample Illustrating the Max-min Allocation Algorithm

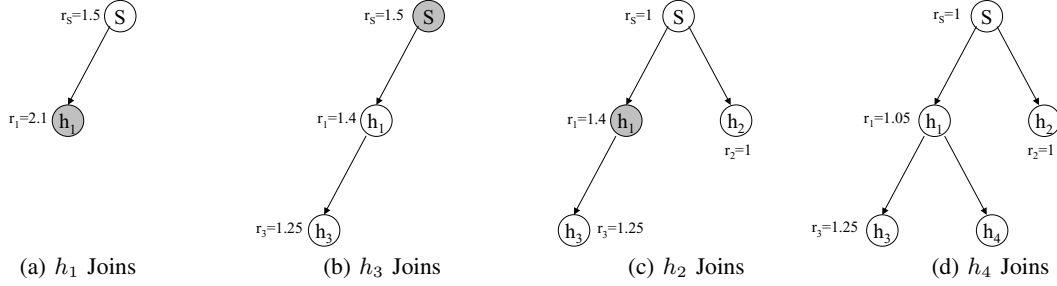


Fig. 5. Sample Illustrating the Tree Construction Algorithm

Initialization	
1	$T \leftarrow \{S\}$
2	Sort $H = \{h_1, \dots, h_N\}$, such that $c_1 \geq \dots \geq c_N$
3	$r_s \leftarrow c_S$
Tree Construction	
4	for $i = 1$ to N
5	$h_k \leftarrow \operatorname{argmax}(r_k \mid k = 1, \dots, i - 1)$
6	if $r_S > r_k$
7	Set S as h_i 's Parent
8	else
9	Set h_k as h_i 's Parent
10	$T \leftarrow T \cup h_i$
11	$r_S \leftarrow \frac{c_S}{n_S + 1}$
12	for $j = 1$ to i
13	$r_j \leftarrow \frac{c_j}{n_j + 1}$

TABLE III

MULTICAST TREE CONSTRUCTION HEURISTIC

The principle of our heuristic is as follows. (1) We always stream from high-end node to low-end node. Formally, if h_i is the child of h_k , i.e., $h_k \rightarrow h_i$, then it must satisfy that $c_i < c_k$. Each end host joins the tree in the order of its access link capacity: the one with the greatest capacity joins first. (2) Each time when a new host h_i joins, it attaches itself to the tree node with the greatest bottleneck bandwidth (r_k for host h_k), which is the ratio of h_i 's access link capacity and $n_i + 1$, n_i being the number of unicast routes routed through h_i 's link. By doing this, we try to maximize the minimum receiving rate in the tree.

We illustrate our heuristic using the sample topology in Fig. 3. Initially, h_1 through h_4 are sorted by the descending

order of their access link capacity. h_1 is chosen as the first one to join the tree (Fig. 5 (a)). Then the server's bottleneck bandwidth is $r_S = c_S/2 = 1.5$, h_1 's bottleneck bandwidth is $r_1 = c_1/2 = 2.1$. Therefore, h_1 is grayed, which means that it is chosen as the parent, which the next joining node, h_3 , will attach to. After joining the tree (Fig. 5 (b)), h_3 's bottleneck bandwidth is $r_3 = c_3/2 = 1.25$, and r_S is unchanged, $r_1 = c_1/3 = 1.4$ and $r_2 = 2/2 = 1$. Now S turns out to be the one with the greatest bottleneck bandwidth. In Fig. 5 (c), h_2 attaches itself to S , which makes $r_S = 1$, $r_1 = 1.4$, $r_2 = 1$ and $r_3 = 1.25$. Finally, h_4 joins the tree by attaching to h_1 , and results in the final rate vector, identical with the final result in Fig. 4. Note that in this example, the final tree in Fig. 5 (d) is the optimal tree T° defined in Sec. III-B. However, our heuristic does not guarantee to return the optimal tree every time. More analysis results on this issue can be found in Sec. V.

During the tree construction, the server S always keeps track of which node in the existing tree has the greatest bottleneck bandwidth. Therefore, each time a new node joins the tree, it first contacts the server, which in turns tells it which node to choose as parent. In each iteration only the bottleneck bandwidths of the new node and its parent node are updated. These two nodes need to report to the server. The total messaging overhead of the tree construction is $3N$. Note that the tree construction and rate allocation algorithms are independent from each other. Each time a new node joins the tree, it can start the max-min rate allocation algorithm in Tab. II by sending rate report to its parent.

Theorem 3 The minimum receiving rate of the tree constructed by the algorithm in Tab. III is at least $\frac{1}{2}$ of the optimal

value.

Experimentally, this algorithm greatly outperforms this approximation bound, which will be shown in Sec. V.

C. Dynamic Client Joins/Leaves

The tree construction algorithm in Tab. III requires nodes to join the tree in the sorted order of their access link capacities. However, in practice, a client may join or leave the multicast session at any time. It is highly expensive to always decompose the old tree and construct a new one from scratch when such an event happens. Moreover, doing so may frequently disturb existing clients in the multicast session.

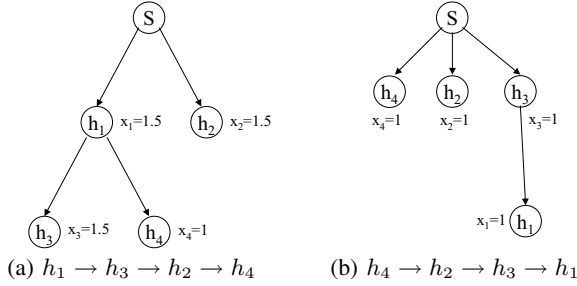


Fig. 6. Multicast Trees Resulted from Different Client Joining Orders

We modify the original algorithm to accommodate these practical issues, but still following the same principle: always streaming from high-end node to low-end node. In the modified algorithm, each time a new node h_i joins, it only considers the set containing the server S and all tree nodes whose link capacities are greater than c_i . From this set, h_i chooses the one with the greatest bottleneck bandwidth as its parent. If h_i leaves, each of its children rejoins the tree the same way h_i does.

By this algorithm, different client joining orders may result in totally different multicast trees. Fig. 6 illustrates such a difference, using the same example in Fig. 5. When clients join the tree in the descending order of their link capacities, the optimal tree is formed, as shown in Fig. 6 (a). However, if this order is reversed, we get a different tree with much degraded streaming rates. This is because every time a new client joins, all the existing tree nodes have smaller link capacities than its own. Therefore, it can only choose to stream from the server S . Eventually, the access link of S becomes highly congested.

To avoid this kind of suboptimal result, we enhance our algorithm with the “parent-child switching” method. Each time a new node h_i joins, it attaches itself to the tree node with the greatest bottleneck bandwidth, say h_k . If h_k 's link capacity is less than h_i 's capacity, i.e., $c_k < c_i$, then h_i switches its position with h_k . Note that h_i and h_k not only exchange their positions, but also exchange their children, if any. After this, h_i further checks whether the link capacity of its new parent is less than c_i , and continues to switch if the answer is yes. This process is repeated until h_i 's new parent has greater link capacity than c_i , or the root S is reached. If h_i leaves, each of its children rejoins the tree the same way h_i does.

Now using the same example in Fig. 6, we illustrate how the enhanced algorithm works. We follow the same client joining order as in Fig. 6 (b). Fig. 7 (a) shows the tree after h_4 and h_2 joins. At this point, both h_2 and S have bottleneck bandwidth of 1. When h_3 joins, it randomly chooses h_2 to attach to. Since $c_2 < c_3$, h_3 switches with h_2 (Fig. 7 (b)). After a round of max-min rate allocation, both h_3 and S have the greatest bottleneck bandwidth, and h_1 chooses to attach to h_3 (Fig. 7 (c)). Since $c_2 < c_3 < c_1$, h_1 first switches with h_3 , then with h_2 . The final tree is shown in Fig. 7 (d), whose rate allocation is $x = (1, 1.25, 1.25, 2)$, a result better than the one in Fig. 6 (b).

Theorem 4 The minimum receiving rate of the tree constructed by the dynamic algorithm is at least $\frac{1}{2}$ of the optimal value.

However, the price for the better performance is that the enhanced algorithm forces some existing tree nodes to switch their parents. We will explore this tradeoff in Sec. V.

V. EXPERIMENTAL RESULTS

We first test the tree construction heuristic in Tab. III: how close its resulted tree approximates the optimal tree. We compare the result of our algorithm with two types of optimal trees: (1) the *Max-min Tree*, the one that generates the most min-favorable rate allocation, i.e. the max-min allocation, and (2) *Max-Throughput Tree*, the one that returns the greatest aggregate rates of all clients, i.e., the maximum overall throughput³. For a multicast group of N clients, we repeat the same experiment for 100 times. Each time, the link capacities of the server and all clients are uniformly distributed between 1 and 5. We analyze our result through the following metrics.

- **Hit Ratio** records how many times our algorithm returns one of the optimal trees.
- **Minimum Rate** indicates how max-min fair a rate allocation is. Apparently, a min-favorable rate allocation results in a higher minimum rate than other ones.
- **Average Throughput** is the average rates of all clients.
- **Link Utilization** measures the percentage of overall link capacities that can be maximally utilized by a tree.

Due to the NP-hardness of the problem, the two optimal trees can only be found in an exhaustive search. By Cayley's theorem, given N clients, there exists $(N + 1)^{N-1}$ different trees[8], which makes the exhaustive search prohibitively expensive when N exceeds 10. Therefore, we are only able to present the results comparison in the case of small multicast groups, where $N < 10$. Although it remains unknown exactly how close it performs to the optimal results when N further increases, our algorithm continues to be steady in terms of average throughput and min-favorability, and remains high link capacity utilization.

Fig. 8 shows that the ratio of our heuristic returning the max-min tree continues to drop from 100% to less than 0% as

³In some cases, these two optimal trees can be the same one.

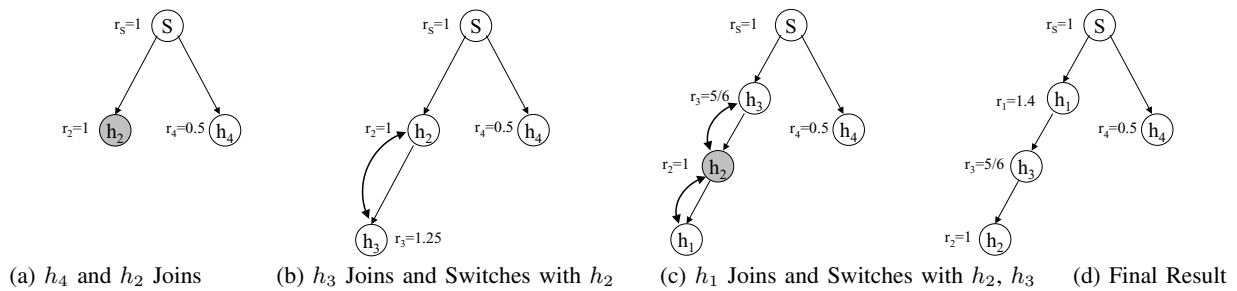


Fig. 7. Sample Illustrating Parent-Child Switching

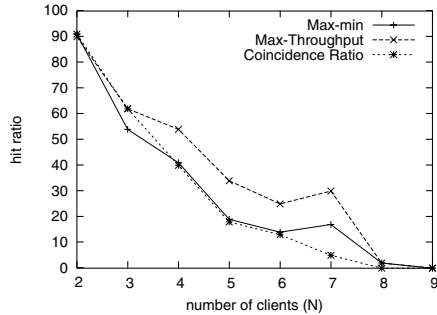
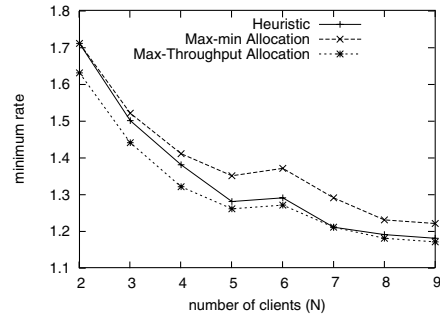
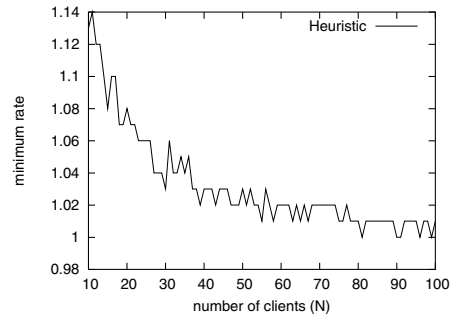


Fig. 8. Hit Ratio



(a) Comparison of Heuristic, Max-min and Max-Throughput Allocations



(b) Heuristic Allocation Alone

Fig. 9. Minimum Rate

N increases to 8. The ratio of returning the max-throughput tree is consistently higher but also quickly drops. The third curve, “coincidence ratio”, indicates the number of times the two optimal trees are actually one. This curve drops quickly from 80% to 0% as N increases to 8. The explanation for the above results is straightforward: as the number of possible trees explosively increases when we enlarge N , it is less and less likely for a tree to meet both optimal goals. It also seems that the principle of our heuristic (always streaming from the high-end client to the low-end client) is more suitable to achieve the goal of high overall throughput than the goal of max-min fairness. However, in the following results, we show that our algorithm is actually well-balanced between these two goals.

In Fig. 9 (a), max-min tree has the highest minimum rate, indicating that it always achieves the fairest rate allocation. The max-throughput tree, on the other hand, is the most unfair one. The curve of our heuristic lies stable in between them. Moreover, Fig. 9 (b) shows that as N increases, it decreases slowly at decelerating pace from 1.14 to 1.

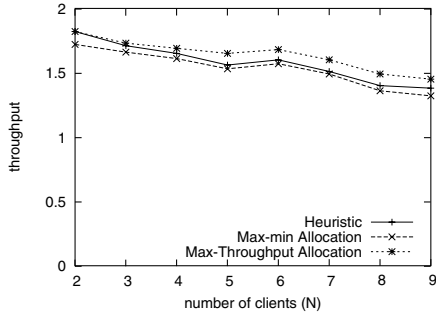
In Fig. 10 (a), we find out that the average throughput of the two optimal trees are in fact very close, where the max-throughput tree performs slightly better than the max-min tree. Also, similar to Fig. 9 (a), the curve of our heuristic lies in between the two of them. Fig. 10 (b) shows the same pattern as what appears in Fig. 9 (a), which implies that the algorithm can continue to construct the tree whose performance is close to the optimal one when N becomes large.

This conjecture is further confirmed by our observation on link utilization. Obviously, higher link utilization means higher

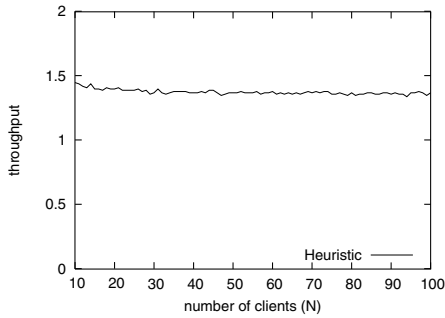
overall throughput. Fig. 11 (a) shows that the max-throughput tree has the best link utilization, which remains steady around 90% when N goes over 5. The max-min tree has the worst utilization, which never goes over 85%. The utilization of our heuristic lies in between them, and climbs to almost 90%. In Fig. 11 (b), the curve remains steady around this point, and never exceeds the range of 88% and 92%.

Now we examine the performance of our algorithm on dealing with dynamic client joins/leaves. The simulation runs for a time period, in which 100 clients join, then leave the multicast session. As shown in Fig. 12, the session size keeps changing along the time. For the same node join/leave sequence, we repeat our experiment for 100 times. Each time, the link capacities of the server and clients are uniformly distributed between 1 and 5.

As one can see from Fig. 13, the algorithm enhanced with parent-child switching technique consistently yields higher and

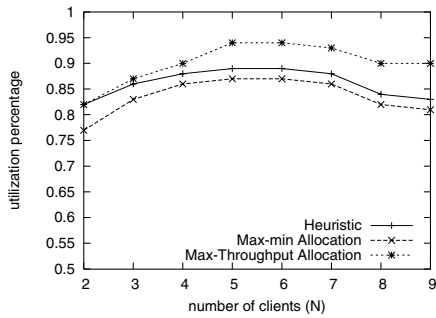


(a) Comparison of Heuristic, Max-min and Max-Throughput Allocations

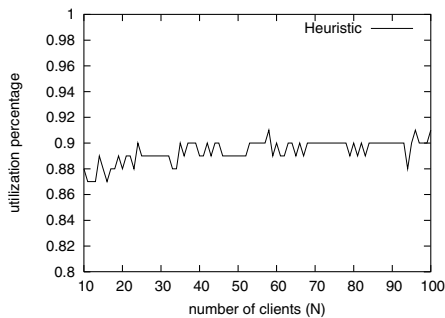


(b) Heuristic Allocation Alone

Fig. 10. Average Throughput



(a) Comparison of Heuristic, Max-min and Max-Throughput Allocations



(b) Heuristic Allocation Alone

Fig. 11. Link Utilization

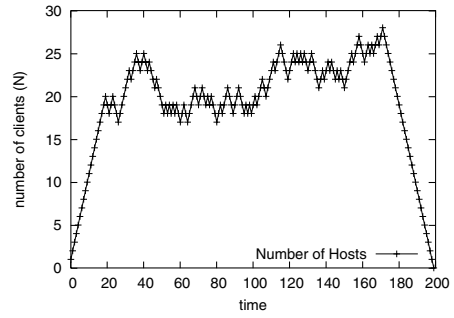


Fig. 12. Multicast Session Size (Dynamic Node Joins/Leaves)

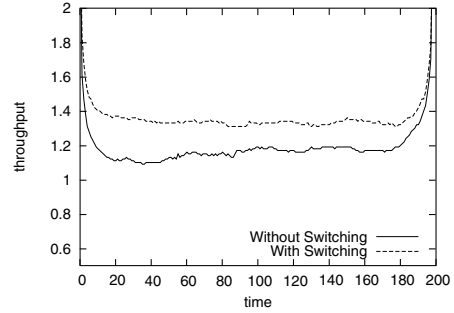


Fig. 13. Average Throughput (Dynamic Node Joins/Leaves)

more steady average throughput than the one without it. Also in Fig. 14, the link utilization ratio of the enhanced algorithm is always around 83%, only within 10% less than the same ratio we have measured for the static algorithm (Fig. 11 (b)). In contrast, the link utilization of the algorithm without parent-child switching is almost 15% less on average. Fig. 15 further shows that the algorithm with parent-child switching is more min-favorable than the one without.

Finally, we note that for the enhanced algorithm, each time when a node joins, or rejoins due to its parent's leaving, the average number of nodes affected by parent-child switching, i.e., the ones which switch positions with the joining node, and their children which have to switch parents, is 0.53, an overhead we consider bearable.

Based on the results presented in this section, we conclude that under heterogeneous network environments, different mul-

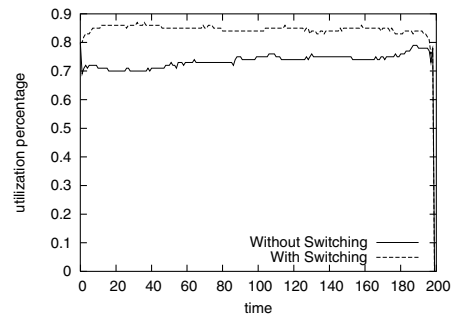


Fig. 14. Link Utilization (Dynamic Node Joins/Leaves)

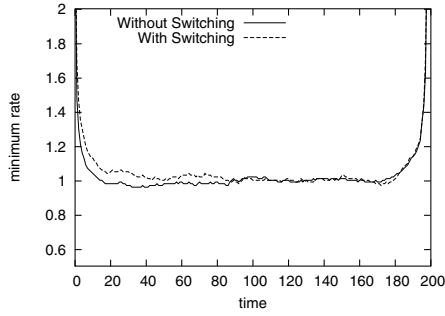


Fig. 15. Minimum Rate (Dynamic Node Joins/Leaves)

icast session scales, and dynamic client joins/leaves, our algorithm consistently achieves high average throughput, almost saturate link utilization, and consistent min-favorability.

VI. RELATED WORKS

We first review related works on optimal rate allocation. This problem has been explored in the context of unicast[9][10] and IP multicast[11][12]. In these works, the optimal rate allocation is the one that maximizes the aggregated utilities of all receivers $h_i (i = 1, \dots, N)$, subject to various constraints, such as network capacity. Here, the receiver utility is defined as a function of h_i 's streaming rate x_i , denoted $U_i(x_i)$. The function value can be understood as the perceived quality, user satisfaction, etc. Meanwhile, various fairness objectives can be achieved when $U(\cdot)$ takes certain forms[13][10]. Especially, when $U_i(x_i) = -(-\log x_i)^\alpha$, max-min allocation can be achieved⁴.

In our previous work[14], we used the utility-based approach to address the problem of optimal rate allocation in overlay multicast. Our goal was formalized as follows.

$$\begin{aligned}
 & \max_{x_i} && \sum_{i=1}^N U_i(x_i) && (10) \\
 & \text{subject to} && \mathbf{A} \cdot \mathbf{x} \leq \mathbf{c} \\
 & && \mathbf{B} \cdot \mathbf{x} \leq \mathbf{0} \\
 & \text{over} && \mathbf{x} \geq \mathbf{0}
 \end{aligned}$$

By non-linear optimization theory, we attacked Problem (10) by solving its dual[15]. A distributed algorithm was proposed, where each receiver adjusts its own streaming rate by exchanging “price” signals with each other. We proved that the algorithm finally converges to the optimal point, where the aggregated utilities are maximized.

Obviously, finding max-min rate allocation (Problem (8) in Sec. III-B) is only a special case of Problem (10). The contribution of this paper, then, is a light-weight distributed max-min rate allocation algorithm built on a simplified but well-accepted network assumption. It only needs one iteration to converge, which saves considerable messaging and measurement overhead, compared to the old algorithm in [14].

⁴In fact, this function infinitely approximates max-min fairness as $\alpha \rightarrow \infty$.

Now we review related works on optimal multicast tree construction, which can be considered as a routing problem. The objective of a routing algorithm can be diversified, such as shortest delay, minimum cost, etc. The objective of our work and those others we will review shortly, is to maximize the throughput, a crucial factor for bandwidth-sensitive applications.

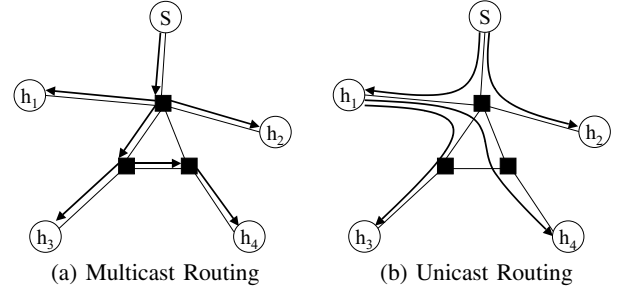


Fig. 16. Routing Problem

As shown in Fig. 16, a commonly used network model is a simple graph with unidirectional links. Each link is associated with a positive weight as its capacity. We consider a single multicast session with a source S and a set of receivers, and no other traffic. Then the optimal routing problem is to maximize the overall receiving rates of all receivers. If we assume the existence of IP multicast, i.e., all nodes in the picture are capable of replicating and forking data, then the problem becomes to find the *maximum bottleneck tree* (Fig. 16 (a)), which can be solved by a variation of Dijkstra’s algorithm[8].

However, in the case of overlay multicast, where only end nodes are capable of replicating and forking data (Fig. 16 (b)), the problem becomes to find routes for a set of unicast connections, such that their overall flow rates are maximized. Cohen et al.[16] showed that in the case of uni-rate multicast, where all receivers must have identical receiving rates, this problem is NP-hard. Wang et al.[17] studied the optimal routing problem for a set of independent unicast connections, and also proved it to be NP-hard. In this paper, we study this problem in the context of multi-rate multicast, and show that even on the simplest star topology⁵, finding the optimal tree is still NP-hard. Kim et al.[7] proposed an optimal tree construction algorithm, but based on the following assumptions. First, only the access link of an end host can be possibly congested. Second, the access link has incoming and outgoing bandwidths that do not affect each other. Finally, given two hosts h_i and h_j , if h_i has greater incoming bandwidth, then h_i ’s average outbound bandwidth to each of its children must also be greater than that of h_j (fair contribution).

Given the difficulty of optimal routing in overlay multicast, it becomes important to study what is a good heuristic to construct a multicast tree. We consider our proposed algorithm

⁵In a star topology, when the two ends of a unicast connection are determined, its route is unique. In this case, the optimal routing problem boils down to finding the optimal data relaying sequence, i.e., the optimal overlay tree.

as an initial work, and will focus our future study along this track.

VII. CONCLUSION

A fundamental challenge of overlay multicast is how to achieve the max-min rate allocation for all clients, which maximally utilizes the network resource, while maintaining max-min fairness. In this paper, we address this challenge in two steps. First, we propose a distributed algorithm, which returns max-min rate allocation for any overlay multicast tree. Second, we study how to find the optimal multicast tree, which returns the optimal max-min rate allocation among all trees. After proving the NP-hardness of this problem, we propose a heuristic algorithm of overlay multicast tree construction. A variation of the heuristic is also designed to handle the dynamic client join/departure. Both of them are guaranteed to construct a multicast tree, whose max-min rate allocation has the minimum receiving rate at least 1/2 of the same value returned by the optimal tree. Experimental results show that our algorithms are able to construct multicast tree with high average throughput, sufficient link utilization, and consistent min-favorability.

REFERENCES

- [1] Y. Chu, R. Rao, and H. Zhang, "A case for end system multicast," in *ACM SIGMETRICS*, 2000.
- [2] S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-driven layered multicast," in *ACM SIGCOMM*, 1996.
- [3] D. Rubenstein, J. Kurose and M. Vetterli, "The impact of multicast layering on network fairness," in *ACM SIGCOMM*, 1999.
- [4] Y. Cui and K. Nahrstedt, "Layered peer-to-peer streaming," in *NOSS-DAV*, 2003.
- [5] E. Amir, S. McCanne, and R. Katz, "An active service framework and its application to real-time multimedia transcoding," in *ACM SIGCOMM*, 1998.
- [6] M. Yajnik, J. Kurose and D. Towsley, "Packet loss in the mbone multicast network," in *IEEE Global Internet*, 1996.
- [7] M. Kim, S. Lam and D. Lee, "Optimal distribution tree for internet streaming media," in *IEEE ICDCS*, 2003.
- [8] N. Deo, *Graph Theory with Applications to Engineering and Computer Science*, 1994.
- [9] S. Low and D. Lapsley, "Optimization flow control, i: Basic algorithm and convergence," *IEEE/ACM Transactions on Networking*, vol. 7, no. 6, 1999.
- [10] F. Kelly, A. Maulloo and D. Tan, "Rate control for communication networks: Shadow prices, proportional fairness and stability," *Journal of Operations Research Society*, vol. 49, no. 3, 1998.
- [11] K. Kar, S. Sarkar and L. Tassiulas, "Optimization based rate control for multirate multicast sessions," in *IEEE INFOCOM*, 2001.
- [12] —, "A low-overhead rate control algorithms for maximizing aggregate receiver utility for multirate multicast sessions," in *SPIE ITCOM*, 2001.
- [13] F. Kelly, "Charging and rate control for elastic traffic," *European Transactions on Telecommunications*, vol. 8, no. 1, 1997.
- [14] Y. Cui, Y. Xue and K. Nahrstedt, "Optimal resource allocation in overlay multicast," in *IEEE ICNP 2003*, 2003.
- [15] D. Bertsekas, *Nonlinear Programming*. Athena Scientific, 1995.
- [16] R. Cohen and G. Kaempfer, "A unicast-based approach for streaming multicast," in *IEEE INFOCOM*, 2001.
- [17] J. Wang, L. Li, S. Low, J. Doyle, "Can shortest-path routing and tcp maximize utility," in *IEEE INFOCOM*, 2003.
- [18] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, 1979.

VIII. APPENDIX 1: PROOF OF THEOREM 1

Proof: It only suffices to show that a special case of the problem can be reduced into a well-known NP problem.

Consider a star topology with $N+1$ clients, shown in Fig. 17 (b). It is sorted such that $c_1 \leq \dots \leq c_N < c_{N+1}$. Also, $\sum_{i=1}^N c_i = c_{N+1} = c_S$, and $c_i \leq \frac{c_S}{2}$ ($i = 1, \dots, N$). All c_i and c_S are integers. The goal is to find an optimal multicast tree $T^\circ \in \mathcal{T}$ with the max-min rate allocation, i.e., $x^*(T^\circ) > m$ $x^*(T | T \in \mathcal{T} \text{ and } T \neq T^\circ)$.

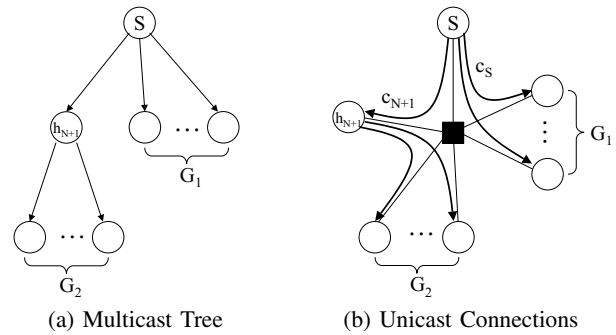


Fig. 17. Proof of Theorem 2

Apparently, (c_1, \dots, c_{N+1}) is the highest achievable max-min rate allocation, where all clients' rate maximizes to its own link capacity. To achieve so, all clients must be directly connected to the server, and $\sum_{i=1}^{N+1} c_i \leq c_S$, which is impossible. However, by the alphabetical nature of max-min rate allocation, the next highest achievable one is $(c_1, \dots, c_N, x_{N+1})$, where $x_{N+1} < c_{N+1}$. This optimal tree is shown in Fig. 17 (a). Obviously, h_1 through h_N must be the leaf nodes, in order to fully utilize their link bandwidth as pure receivers. They are divided into two groups, one group as the children of the server S , denoted as G_1 , the other as the children of h_{N+1} , denoted as G_2 . Let $c(G_1) = \sum_{h_i \in G_1} c_i$, $c(G_2) = \sum_{h_i \in G_2} c_i$, then $c(G_1) + c(G_2) = c_{N+1}$ and $x_{N+1} = c(G_1)$. It must satisfy that $x_{N+1} < \frac{c_{N+1}}{2}$, since otherwise there will not be enough residual bandwidth from S and x_{N+1} to support $c(G_1) + c(G_2)$. Therefore, to maximize x_{N+1} , G_1 and G_2 have to be divided such that $\epsilon = |c(G_1) - c(G_2)|$ is minimized.

This optimal partition also guarantees that $x_{N+1} \geq c_i$ ($h_i \in G_2$). Otherwise, there must exist a client h'_i in G_2 , such that $x_{N+1} = c(G_1) < c'_i \leq \frac{c_S}{2}$. If so, there exists another partition, where only h'_i belongs to G'_2 and all the others belong to G'_1 . Then we find out that $|c(G'_1) - c(G'_2)| < \epsilon$, which contradicts the claim that G_1 and G_2 are the optimal partition.

However, minimizing ϵ is known as the integer partition problem, i.e., given a set of integers, divide them into two subsets, such that the difference of their sums is minimized. This problem is NP-complete[18]. ■

IX. APPENDIX 2: PROOF OF THEOREM 2

Proof: We show that in x^* , increasing x_i will inevitably decrease some rate x_j , which is already smaller than x_i (Proposition 1).

We first consider the case when h_i 's link is fully utilized. If h_i is a leaf node, then it already hits the maximum rate it can get. Otherwise, increasing x_i will decrease the rate of its children, which is already smaller or equal to x_i .

If h_i 's link is not fully utilized, then x_i is the maximum among h_i 's siblings' rates, and equals to the rate of its parent, say h_p . (1) If h_p 's link is fully utilized, then increasing x_i will force h_p to decrease x_i 's sibling's rate, which is already smaller or equal to x_i , or decrease its own rate x_p , which is against the data constraint, i.e., $x_i \leq x_p$. (2) If h_p 's link is not fully utilized either, then similar to x_i , x_p is the maximum among h_p 's siblings' rates, and equals to the rate of h_p 's parent $h_{p'}$. $h_{p'}$ faces the same dilemma as h_p in case (1), if $h_{p'}$'s link is fully utilized. In summary, this procedure is repeated as we further go up the tree, until some node is met, whose link turns out to be the bottleneck, which is guaranteed by our algorithm. ■

X. APPENDIX 3: PROOF OF THEOREM 3

Proof: Suppose IP multicast is available in the star topology in Fig. 17 (b). Then the streaming rate allocation is $(\min\{c_S, c_N\}, \dots, \min\{c_S, c_1\})$, assuming that the nodes are sorted by the descending order of their access bandwidth, i.e., $c_N \leq c_{N-1} \dots \leq c_1$. Such a rate allocation is obviously superior than the optimal rate allocation in the overlay multicast setting. However, the following rate allocation is feasible in overlay multicast: $\hat{x} = (\frac{\min\{c_S, c_N\}}{2}, \dots, \frac{\min\{c_S, c_1\}}{2})$, which can be achieved by forming a chain from the server S to h_1, h_2, \dots , then finally to h_N . This means that there always exists a rate allocation, whose approximation ratio to the optimal allocation is strictly more than 1/2. Therefore, we only need to show that the tree constructed by the algorithm in Tab. III is able to return a rate allocation $x \geq_m \hat{x}$.

To prove this, we use induction. When the first node h_1 attaches to the server S , the current minimum rate is $\min\{c_S, c_1\}$, and the maximum bottleneck bandwidth of the current tree is at least $\frac{c_1}{2}$, which means that the next node h_2 can attach to h_1 and share at least half of h_1 's access bandwidth, which is greater than $\frac{c_2}{2}$. Now suppose n nodes are already attached to the tree, and the current minimum rate is greater than or equal to $\frac{\min\{c_S, c_n\}}{2}$. Since h_n is the last one attached, it must be a leaf node. Thus its bottleneck bandwidth is $\frac{c_n}{2}$, which gives a lower bound to the maximum bottleneck bandwidth offered to the next node h_{n+1} , which is greater than $\frac{c_{n+1}}{2}$. Therefore, during each stage n of the tree construction, its bottleneck bandwidth is at least half of c_n , the minimum access bandwidth of all nodes attached to the tree, which means that the minimum receiving rate for the next joining node n_{n+1} is at least $\frac{c_{n+1}}{2}$. Thus, the rate allocation of the tree is at least as min-favorable as \hat{x} . ■

XI. APPENDIX 4: PROOF OF THEOREM 4

Proof: We first show that parent-child switching always improves the rate allocation of the existing tree. Note that in the algorithm, after the new node first attaches to the tree, it climbs up the tree along the path from the root to itself, until

the access bandwidths of all client nodes along the path are in descending order. For example, in Fig. 7 (c), when h_1 joins the tree, the path order is $S \rightarrow h_3 \rightarrow h_2 \rightarrow h_1$. After switching, it becomes $S \rightarrow h_1 \rightarrow h_3 \rightarrow h_2$, since $c_1 > c_3 > c_2$. Since h_3 is replaced by h_1 and $c_3 < c_1$, the receiving rates of all h_3 's subtrees before switching (if there is any) will not be decreased. Similarly, since h_2 is replaced by h_3 and $c_2 < c_3$, the receiving rates of all h_2 's subtrees before switching (if there is any) will not be decreased. However, h_1 is replaced by h_2 and $c_2 < c_1$. Nevertheless, before switching, h_1 's receiving rate was constrained by h_2 's access bandwidth and h_1 was a leaf node. Therefore, when h_2 takes h_1 's position, its receiving rate will not be less than h_1 's receiving rate before switching.

We again use induction to prove our theorem. Let us denote c_{min} as the minimum access bandwidth of all nodes attached to the tree so far, and h_{min} as the node whose access bandwidth is c_{min} . We assume the joining order is h_1, \dots, h_N , but not necessarily sorted based on their access bandwidths. When only h_1 is attached to the tree, h_1 is h_{min} , which is a leaf node. Then the minimum receiving rate is $\min\{c_S, c_{min}\}$, and the bottleneck bandwidth is at least $\frac{c_{min}}{2}$. Now suppose n nodes are already attached to the tree, and the current minimum receiving rate is at least $\frac{\min\{c_S, c_{min}\}}{2}$. Also, h_{min} must be a leaf node due to parent-child switching, implying that the bottleneck bandwidth is at least $\frac{c_{min}}{2}$. Therefore, when a new node h_{new} attaches to the tree, its receiving rate is at least $\frac{\min\{c_S, c_{min}, c_{n+1}\}}{2}$. If $c_{n+1} \leq c_{min}$, then according to our definition, c_{min} will be updated as $c_{min} \leftarrow c_{n+1}$. If $c_{n+1} > c_{min}$, there are two cases. (1) If h_{n+1} is attached to h_{min} , then after parent-child switching, h_{min} will still be a leaf node. (2) Otherwise, h_{min} will stay as a leaf node. Also as we have proved, parent-switching does not decrease the minimum receiving rate. Now we can conclude that after h_{n+1} joins the tree, the following are still true: the minimum receiving rate is at least $\frac{\min\{c_S, c_{min}\}}{2}$, and h_{min} is a leaf node, implying that the bottleneck bandwidth is at least $\frac{c_{min}}{2}$. From the proof of **Theorem 3**, we can see that the rate allocation of the tree is at least as min-favorable as \hat{x} . ■