

# Maximizing Throughput in Layered Peer-to-peer Streaming

Liang Dai, Yi Cui and Yuan Xue

**Abstract**—Layered streaming is an effective solution to address the receiver heterogeneity in peer-to-peer (P2P) multimedia distribution. This paper targets a fundamental challenge in this application, i.e., how to find optimal routing structure maximizing receiver throughput and achieving intra-layer and inter-layer fairness. We formulate the problem using the multicommodity flow theory, and propose a series of routing algorithms that fully explore the tradeoff between theoretical optimality and practicability. For each algorithm, we have proved its optimality to achieve maximum throughput under fairness constraint, or approximation bound to the optimal rate. Experimental results confirm our algorithms to greatly outperform their theoretical bounds.

## I. INTRODUCTION

A fundamental problem in Internet-based media distribution is routing: how to establish an optimal routing structure (e.g., tree, mesh), which delivers the content from the sender to all receivers, simultaneously achieving certain optimization objective, such as throughput, or delay. Adding to the complexity of this problem is the *heterogeneity* challenge. As users connect to the Internet via access technologies (Ethernet, dial-up modem, and ADSL, etc.) with order-of-magnitude difference regarding inbound/outbound capacities, a one-size-fit-all streaming quality simply cannot be found. The remedy to this challenge is the layer-encoded streaming approach[1][2][3], in which the video stream is partitioned into multiple layers and fed into parallel routing structures. Each receiver needs only join a subset of them to recover the video with certain quality degradation.

In the application context of layer-encoded video streaming, this paper studies the optimal routing problem with the maximum throughput objective, i.e., given the sender and a set of receivers with heterogeneous demands, how to find the optimal routing structures that maximize the end-to-end throughputs of all receivers in proportion to their demands?

While layer-encoded media distribution can be supported by many communication paradigms (IP multicast[1], overlay multicast[3], P2P network[2], etc.), we confine our work to the domain of P2P streaming due to its practicability proved by recent large-scale Internet deployments[4]. In particular, our solution bears the following design objectives. (1) *Intra-layer and inter-layer fairness* must be maintained among receivers subscribing to different layers of the video stream. If each layer has a demand streaming rate, the final throughputs to all its receivers should be in the same proportion to this demand. In addition, this proportion should be the same across all layers. (2) *Topology-awareness* must be taken into account in making routing decisions. In P2P network, content is streamed from

one peer to another via an *overlay edge* encompassing multiple physical links. Unlike the fixed outbound/inbound capacity of each peer, the achievable throughputs of overlay paths are dynamic due to background traffic, and inter-dependent due to the link-correlation phenomenon pertinent to overlay and P2P networks. An effective metric must be introduced to capture the capacity availability of an overlay edge.

We model this problem using the multicommodity flow theory. We view different layers of the video content as commodities to be distributed. Each commodity (layer) is routed via finding the minimum spanning tree (MST) spanning all its receivers in the P2P network. Here, the metric for MST routing is an artificial length assigned to each overlay edge as an asymptotic function of its traffic load. Evidently, the attempt to find the minimum routing structure, i.e., MST, equals to finding the most lightly-loaded paths. From this simple idea, our work makes the following contributions. (1) We develop the optimal algorithm, which proves to return the maximum achievable throughputs for all layers in proportion to their demands. The key component of the algorithm is the definition of the overlay edge length function. An important assumption of the optimal algorithm is that the data at each layer is arbitrarily splittable. (2) From the optimal algorithm, we develop its realistic variation, which allows each layer to be unsplittable, or can be split into no more than  $k$  trees. In other words, the entire content of a layer is distributed via a single or limited number of multicast trees. We also prove the competitive ratio of throughputs between the two algorithms to be constrained by an upper-bound. (3) We further modify the algorithm by altering the definition of overlay edge length function in accordance with the fact that a peer might only have partial knowledge of an overlay edge. The aforementioned upper-bound stays unchanged in the modified algorithm.

All above contributions are evaluated via simulated experiments. Due to space constraint, we move proofs the theorems of this paper to our technical report[5]. The rest of this paper is organized as follows.

The rest of this paper is organized as follows. In Sec. II, we formulate the problem in its most basic setting, which helps to illustrate the theoretical background of our algorithms. Sec. III presents the algorithms and their theoretical performance bound, and discusses possible extensions. Sec. IV presents the experimental results. We finally discuss related work at Sec. V and conclude at Sec. VI.

## II. MODEL

We formulate the routing problem in layered peer-to-peer streaming as a multicommodity flow problem. Let  $G = (\mathcal{V}, \mathcal{L})$  be a directed graph, with capacity  $c_l$  on each physical edge

Liang Dai, Yi Cui and Yuan Xue are with the Department of Electrical Engineering and Computer Science, Vanderbilt University. Their email addresses are {liang.dai, yi.cui, yuan.xue}@vanderbilt.edu.

$l \in \mathcal{L}$ . Suppose the video is decomposed into layer set  $\mathcal{M}$ . Each layer  $m \in \mathcal{M}$  is considered a commodity.  $\mathcal{R}(m)$  is the set containing all receivers of  $m$ ,  $S(m)$  is the sender<sup>1</sup>.  $dem(m)$  is the demand of  $m$ , which is the desired streaming rate from  $S(m)$  to all nodes in  $\mathcal{R}(m)$ . We summarize these notations into the overlay graph model.

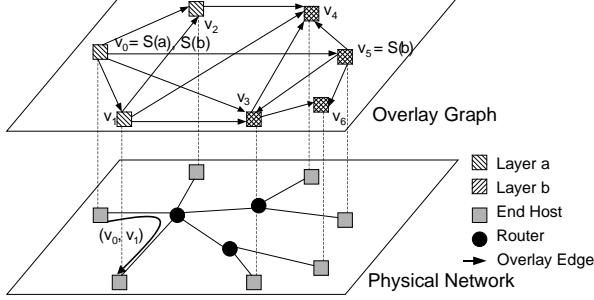


Fig. 1. Illustration of Layered P2P Streaming

For a layer  $m \in \mathcal{M}$ , we define its overlay graph  $G(m) = (\mathcal{V}(m), \mathcal{E}(m))$ . In this graph, the node set  $\mathcal{V}(m) = \{S(m)\} \cup \mathcal{R}(m)$  includes the sender and all receivers of layer  $m$ . A directed edge  $(v_s, v_t) \in \mathcal{E}(m)$  represents the data dependency between two nodes  $v_s$  and  $v_t$ , i.e.,  $v_t$  can retrieve data from  $v_s$ . In Fig. 1, two layers,  $a$  and  $b$ , are distributed by the server  $v_0$ .  $\mathcal{V}(a) = \{v_0, v_1, v_2, v_3, v_4, v_5, v_6\}$ . Here, node  $v_0$  is the sender of layer  $a$ , and nodes  $v_1$  through  $v_6$  are the receivers of  $a$ . For layer  $b$ ,  $\mathcal{V}(b) = \{v_0, v_3, v_4, v_5, v_6\}$ . Here, node  $v_0$  is the sender of layer  $b$ , and the rest belong to the receiver set  $\mathcal{R}(b)$ . Each overlay edge, e.g.,  $(v_0, v_1)$  in Fig. 1, corresponds to the unicast route at the physical network. Note that our formulation applies to both the cumulative layered streaming[1][2] (exemplified in Fig. 1) and non-cumulative layered streaming (e.g., multiple description coding).

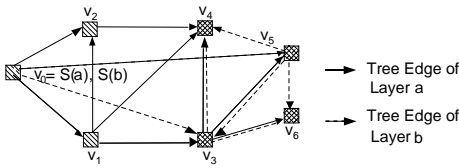


Fig. 2. Illustration of Overlay Spanning Tree

On top of each overlay graph, different spanning trees can be found. Fig. 2 shows a sample spanning tree for the overlay graphs  $G(a)$  and  $G(b)$  in Fig. 1. With the formulation of overlay graph, the problem boils down to, for each layer  $m \in \mathcal{M}$ , finding a set of spanning trees on top of its overlay graph  $G(m)$ . The flow rate of layer  $m$  is the aggregated rate of all its spanning trees. We collect these trees into the set  $\mathcal{T}(m) = \{t_j(m)\}$ . We use  $f_j(m)$  to denote the flow of layer  $m$  sent along the tree  $t_j(m)$ . Our goal is to maximize the flow rates of all layers, formulated as the following linear

<sup>1</sup>In the single-server case,  $S(m)$  refers to the same node for all  $m \in \mathcal{M}$ . In the multi-server case, layers could be distributed by different servers.

programming problem.

**M :**

$$\text{maximize } f \quad (1)$$

$$\text{subject to } \sum_{j=1}^{|\mathcal{T}(m)|} f_j(m) \geq f \cdot dem(m), m \in \mathcal{M} \quad (2)$$

$$\sum_{m \in \mathcal{M}} \sum_{j=1}^{|\mathcal{T}(m)|} f_j(m) \cdot n_l(t_j(m)) \leq c_l, l \in \mathcal{L} \quad (3)$$

$$f \geq 0, f_j(m) \geq 0, 1 \leq j \leq |\mathcal{T}(m)|, \forall m \in \mathcal{M}$$

The objective of **M** is to maximize the scalar value  $f$ , subject to the fairness and capacity constraints. Inequality (2) enforces *fairness constraint* by requiring that the comparative ratio of traffic routed for different commodities satisfies the comparative ratio of their demands, i.e., at least  $f \cdot dem(m)$  units of commodity flow can be routed simultaneously for each layer  $m \in \mathcal{M}$ . Thus, the absolute value of  $dem(m)$  is meaningless, as we can easily tune the value of  $f$  by scaling up/down all demands, while  $f \cdot dem(m)$  stays unchanged. In other words, weighted max-min fairness is maintained by Inequality (2). Inequality (3) specifies the *capacity constraint* that the traffic routed on each physical edge  $l \in \mathcal{L}$  does not exceed its capacity  $c_l$ . Note here that  $n_l(t_j(m))$  is an integer value denoting the number of appearances of  $l$  in  $t_j(m)$ . This value could be greater than one, since a physical edge may appear in a tree more than once, a unique feature in P2P network.

### III. ALGORITHMS

Before presenting the algorithm for **M**, we first formulate its dual as follows.

**D :**

$$\text{minimize } \sum_{l \in \mathcal{L}} c_l \cdot d_l$$

$$\text{subject to } \sum_{l \in \mathcal{L}} n_l(t_j(m)) \cdot d_l \geq l(m), m \in \mathcal{M} \quad (4)$$

$$\sum_{m \in \mathcal{M}} z(m) \cdot dem(m) \geq 1 \quad (5)$$

$$d_l \geq 0, \forall l \in \mathcal{L}, z(m) \geq 0, \forall m \in \mathcal{M}$$

**D** corresponds to the problem of assigning length  $d_l$  to each edge  $l \in \mathcal{L}$  and weight  $z(m)$  to each layer  $m \in \mathcal{M}$ , such that for  $m$ , the length of any spanning tree in  $\mathcal{T}(m)$  is at least  $z(m)$ , and the weighted sum of  $z(m)$  by  $dem(m)$  over all sessions is at least 1. By LP duality theory, the minimum of **D** is the maximum of **M**.

We can also interpret this problem from the viewpoint of supply and demand. Here,  $d_l$  represents the marginal price of  $l$ , i.e., the cost of using an additional unit of capacity of  $l$ . The more traffic is routed through  $l$ , the more expensive its bandwidth becomes, specified via  $d_l$ . In a similar fashion,  $z(m)$  represents the marginal cost of not satisfying another unit of  $dem(m)$ , the traffic demand of layer  $m$ .

### A. Optimal Algorithm

Based on this interpretation, the optimal algorithm is designed to run iteratively, in each iteration solving the primal  $\mathbf{M}$  and dual  $\mathbf{D}$  alternatively, as given in Tab. I.

1	$\forall l \in \mathcal{L}, d_l \leftarrow \beta/c_l, \sigma_l \leftarrow 0$
2	$f_j(m) \leftarrow 0, t_j(m) \in \mathcal{T}(m), m \in \mathcal{M}$
3	<b>while</b> $\sum_{l \in \mathcal{L}} c_l \cdot d_l < 1$
4	<b>for</b> $\forall m \in \mathcal{M}$ <b>do</b>
5	$dem'(m) \leftarrow dem(m)$
6	<b>while</b> $\sum_{l \in \mathcal{L}} c_l \cdot d_l < 1$ <b>and</b> $dem'(m) > 0$
7	$t \leftarrow$ minimum overlay spanning tree in $\mathcal{T}(m)$ using $d_l$
8	$c \leftarrow \min\{dem'(m), \min_{l \in t} \frac{c_l}{n_l(t)}\}$
9	$dem'(m) \leftarrow dem'(m) - c$
10	$f(t) \leftarrow f(t) + c$
11	$\forall l \in t, d_l \leftarrow d_l(1 + \epsilon \frac{n_l(t)c}{c_l}), \sigma_l \leftarrow \sigma_l + \frac{c}{c_l}$
12	<b>end while</b>
13	<b>end for</b>
14	$\sigma_{\max} \leftarrow \max_{l \in \mathcal{L}} \sigma_l$

TABLE I  
OPTIMAL ALGORITHM FOR LAYERED P2P STREAMING

At the *initialization* phase (Lines 1 and 2), we assign initial length  $d_l$  for each physical edge  $l \in \mathcal{L}$ , and initialize  $\sigma_l$  as 0, which denotes traffic load of  $l$ , i.e., the percentage of its occupied capacity. We also initialize the tree set  $\mathcal{T}(m)$  for each layer  $m$  as empty. The rest of the algorithm proceeds in phases. Each phase contains  $|\mathcal{M}|$  iterations, each runs for a layer  $m \in \mathcal{M}$  and consists of three substeps.

*Finding Minimum Overlay Spanning Tree* (Line 7) is the first step: for each layer  $m \in \mathcal{M}$ , we first construct its overlay graph, a complete graph encompassing all members of  $m$ . Each edge corresponds to the unicast route connecting the two end nodes. The length of the edge is the aggregated length of all physical edges of this unicast route. Then we obtain the minimum overlay spanning tree  $t$  by running the MST algorithm on top of the overlay graph. What follows is the second step *Routing Traffic* (Lines 8 to 10), where we route traffic on  $t$ , whose amount is determined by the capacity of the bottleneck link. At the final step *Edge Length Update*: for each physical link  $l \in \mathcal{L}$ , we update its length based on its traffic increment as defined in Line 11, meanwhile updating its traffic load indicator  $\sigma_l$ . Each iteration is finished when  $dem(m)$  amount of traffic has been routed in the above three steps. Based on the function definition in Line 11, the edge length  $d_l$  increases asymptotically in each phase. Thus, the condition defined in Line 3 will be violated, which terminates the algorithm.

Finally, we note that the traffic routed in the algorithm might exceed the capacity of the physical edge, formally,  $\sigma_l > 1$ . By scaling the traffic routed for each layer  $m$  by  $\sigma_{\max}$ , the traffic load indicator of the most congested link, we should obtain a feasible solution.

**Theorem 1:** If  $\beta = (|E|/(1 - \epsilon))^{-1/\epsilon}$ , then the final flow generated by the optimal algorithm has a value at least  $(1 - 3\epsilon)$  times the optimal value of  $\mathbf{M}$ .

Our optimal algorithm is a fully polynomial time approximation scheme (FPTAS) to problem  $\mathbf{M}$ . A FPTAS is a family of algorithms that finds an  $\epsilon$ -approximate solution for any error parameter  $\epsilon > 0$ . Its running time is polynomial in the size of the network ( $|\mathcal{V}|$  and  $|\mathcal{L}|$ ), the number of commodities ( $|\mathcal{M}|$ ), and  $1/\epsilon$ . Evidently, smaller  $\epsilon$  results in better approximation to the optimal point, with longer running time. Our algorithm is based on the FPTAS scheme proposed by Garg and Koneman[6], later improved by Fleischer[7].

However, the solvability of the problem  $\mathbf{M}$  relies on the following unrealistic assumptions. First, the traffic of each layer  $m$  is *arbitrarily splittable*: the streaming traffic of each layer can be arbitrarily split and fed into unlimited number of trees. Second, the overlay network has the *complete topology knowledge* of the underlying physical network, as well as the capacity of each physical link. In the remainder of this section, we progressively develop realistic algorithms with the absence of these assumptions. Nevertheless, the basic methodology presented in the optimal algorithm will survive and remain to play the central role.

### B. Online Algorithm to Route Unsplittable Traffic

To enforce the content at each layer unsplittable, we need to introduce a 0–1 variable  $x_j(m)$  that only allows a tree  $t_j(m)$  to carry none or whole traffic of layer  $m$ . Then the problem  $\mathbf{M}$  is reformulated as follows.

$$\begin{aligned}
 & \mathbf{M}_I : \\
 & \text{maximize} \quad f \\
 & \text{subject to} \quad \sum_{j=1}^{|\mathcal{T}(m)|} f_j(m) \cdot x_j(m) \geq f \cdot dem(m), m \in \mathcal{M} \\
 & \quad \sum_{m \in \mathcal{M}} \sum_{j=1}^{|\mathcal{T}(m)|} f_j(m) \cdot x_j(m) \cdot n_l(t_j(m)) \leq c_l, l \in \mathcal{L} \\
 & \quad \sum_{j=1}^{|\mathcal{T}(m)|} x_j(m) = 1 \\
 & \quad f \geq 0, f_j(m) \geq 0, x_j(m) \in \{0, 1\}, 1 \leq j \leq |\mathcal{T}(m)|
 \end{aligned}$$

$\mathbf{M}_I$  is an 0–1 integer programming problem, which is NP-hard[8]. Also it can be easily extended to the case when a layer  $m$  has to be extended to at most  $k$  trees. We can view  $m$  as the collection of  $k$  commodities (layers) which happen to have the same set of receivers. The sum of their demands equals to  $dem(m)$ , and each commodity can have only one tree.

1	$\forall l \in \mathcal{L}, d_l \leftarrow \beta/c_l, \sigma_l \leftarrow 0$
2	$f_j(m) \leftarrow 0, t_j(m) \in \mathcal{T}(m), m \in \mathcal{M}$
3	<b>for</b> $\forall m \in \mathcal{M}$ <b>do</b>
4	$t \leftarrow$ minimum overlay spanning tree in $\mathcal{T}(m)$ using $d_e$
5	$f(t) \leftarrow f(t) + dem(m)$
6	$\forall l \in t, d_l \leftarrow d_l(1 + \epsilon \frac{dem(m)}{c_l}), \sigma_l \leftarrow \sigma_l + \frac{n_l(t)dem(m)}{c_l}$
7	$\sigma_{\max} \leftarrow \max_{l \in \mathcal{L}} \sigma_l$

TABLE II  
ONLINE ALGORITHM TO ROUTE UNSPLITTABLE TRAFFIC

The proposed algorithm, as outlined in Tab. II, has the same structure as the optimal algorithm. However, the traffic is routed in an *online* fashion, where in each iteration, the traffic for a layer is routed via one tree and the edge length is updated accordingly based on the same function shown in the optimal algorithm, until all layers has been served. There is no need to further continue the iteration to satisfy certain stop condition as in the optimal algorithm.

We measure the performance of the online algorithm by comparing its achievable throughput  $f_{online}$  by the optimal value  $f^*$  obtained via the optimal algorithm, in terms of their competitive ratio  $f^*/f_{online}$ . The following theorem verifies that there exists a worst case bound to this ratio.

**Theorem 2:** The online algorithm returns the competitive ratio bounded by  $\log |\mathcal{L}|$ .

Note that this bound is the best bound known for online algorithms so far[9]. Compared to Theorem 1, we can see that to achieve this bound, the online algorithm has no prerequisite on the initial length  $\beta$  and the step size  $\epsilon$ . In addition, the performance bound is not affected by the sequence that difference layers are routed by the algorithm.

### C. Modified Online Algorithm with Partial Topology Knowledge

The algorithm presented so far have made a strong assumption that that the overlay network has entire knowledge of its underlying physical network. This means that in order to function properly, the algorithm must know the complete topology of the underlying network, and the capacity of each physical link. However, such complete knowledge can be hardly acquired in practice. Even if possible, the link-correlation problem would incur prohibitive communication overhead. If a physical link is shared by a group of overlay edges, then each time the traffic on any overlay edge changes, the peer controlling this edge will have to notify all other overlay edges (controlled by different peers) about this change.

Based on these concerns, a more desirable alternative is to have each peer update the lengths of its own overlay edges, instead of updating the weights of each physical link en route, also without interfering with other overlay edges.

1	$\forall l \in \mathcal{L}, \sigma_l \leftarrow 0$
2	$\forall e \in \mathcal{E}(m) (m \in \mathcal{M}), d_e \leftarrow  e \beta/c_e, c_e \leftarrow \min\{c_l \mid l \in e\}$
3	$f_j(m) \leftarrow 0, t_j(m) \in \mathcal{T}(m), m \in \mathcal{M}$
4	<b>for</b> $\forall m \in \mathcal{M}$ <b>do</b>
5	$t \leftarrow$ minimum overlay spanning tree in $\mathcal{T}(m)$ using $d_e$
6	$f(t) \leftarrow f(t) + dem(m)$
7	$\forall e \in t, d_e \leftarrow d_e(1 + \epsilon \frac{dem(m)}{c_e})$
8	$\forall l \in e, \sigma_l \leftarrow \sigma_l + \frac{n_l(t)dem(m)}{c_l}$
9	$\sigma_{\max} \leftarrow \max_{l \in \mathcal{L}} \sigma_l$

TABLE III  
ONLINE ALGORITHM WITH PARTIAL TOPOLOGY KNOWLEDGE

The modified algorithm, as shown in Tab. III, does not alter the algorithm structure, but only redefine the edge length

update function (Line 7) from physical-link-based to overlay-edge-based. Here, the length of an overlay edge  $e$  depends on the following information: (1)  $|e|$ , the number of physical links contained in the unicast route of  $e$ , and (2)  $c_e$ , the capacity of the bottleneck physical link on this route. The length of  $e$  is initialized as a constant  $\beta$  timed by the length  $|e|$ , and normalized by its capacity  $c_e$ . This definition gives smaller lengths to those overlay edges with shorter unicast route and larger bottleneck capacity, which are in turn more likely to be chosen by the modified online algorithm.

To acquire the hop count  $|e|$  and bottleneck capacity  $c_e$  of the overlay edge  $e$ , we can rely on the following end-to-end measurement techniques.  $|e|$  can be found by network path finding tools such as traceroute. The bottleneck bandwidth along the unicast route of an overlay edge can be measured by tools such as Packet Pair, pathrate.

The following theorem shows the bound acquired in Theorem 2 stays unchanged with the new edge length update function.

**Theorem 3:** The modified online algorithm returns the competitive ratio bounded by  $\log |\mathcal{L}|$ .

### D. Discussion

Finally, we discuss the extensibility of the proposed algorithms to other challenges, mainly competition of multiple P2P streaming sessions, and churn (dynamic peer joining/leaving).

From an alternative perspective, we can view each layer as a multicast session with its subscribed receivers. Bearing this in mind, by revisiting the formulation of problem M, we find that the proposed algorithms can be easily extended to the case of multiple streaming sessions. More relevant results can be found in our previous work[10] addressing the competition of multiple multicast sessions.

Our algorithms also leaves extension space to address the dynamic peer joining/leaving scenario. The solution originates from the online nature of the algorithms introduced in this work, i.e., we can dynamically route traffic for any newly-arrived multicast sessions or peers. Our previous work on this subject [11] has shown that with peer churning, the performance bound established in Theorem 2 and 3 still holds.

## IV. SIMULATION

### A. Simulation Setup and Methodology

In this section, we experimentally evaluate the performance of our algorithms. The simulation is based on the topology generated by BRITE generator. We create a 1000-router topology, upon which 100 peers are randomly attached to. The bandwidth for each physical varies from 10 to 1024, with average of 518. Each router can only be attached by at most one client. The video stream to be distributed is divided into 5 layers. The demanding rate by each layer is 430, 370, 290, 280 and 170 respectively. Each layer has a set of peers as its receivers. In particular, we simulate the cumulative layer streaming scenario, i.e., a peer subscribing layer  $n + 1$  must

also subscribe layer  $n$  to recover the signal. Obviously, lower layers have more receivers.

We run all three algorithms presented in Sec. III. Especially, for online and modified online algorithms, we experiment with both single-tree and multi-tree approaches. We use mainly two metrics to evaluate the performance of the proposed algorithms: *throughput* and *capacity utilization*.

For each algorithm, we measure the throughput achieved per layer, and the overall throughput defined as the summary of end-to-end throughput received by all receivers across all layers. Note that since the optimal algorithm returns the maximally achievable rate for each layer subject to the weighted max-min fairness, we measure the performance of online algorithms in terms of throughput ratio, i.e., the throughput of online algorithm normalized by its counterpart by the optimal algorithm. In fact, this metric is reversion of the competitive ratio defined in Theorem 2.

Capacity utilization is measured as the utilization ratio (percentage of capacity utilized) for each physical link in the network. By sorting all links based on their utilization ratios, we are able to view a complete picture of the capacity utilization in the entire network.

### B. Performance Comparison

We first study the performance of the optimal algorithm. To well balance the tradeoff between approximation and running time overhead, we set the step size  $\epsilon = 0.01$ . From Fig. 3 (a), we see the throughput received across different layers are in proportion to the demands defined in the last subsection. Meanwhile, we also pay considerable cost to the tree management overhead, reflected in Fig. 3 (b), which shows number of trees required by each layer to achieve such throughput.

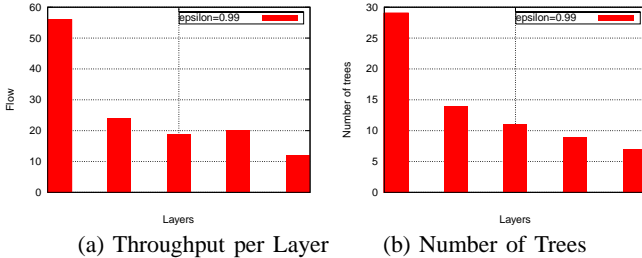


Fig. 3. Performance of the Optimal Algorithm ( $\epsilon = 0.01$ )

We then illustrate the performance of the online algorithm, starting with the throughput ratio relative to the optimal algorithm. In Fig. 4, we selectively show the results when we set the step size  $\epsilon$  to 10 and 100. A main observation obtain from both figures is that the performance of the online algorithm far exceeds the worst-cast bound established in Theorem 2. Also as we increase the number of trees for each layer from 1 to 4, we see reasonable throughput improvement. However, there is clear diminishing return as the number of trees per layer further grows. Interestingly, the overall throughput of the online algorithm is better than the optimal algorithm in

some cases. This is caused by the violation of weighted max-min fairness. As shown in both pictures, while the throughput of layers 3, 4, and 5 can outperform their counterparts for the optimal algorithm, layers 1 and 2 suffer from underperforming. In other words, the ideally fair online algorithm should achieve the same throughput ratio across all layers.

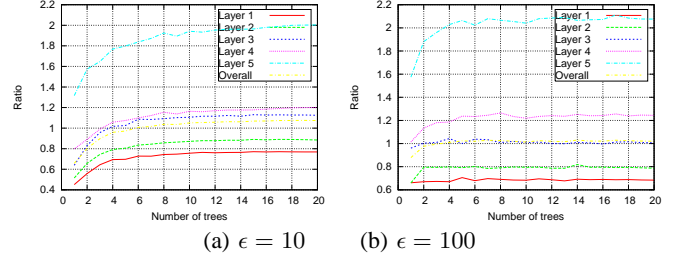


Fig. 4. Throughput of the Online Algorithm

We then selectively show the link utilization for different step size  $\epsilon$ , and number of trees per layer. In Fig. 5 (a), more than half of the links in the network is unutilized at all. In Fig. 5 (b), more links are utilized as we increase the number of trees. Evidently, having multiple trees help alleviate the problem of asymmetric capacity utilization.

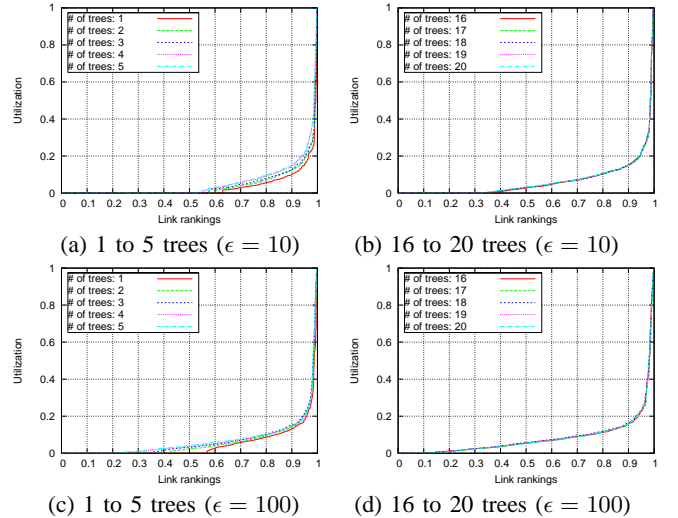


Fig. 5. Link Utilization of the Online Algorithm

When studying the throughput of the modified online algorithm, we notice the negative effect of having multiple trees. In Fig. 6 (a), the algorithm's performance reaches the highest as the number of trees per layer is 3, then starts to decline continuously. Same phenomenon is observed in Fig. 6 (b), when  $\epsilon = 100$ . The cause for this downfall is as follows. As the modified edge length update function cuts off the correlation among overlay edges, an overlay edge may seem empty or have few traffic, but in fact have heavy traffic occupied at some bottleneck links shared with other overlay edges. Unaware of this situation, a peer might still choose to route its traffic through this seemingly lightly-loaded route, which further

aggravate the situation. Routing traffic through multiple trees increases the chance of such events.

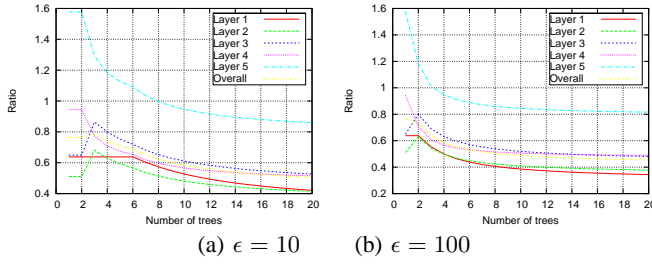


Fig. 6. Throughput of the Modified Online Algorithm

By studying the capacity utilization of the modified algorithm in Fig. 7, we also observe that the wrongfully-estimated load on the overlay edge will reuse the heavily-loaded physical link, thereby cause more physical links to be undiscovered and unutilized.

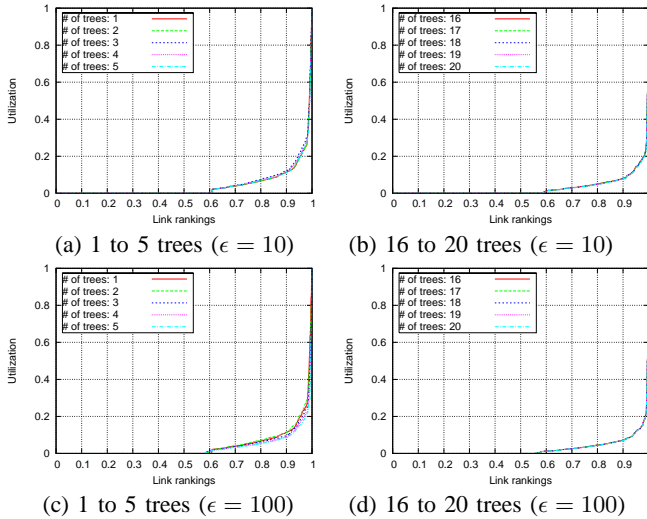


Fig. 7. Link Utilization of the Modified Online Algorithm

Nevertheless, the most important confirmation to the modified online algorithm, also evidenced in Fig. 6, is that the performance of this algorithm also far more exceeds its worst-case bound established in Theorem 3.

## V. RELATED WORK

The idea of assigning lengths to links to reflect their traffic conditions has been extensively explored in the domain of online unicast routing [9] [12] and multicast routing [13]. In this work, we further extend the idea to the domain of P2P network. Furthermore, we use a new weight update function different to the exponential function adopted by these solutions.

Several works propose overlay tree/network construction solutions that take into account the topology of the physical network. In TAG (Topology-Aware Grouping) [14], the information about overlap in routes to the sender among group members is used to guide the construction of overlay tree. In

[15], a distributed binning scheme is proposed where overlay nodes partition themselves into bins such that nodes that fall within a given bin are relatively close to one another in terms of network latency.

Finally, [16] has proposed solutions to build multiple disjoint overlay MSTs for each session. However, since the routing metric is defined as the static latency, not the dynamic weight to reflect the traffic condition, this solution does not have any performance guarantee.

## VI. CONCLUSION

In this paper, we target on optimal routing solution to maximize throughput under the fairness constraint in layered P2P streaming. We formulate the problem using the multicommodity flow theory. Based on this formulation, we first develop the optimal algorithm, then adapt it into two online algorithms, addressing certain practical issues as content splitting and partial topology knowledge. For the online algorithms, we further establish the worst-case approximation bound to the optimal rate. Experimental results confirm both online algorithms to greatly outperform their theoretical bounds.

## REFERENCES

- [1] S. Mccanne, V. Jacobson, and M. Vetterli, "Receiver-driven layered multicast," in *Proc. of ACM SIGCOMM*, 1996.
- [2] Y. Cui and K. Nahrstedt, "Layered peer-to-peer streaming," in *Proc. of ACM NOSSDAV*, 2003.
- [3] Nazanin Magharei and Reza Rejaie, "Adaptive receiver-driven streaming from multiple senders," *ACM/Springer Multimedia Systems Journal*, 2006.
- [4] Xinyan Zhang, Jiangchuan Liu, Bo Li, and Y. S. P. Yum, "Coolstreaming/donet: a data-driven overlay network for peer-to-peer live media streaming," in *Proc. of INFOCOM*, 2005.
- [5] Liang Dai and Yi Cui, "Maximizing throughput in layered peer-to-peer streaming," in <http://vanets.vuse.vanderbilt.edu/icc2007-report.pdf>, 2006.
- [6] N. Garg and J. Konemann, "Faster and simpler algorithms for multi-commodity flow and other fractional packing problems," in *Proc. of IEEE FOCS*, 1998.
- [7] L. K. Fleischer, "Approximating fractional multicommodity flow independent of the number of commodities," *SIAM Journal of Discrete Mathematics*, vol. 13, 2000.
- [8] R. M. Karp, "Reducibility among combinatorial problems," *Complexity of Computer Computations*, 1972.
- [9] J. Aspnes, Y. Azar, A. Fiat, S. Plotkin, and O. Waarts, "On-line routing of virtual circuits with applications to load balancing and machine scheduling," *Journal of ACM*, vol. 44, 1997.
- [10] Y. Cui, B. Li, and K. Nahrstedt, "On achieving optimized capacity utilization in application overlay networks with multiple competing sessions," in *Proc. of ACM Symposium on Parallel Algorithms and Architectures*, 2004.
- [11] Yi Cui and Klara Nahrstedt, "High-bandwidth routing in dynamic peer-to-peer streaming," in *P2PMMS*, 2005.
- [12] B. Awerbuch, Y. Azar, S. Plotkin, and O. Waarts, "Competitive routing of virtual circuits with unknown duration," in *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 1995.
- [13] A. Goel, M. Henzinger, and S. Plotkin, "Online throughput-competitive algorithm for multicast routing and admission control," in *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 1998.
- [14] M. Kwon and S. Fahmy, "Topology-aware overlay networks for group communication," in *ACM NOSSDAV*, 2002.
- [15] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, "Topologically-aware overlay construction and server selection," in *IEEE INFOCOM*, 2002.
- [16] A. Young, J. Chen, Z. Ma, A. Krishnamurthy, L. Peterson, and R. Y. Wang, "Overlay mesh construction using interleaved spanning trees," in *IEEE INFOCOM*, 2004.