

# On the Optimality of Layered Video Streaming Rate in a P2P Mesh Network

Tareq Hossain, Yi Cui, and Yuan Xue  
Vanderbilt Advanced NETWORK Systems Laboratory  
Department of Electrical Engineering and Computer Science  
Vanderbilt University, Nashville, TN 37212  
Email: {tareq.hossain, yi.cui, yuan.xue}@vanderbilt.edu

**Abstract**—Layered streaming is an effective solution to address receiver heterogeneity in a Peer-to-peer (P2P) network. This paper addresses the problem of rate allocation for layered video stream in p2p mesh networks. We present a distributed rate allocation algorithm that achieves close to optimal layer allocation among peers receiving the same video. We use load balancing technique that evenly distributes layer requests to a parent peer. Weight based layer allocation technique ensures that a child peer with higher children receives higher priority in servicing layer allocation request from its parents. Results show that for up to seven layers streamed by a server, the algorithm maintains a layer delivery ratio of 88% or more among all peers in a network.

## I. INTRODUCTION

Recent advancement in the field of video compression and networking technologies have resulted in wide deployment of video distribution applications. These applications enable the end users with ubiquitous accessibility to media streaming services such as live broadcasting, VoD, video conferencing, etc. Peer-to-peer is one such powerful technology platform for a variety of multimedia streaming applications over the Internet. The P2P's success is due to its instant deployability; it allows almost ubiquitous network coverage in the absence of CDN services and IP multicast. In addition, a P2P system is extremely cost-effective since it utilizes the resources (CPU cycles, storage space, and uplink bandwidth) of peer machines.

Existing approaches for P2P streaming can be divided into two classes: *tree based* and *mesh based*. The tree based approach extends the idea of end-system multicast [1]. A mesh based P2P streaming is derived from file swarming mechanisms (such as BitTorrent) where participating peers form a randomly connected mesh. Due to its multiple incoming connection for each peer, a multi-path mesh can fully utilize the network resources of its peers. Furthermore, peers experience a higher degree of stability [2] in a mesh based network compared to a tree based network. This results in a higher quality for the delivered video.

Typically, P2P applications lack QoS due to missing provisioning mechanisms by the underlying network. Therefore, any solution to cope with such network congestion requires rate adaptation of the data stream. Layered video multicast allows us to avoid network congestion [3]. Layered encoding compresses raw video sequences into non-overlapped layers, where each layer depends on all of the previous layers. The *base layer* contains data that constitutes the most important

features of the video. Additional layers, called *enhancement layers*, contain data that progressively adapts the reconstructed video quality in a fast and easy way to changing network conditions or specific capabilities of the end users [4].

The inherent heterogeneity in receiver capabilities of a P2P network requires that the streaming quality be different for each peer. In the case of simulcast, the heterogeneity of a P2P network requires different bit rates of video and consequently different streams. This forces the overlay network to divide into smaller sub-groups with each receiving only one version of the stream and therefore, becomes vulnerable to bandwidth fluctuation and peer churning. Layered video overcomes this by offering one video stream that can serve a variety of bit rates. However, this increased flexibility also produces a data overhead that reduces the coding efficiency of the video stream. Therefore, rate-optimization becomes highly important to achieve desired video quality across the network.

Several schemes have been proposed that addresses scheduling [5], route-optimization [6], and rate-optimization [7] issues in layered streaming. Our previous work has already proved that layer allocation in P2P streaming is NP-complete [7]. Therefore, in this paper, we focus on two metrics that allow us to achieve close to optimal solution. Specifically, our algorithm focuses on the *load balancing* and *weight based layer allocation* to achieve high layer allocation among peers in the network. In load balancing, a child peer evenly distributes the layer request among all available parents. This allows a parent peer to serve multiple child peers. In weight based layer allocation, a parent prioritizes layer allocation among its children based on the number of descendants each child peer have.

The remainder of this paper is organized as follows. In Sec. II, we present related works. Sec. III describes the network model, rate allocation and optimization constraints are described in Sec. IV. Sec. V presents our distributed algorithm. In Sec. VI we present the simulation results and conclude in Sec. VII.

## II. RELATED WORK

McCanne, et al. [3] first proposed a receiver-driven layered streaming approach to address network heterogeneity. Ross, Saporilla and Cuestos [8], [9] proposed rate-optimization algorithm for layer based videos in a P2P network. The strategy

calls for dividing available bandwidth between the base-layer and the enhancement-layer of a two-layered stream. The authors conclude that heuristics which take into account the amount of data remaining in the pre-fetch buffer outperform static division of bandwidth. They also conclude that maximizing just the overall amount of data stream causes undesirable fluctuations in quality, and provide additional heuristics that produce a smoother stream while minimally reducing the overall quality.

Rajendran, et al. [10] extended this work by proposing an on-line pre-fetch algorithm with  $N$  number of layers. The quality measurement of this algorithm also takes into account first and second order variations, borne out in subjective experiments mentioned in [11]. The algorithm pre-fetches layers of future portions of the video in small chunks, as earlier portions are being played back, with the aim of using less bandwidth, increased smoothness, and decreased variability of the number of layers used.

Liu, et al. [12] proposed a heuristic algorithm that optimizes the layer by allocating from the base layer to the enhancement layers in turn and by allocating each layer from a suitable supplier. Padmanabhan, et al. [13] introduced multiple distribution trees to make live media streaming robust in a heterogeneous peer-to-peer network. Liu, et al. [14] used an incentive based algorithm to optimize the rate allocation in live P2P streaming.

Finally, Rajaie, et al. [5] proposes a framework for P2P adaptive layered streaming. This framework proposes a receiver driven approach where the receiver coordinates the delivery of layers from active senders by adaptively determining a subset of senders that maximizes throughput and overall quality (i.e. the number of layers).

### III. NETWORK MODEL

We consider a P2P network consisting of  $H$  end hosts, denoted as  $\mathcal{H} = \{h_i | i = 0, 1, 2, \dots, H\}$ . Host  $h_0$  acts as the server, and other end hosts are peers. We define a set of layers  $\mathcal{L} = \{l_i | i = 1, 2, \dots, L\}$  and the rate assigned to a particular layer  $l$  is  $x_l$ . We also define a flow  $f_{ij}$  that can direct from a peer  $h_i$  to another peer  $h_j$  except from a peer to the server  $h_0$ . If  $h_i$  is a parent of  $h_j$  and  $h_j$  is a child of  $h_i$ , the relationship between the two peers is denoted as  $h_i \rightarrow h_j$ . We also collect all these flows into the set  $\mathcal{F} = \{0, 1, 2, \dots, F\}$ . The set of layers in flow  $f$  is defined as  $\mathcal{L}(f)$ . Therefore, each flow  $f$  has a rate  $x_f$ :

$$x_f = \sum_{l \in \mathcal{L}(f)} x_l \quad (1)$$

For a peer  $h_i \in \mathcal{H}$ , we define  $\mathcal{S}_i^f = \{f_{ij} | \forall j, h_i \rightarrow h_j\}$  to be the set of flows directed to its child peers and  $\mathcal{S}_i^h = \{h_j | \forall j, h_i \rightarrow h_j\}$  to be the set of its child peers. Similarly, we define  $\mathcal{R}_i^f = \{f_{ki} | \forall k, h_k \rightarrow h_i\}$  to be the set of flows directed to  $h_i$  from its parents and  $\mathcal{R}_i^h = \{h_j | \forall j, h_j \rightarrow h_i\}$  to be the set of its parent peers. Therefore, the set of layers received by a peer  $h$  from all its parents is defined as  $\mathcal{L}(h) = \cup_{f \in \mathcal{R}^f} \mathcal{L}(f)$  and the total number of layers is  $\lambda = |\mathcal{L}(h)|$ .

For the peer  $h$ , the total streaming rate  $x_h$  received from all parents in a P2P mesh is then defined as:

$$x_h = \sum_{l \in \mathcal{L}(h)} x_l \quad (2)$$

Each flow  $f$  takes place on the unicast path that connects two peers and encompasses a set of physical links on the Internet. We collect all physical links encompassed by all flows into a vector as  $\mathcal{P} = \{p_i | i = 0, 1, 2, \dots, P\}$ . The bandwidth of each link is defined as  $c_p$ , which is collected into a capacity vector  $\mathbf{c} = (c_p, p \in \mathcal{P})$ . Based on this definition, for each link  $p$ , we define its flow set  $\mathcal{F}(p) = \{f \in \mathcal{F} | p \in \mathcal{P}\}$  as the set of flows that pass through it.

Due to the capacity constraint on each link  $p$ , the total rate for all the flows in  $\mathcal{F}(p)$  cannot exceed the link capacity  $c_p$ , i.e.,  $\sum_{f \in \mathcal{F}(p)} x_f \leq c_p$ . Formally, let  $\mathbf{A}$  be an  $P \times F$  matrix, such that  $A_{pf} = 1$  if flow  $f$  goes through the link  $p$ , otherwise  $A_{pf} = 0$ .

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

In this paper, our physical topology is based on the assumption [15] that the bottleneck link of an end-to-end connection only happens at the uplink of the sending peer. As such, since the capacity constraint of other links never gets violated, they can be removed from the above inequality. This effectively reduces the link set  $\mathcal{P}$  to only contain the uplinks of all peers and the server. As illustrated in Fig. 1(a), we give such a special topology the term *star topology*. Under this topology, the size of the link set  $\mathcal{P}$  equals the size of the end host set  $\mathcal{H}$ .

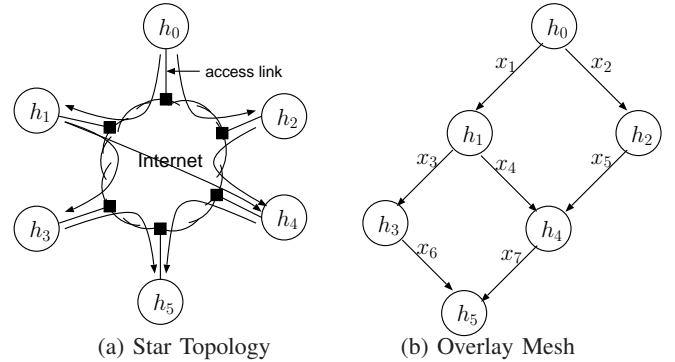


Fig. 1. P2P Streaming Illustration

We collect the notations used in this section in Tab. I

### IV. RATE ALLOCATION

Consider peers  $h_i$  and  $h_j$  such that  $h_i \rightarrow h_j$ , we define  $\sigma$  to be a binary variable with the following properties:

$$\sigma_{ij}^l = \begin{cases} 1 & \text{if layer } l \text{ is present in } f_{ij} \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

TABLE I  
NETWORK MODEL NOTATIONS

Notation	Definition
$h \in \mathcal{H} = \{1, 2, \dots, H\}$	End host
$f \in \mathcal{F} = \{1, 2, \dots, F\}$	Unicast flow in overlay multicast
$p \in \mathcal{P} = \{1, 2, \dots, P\}$	Physical network links
$l \in \mathcal{L} = \{1, 2, \dots, L\}$	Streaming layers
$x_f, f \in \mathcal{F}$	Flow rate of $f \in \mathcal{F}$
$x_h, h \in \mathcal{H}$	Total streaming rates from parents
$x_l, l \in \mathcal{L}$	The rate required to stream layer $l$
$c = (c_p, p \in \mathcal{P})$	Link capacity of vector
$h_k \rightarrow h_i$	Host $h_i$ is the child of host $h_k$
$\mathcal{L}(f) \subseteq \mathcal{L}$	Set of layers in flow $f$
$\mathcal{S}_i^h, \mathcal{S}_i^f$	Set of child peers and outgoing flows for peer $h_i$
$\mathcal{R}_i^h, \mathcal{R}_i^f$	Set of parent peers and incoming flows for peer $h_i$
$A = (A_{pf})_{P \times F}$	Link Capacity Constraint Matrix

The objective function<sup>1</sup> adopted based on the definition in Eq. (2) is given as:

$$\max. \sum_{h \in \mathcal{H}} x_h \quad (5)$$

$$\text{subject to} \quad \sum_{j=1}^H \left( \sum_{l=1}^L \sigma_{ij}^l x_l \right) \leq c_{hi} \quad (6)$$

$$\sum_{i=1}^H \left[ \sigma_{ij}^l - \sigma_{ij}^{l-1} \right] \leq 0 \quad (7)$$

$$\sum_{i=1}^H \sigma_{ij}^l \geq \sigma_{jk}^l \quad (8)$$

$$\sum_{i=1}^H \sigma_{ij}^l \leq 1 \quad (9)$$

where Eq. (5) is a non-linear function of the total allocated rate for all the peers in the network. The rate  $x_l$  for each layer is derived from the set  $\mathcal{L}$  based on a constant rate that can be generated empirically.

Eq. (6) states the capacity constraints for each peer. Eq. (7) states that the layers received by a peer from all its parents in  $\mathcal{L}(h)$  are consecutive. For example, if a peer has layer 3, the peer must have layer 1 and layer 2. Eq. (8) states that a peer cannot receive a particular layer if none of its parents have that layer. Finally the duplicity constraint in Eq. (9) states that a peer should not receive same layer from multiple parents.

## V. DISTRIBUTED ALGORITHM

We start with a high level description of our algorithm followed by detailed discussions of various aspects.

<sup>1</sup>For simplicity purposes, we assume that the objective function excludes the rate distortion function for the server  $h_0$ . This can be easily achieved by assigning the value 0 to the rate distortion function of  $h_0$ .

TABLE II  
LAYER-GEN: VALID LAYER GENERATION

---

```

For each end host  $h_j \in \mathcal{H}$ 
1:  $\mathcal{N} = \emptyset$ 
2:
3: for all  $\omega \in \Omega_i$  do
4:    $\Omega_i \leftarrow \Omega_i - \omega$ 
5:    $matched = 1$ 
6:    $totalLayers = 0$ 
7:   for all  $\delta_i \in \omega$  do
8:     if  $\sum_{l=1}^{\lambda_i} x_i \sigma_{ij}^l \leq c_i$  then
9:        $\omega \leftarrow \omega - \delta_i$ 
10:      end if
11:    end for
12:     $l = 0$ 
13:    while (1) do
14:       $numParents = 0$ 
15:       $layers = 0$ 
16:      for all  $\delta_i \in \omega$  such that  $\delta_i \in M^{h_i}, h_i \in \mathcal{R}_j^h$  do
17:        if  $l < \lambda_i$  then
18:           $layers++ = \sigma_{ij}^l$ 
19:           $numParent ++$ 
20:        end if
21:      end for
22:      if  $layers > 1$  or  $l \geq totalLayers + 1$  then
23:         $matched = 0$ 
24:        break
25:      end if
26:      if  $numParents == 0$  then
27:        break
28:      end if
29:       $totalLayers++ = layers$ 
30:       $l ++$ 
31:    end while
32:    if  $matched$  then
33:       $\mathcal{N} \leftarrow \mathcal{N} + \omega$ 
34:    end if
35:  end for

```

---

A child initially inquires about the number of layers available from a parent peer. It then generates a valid combination of layers, ranks them, and sends this layer combination information to its parents. After receiving a layer allocation request from its children, a parent allocates a combination of layers that maximizes the utilization of its uplink bandwidth. The layer allocation is based on the ranking information provided by its children.

We now describe the algorithm in detail:

### A. Layer Combination Generations

Each peer generates unique layer combinations for its parent based on the available layers information received. For a peer  $h_j$  with parents  $h_i$  having  $\lambda_i$  number of layers, there exist a total of  $2^{\lambda_i}$  layer combinations  $h_j$  can request from  $h_i$ . We collect these combination vectors in a set  $M^{h_i} = \{\delta_i^m \mid m = 1, 2, \dots, 2^{\lambda_i}; \delta = [\sigma_{ij}^1, \sigma_{ij}^2, \dots, \sigma_{ij}^l, \dots, \sigma_{ij}^{\lambda_i}]\}$ . The vector  $\delta$  is a binary value vector that determines the presence of a layer between a parent and a child as mentioned in Eq. (4). For each peer  $h_j$ , let  $n = |\mathcal{R}_j^h|$  be the number of parents, where  $h_i \in \mathcal{R}_j^h$ . We now define  $\Omega_j = (M^{h_1} \times M^{h_2} \times \dots \times M^{h_i} \times \dots \times M^{h_n})$  be an  $n$ -tuple, where each element set  $\omega \in \Omega = \{\delta^1 \in M^{h_1}, \delta^2 \in M^{h_2}, \dots, \delta^n \in M^{h_n}\}$  is a set of possible layer combination

vectors from its parents. The algorithm in Tab. II generates valid layer combinations for a peer  $h_j$  based on its parent set  $\mathcal{R}_j^h$ . The *for loop* in line 7 eliminates any layer combinations that violate the capacity constraints of a parent peer mentioned in Eq. (6). The condition in line 22 eliminates discontinuity of Eq. (7) and duplicity of Eq. (9) in the layer combinations generated. The *for loop* in line 16 ensures that a layer received by a child must be present among its parent mentioned in Eq. (8).

The resulting set  $\mathcal{N}$  consists of all the valid layer combinations that satisfy the constraints from Eq. (6)-(9). Even though each parent  $h_i$  with  $\lambda_i$  number of layers generates  $2^{\lambda_i}$  possible combinations, the total number of valid combinations of layers in set  $\mathcal{N}$  is significantly smaller after satisfying all the constraints.

### B. Preference Ranking

After generating all the valid layer combinations in  $\mathcal{N}$ , a child peer  $h_j$  then uses *rate maximization* and *load balancing* methodology to rank the layer combinations. Since a rank  $r$  is assigned to each  $\omega$ , each layer combination  $\delta \in \omega$  also carries this rank value.

First we define the set of empty layer combination vectors in  $\omega \in \mathcal{N}$ . The set is defined as  $\Theta \subseteq \omega$ , such that  $\Theta = \{\delta \mid \delta = \emptyset\}$ . The set  $\mathcal{N} = \{\omega_1, \omega_2, \dots\}$  is ranked  $\{r_1, r_2, \dots\}$  based on the following preferences:

1) *Rate Maximization*: In order to ensure that each peer receives maximum number of layers possible, a child peer ranks the elements of  $\mathcal{N}$  such that the combination that delivers higher number of layers has lower ranking compared to a combination that delivers lower number of layers. Formally, this condition can be expressed by:

$$\sum_{\delta_i \in \omega_1} \sum_{l=1}^{\lambda_i} \sigma_{ij}^l \geq \sum_{\delta_i \in \omega_2} \sum_{l=1}^{\lambda_i} \sigma_{ij}^l \quad (10)$$

2) *Load Balancing*: If two layer combination vectors,  $\omega_1$  and  $\omega_2$  deliver the same number of layers, we use load balancing technique to rank them such that layer combination request by a child peer will be distributed among as many parents as possible. Therefore, layer combinations that can evenly distribute the layer requests among multiple parents are given higher preference (i.e., lower ranking). This is achieved by the following load balancing equation:

$$\sum_{\delta_i \in \omega_1} \frac{\sum_{l=1}^{\lambda_i} \sigma_{ij}^l x_l}{c_{h_i}} (|\Theta_1| + 1) \leq \sum_{\delta_i \in \omega_2} \frac{\sum_{l=1}^{\lambda_i} \sigma_{ij}^l x_l}{c_{h_i}} (|\Theta_2| + 1) \quad (11)$$

We use the number of empty layer combinations vector  $\delta = \emptyset$  as a metric to assign ranks to each layer combination set. Therefore, a set  $\omega$  with higher number of empty layer combinations requested from parents receives a lower preference (i.e., higher ranking).

### C. Weight Based Layer Allocation

Maximizing the total number of layers received by peers across the network depends on the proper allocation of layers by each parent such that a child with fewer children receives fewer layers compared to another child. However, a child connected to fewer parent should receive more layers compared to a child with higher number of parents. In a mesh topology, a child can always receive more layers from any of its parents. Therefore, in order ensure fairness, a parent peer must consider both the incoming and outgoing degree of a child peer when allocating layers. For a peer  $h_j$ , we define the cumulative incoming and outgoing links as  $\alpha_j$  and  $\beta_j$ . Formally, they are defined as:

$$\alpha_j = |\mathcal{R}_j^h| + \sum_{h_k \in \mathcal{S}_j^h} \alpha_k \quad (12)$$

$$\beta_j = |\mathcal{S}_j^h| + \sum_{h_k \in \mathcal{S}_j^h} \beta_k \quad (13)$$

Therefore, the *link ratio* for each peer  $h_j$  is:

$$\ell_j = \eta \cdot \frac{\alpha_j - \beta_j}{\alpha_j + \beta_j + 1} \quad (14)$$

where  $\eta$  is a proportionality constant that can be adjusted to assign priorities to children. A child peer with lower  $\ell$  has higher priority in receiving its preferred layers from its parents. A parent then distributes its available uplink bandwidth based on this ratio. Fairness is ensured in this process because a peer with fewer child will receive fewer layers than a peer with more child.

However, this process could lead to some child peers receiving no layers from its parents. To prevent starvation, we define a *link-ratio band*  $\gamma$  such that a peer  $h_j$  having two child peer  $h_i$  and  $h_k$  with rank  $r_i$  and  $r_k$  has the following property:

$$\lceil (\ell_i - \ell_k) / \gamma \rceil \geq r_i - r_k \quad (15)$$

where  $\ell_i > \ell_k$ . Therefore, if  $\lceil (\ell_i - \ell_k) / \gamma \rceil$  is 2 and  $r_k$  is 1, then peer  $h_i$  can expect a layer combination assignment having rank at least 3. If  $h_j$  were to assign layer combination having rank 4 to  $h_i$  then  $r_k$  must be lowered to rank 2.

### D. Preference Matching

For a child peer, if all the layers received from parents have the same preference rank  $r$ , then there will be no duplicate layers. The parents initially attempt to accommodate the layers with the lowest rank to its children. A child receiving layers with higher rank implies that the parent is capacity limited and has other children with lower link ratio i.e., receives priority in layer allocation.

In this situation, a child determines a layer combination rank that satisfies the existing layer rate allocation constraint by its parents. A parent only allocates a requested layer combination if the bandwidth required to service all the layers in the combination is less than what is allocated for that particular child based on its link ratio.

TABLE III  
DISTRIBUTED ALGORITHM & SIMULATION NOTATIONS

Notation	Definition
$\lambda_i$	Number of layers peer $h_i$ has
$\delta$	Binary layer combination vector
$\mathcal{M}^{h_i}$	Set of layer combination vector $\delta$ for peer $h_i$
$\Omega_i$	n-tuple
$\omega \in \Omega$	Each element in n-tuple with layer combination from parents
$\mathcal{N}$	Set of valid layer combinations
$\Theta_i \subseteq \omega_i$	Set containing empty layer combinations
$r$	Ranks assigned to layer combination set
$\alpha, \beta$	Cumulative incoming, outgoing links
$\ell, \eta$	Link-ratio, proportionality const.
$\gamma$	Link-ratio band
$s_h^c$	Spare-capacity coefficient
$s_h^b$	Bandwidth coefficient
$x_{\mathcal{S}(h)}$	Total outgoing rate
$x_{\mathcal{R}(h)}$	Total incoming rate

### E. Layer Rate Update

A child peer periodically communicates with its parents to determine the availability of layers. The layer allocation by a parent also changes if the link ratio of its children or the available uplink bandwidth changes.

We collect the notations used in this section and the following section in Tab. III.

## VI. SIMULATION

In this section, we present simulation results. We start by presenting our mesh network construction procedure.

### A. Mesh Construction

Here we present a simple mesh construction procedure in order to test our algorithm. However, our solution is topology independent and works with any mesh construction solution. When a new peer wishes to join the network, we use the spare capacity information of the existing peers to determine a suitable parent. Formally, for each peer  $h \in \mathcal{H}$ , we define a *spare-capacity coefficient*<sup>2</sup> as:

$$s_h^c = [c_h - \sum_{f \in \mathcal{S}^f} x_f]_0^{x_{\mathcal{R}(h)}} \quad (16)$$

where  $x_{\mathcal{R}(h)} = \sum_{f \in \mathcal{R}^f} x_f$  is the total incoming flow rate of peer  $h$ . The peers also calculate their *bandwidth-coefficient*<sup>3</sup> to decide if they should seek multiple parents. Formally, this ratio is defined as:

$$s_h^b = \frac{x_{\mathcal{R}(h)}}{x_{\mathcal{S}(h)}} \quad (17)$$

<sup>2</sup>The upper-bound in Eq. (16) does not apply to the server  $h_0$  as its  $s_h^c$  is the difference between its capacity and the sum of its child flow rates

<sup>3</sup>The server has a bandwidth-coefficient of 0. It can be thought of as the seed in a torrent network

Here  $x_{\mathcal{S}(h)}$  defines the total outgoing rate. A peer having a  $s_h^b < 1$  implies that it dedicates more bandwidth to its children than it receives. In order to address this imbalance, a peer seeks a parent with  $s_h^b > 1$ , i.e., a peer that receives more than it gives. During the simulation, we ensure that this process does not create a cycle in the mesh.

At the end of each rate update cycle, leaf peers send the spare-capacity coefficient information and the link-ratio information to their parents. A parent then decides the best candidate with the highest coefficient value and sends this information along with the bandwidth-coefficient to its own parent. The ID of the candidate with highest spare capacity eventually propagates to the server. The server also receives bandwidth-coefficient information of all the peers. The server then provides a new peer with the parent information, so it can join the network as a child of that parent. It also provides existing peers with the ID of peers with  $s_h^b > 1$  such that it does not create a cycle.

The following procedures formally summarize the mesh construction process:

- **Rate Update:** At the end of each layer rate allocation, a peer updates its bandwidth-coefficient and spare-capacity coefficient, i.e., the maximum rate it can provide to a new child. Peers then send this information to their parent.
- **ID Peer:** If child peers exist, a parent peer decides the best candidate based on its own spare coefficient and the spare coefficient of its children. It then sends this information to its own parent. This information eventually reaches the server.
- **Join:** The new peer sends a join request to the server. After receiving the parent information, the new peer joins the mesh as a child of the peer that maximizes its own receiving rate.
- **Req. Parent:** A peer wishing to have more than one parent sends a request through its parent to the server. The server then replies with an appropriate parent ID.

After each *join* operation, all peers perform the *rate update* and *ID peer* operation to decide the new best parent candidate for future peers.

### B. Results

We use a streaming server that provides 7 layers each at 250Kbps. The bandwidth of each peer is randomly assigned between 2Mbps and 3Mbps. During the course of the simulation, *in-degree/out-degree* ratio and the value of  $\eta$  for each peer is kept to 1.

Fig. 3 shows the average layer delivery ratio for various number of layers streamed by the server. Even as the number of available streaming layers increases, the average layer delivery ratio continues to remain close to or above 90%.

Fig. 2 shows the average delivery ratio for various layers as the number of nodes in the network increases.

## VII. CONCLUSION

In this paper, we present a layer allocation algorithm for a P2P mesh network that targets to achieve close to optimal rate

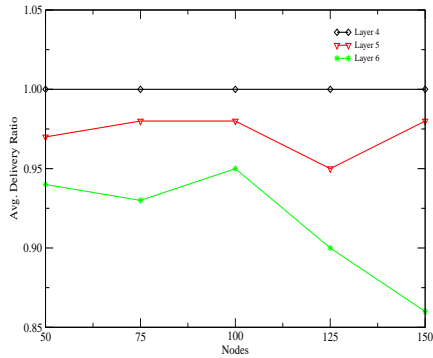


Fig. 2. Average layer delivery ratio for different number of nodes

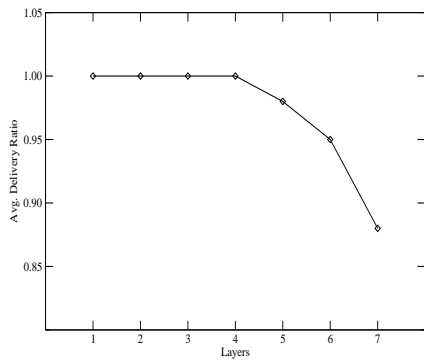


Fig. 3. Average layer delivery ratio for different number of layers with 100 nodes

among peers. We consider load balancing and weight based layer allocation. This ensures that a child evenly distributes layer allocation request among all its parents and a parent allocates higher rates to a child that in turn has more children than other child. Simulation shows that for up to 7 layers, the algorithm achieves a layer delivery ratio of 90% more.

#### REFERENCES

- [1] Y. Chu, R. Rao, and H. Zhang, "A case for end system multicast," *In ACM SIGMETRICS*, 2000.
- [2] N. Magharei, R. Rejaie, and Y. Guo, "Mesh or multiple-tree: A comparative study of live p2p streaming approaches," in *IEEE INFOCOM*, 2007.
- [3] S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-driven layered multicast," *Proc. SIGCOMM*, vol. 26, pp. 117–130, 1996.
- [4] J. Liu, B. Li, and Y. Zhang, "Adaptive video multicast over the internet," *In IEEE Multimedia*, vol. 10, 2003.
- [5] R. Rajaie and A. Ortega, "Pals: Peer-to-peer adaptive layered streaming," *NOSSDAV*, 2003.
- [6] L. Dai, Y. Cui, and Y. Xue, "Maximizing throughput in layered peer-to-peer streaming," *In Proc. IEEE ICC*, 2007.
- [7] Y. Cui and K. Nahrstedt, "Layered peer-to-peer streaming," *In Proc. NOSSDAV*, 2003.
- [8] P. Cuetos and K. Ross, "Adaptive rate control for streaming stored fine-grained scalable video," *NOSSDAV*, pp. 3–12, 2002.
- [9] D. Saporilla and K. Ross, "Streaming stored continuous media over fair-share bandwidth," *NOSSDAV*, 2002.

- [10] R. Rajendran and D. Rebenstein, "Optimizing the quality of scalable video streams on p2p networks," *Elsevier Science*, 2006.
- [11] M. Zink, O. Kunzel, J. Schmitt, and R. Steinmetz, "Subjective impression of variations in layer encoded videos," *international Workshop on Quality of Service*, 2003.
- [12] Y. Liu, W. Dou, and Z. Liu, "Layer allocation algorithm in layered peer-to-peer streaming," *International Federation for Information Processing*, 2004.
- [13] V. Padmanabhan, W. Helen, and P. Chou, "Resilient peer-to-peer streaming," *ICNP*, 2003.
- [14] Z. Liu, Y. Shen, S. Panwar, K. Ross, and Y. Wang, "Using layered video to provide incentives in p2p live streaming," *P2P-TV*, 2007.
- [15] M. Chen, M. Ponc, S. Sengupta, J. Li, and P. A. Chou, "Utility maximization in peer-to-peer systems," in *SIGMETRICS '08: Proceedings of the 2008 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*. ACM, 2008, pp. 169–180.