Optimizing the Throughput of Data-Driven Peer-to-Peer Streaming

Meng Zhang, *Student Member*, *IEEE*, Yongqiang Xiong, *Member*, *IEEE*,
Qian Zhang, *Senior Member*, *IEEE*, Lifeng Sun, *Member*, *IEEE*, and Shiqiang Yang, *Member*, *IEEE*

Abstract—During recent years, the Internet has witnessed a rapid growth in deployment of data-driven (or swarming based) peer-to-peer (P2P) media streaming. In these applications, each node independently selects some other nodes as its neighbors (i.e., gossip-style overlay construction) and exchanges streaming data with the neighbors (i.e., data scheduling). To improve the performance of such protocol, many existing works focus on the gossip-style overlay construction issue. However, few of them concentrate on optimizing the streaming data scheduling to maximize the throughput of a constructed overlay. In this paper, we analytically study the scheduling problem in data-driven streaming system and model it as a classical min-cost network flow problem. We then propose both the global optimal scheduling scheme and distributed heuristic algorithm to optimize the system throughput. Furthermore, we introduce layered video coding into data-driven protocol and extend our algorithm to deal with the end-host heterogeneity. The results of simulation with the real-world traces indicate that our distributed algorithm significantly outperforms conventional ad hoc scheduling strategies especially in stringent buffer and bandwidth constraints.

Index Terms—Peer-to-peer, data driven, block scheduling, min cost flow, throughput, delivery ratio.

1 Introduction

The Internet has witnessed a rapid growth in deployment of data-driven or swarming-based peer-to-peer streaming systems (P2P streaming systems) from the year of 2005, such as [1], [2], and [3], since the data-driven protocol is first proposed in academia [4], [5], [6], [7], [8]. Such protocol achieves great success mainly due to the good scalability, as well as the robustness to the high churn of the participating nodes. Recent exciting reports show that systems based on this type of protocol have the power to enable over 230,000 users simultaneously watching a hot live event with 300-500 kilobits per second (Kbps) by only one streaming server on the global Internet [2], [1].

The basic idea of data-driven streaming protocol is very simple and similar to that of Bit-Torrent [9]. The protocol contains two steps. In the first step, each node independently selects its neighbors so as to form an unstructured overlay network, called the gossip-style *overlay construction* or membership management. The second step is named *block scheduling*: The live media content is divided into blocks (or segments or packets), and every node announces what blocks it has to its neighbors. Then each node explicitly requests the blocks of interest from its neighbors according to their announcement. Obviously, the performance of data-driven protocol directly relies on the algorithms in these two steps.

To improve the performance of data-driven protocol, most of the recent papers focused on the construction problem of the first step. Researchers proposed different schemes to build unstructured overlays to improve its efficiency or robustness [10], [11], [12]. However, the second step, i.e., the block scheduling has not been well discussed in the literature yet. The scheduling methods used in most of the pioneering works with respect to the data-driven/swarming-based streaming are somewhat ad hoc. These conventional scheduling strategies mainly include pure random strategy [4], local rarest first (LRF) strategy [6] and round-robin strategy [5]. Actually, *how to do optimal block scheduling to maximize the throughput of data-driven streaming under a constructed overlay network* is a challenge issue.

In this paper, we present our analytical model and corresponding solutions to tackle the block scheduling problem in data-driven protocol. We first model this scheduling problem as a classical min-cost network flow problem and propose a global optimal solution in order to find out the ideal throughput improvement in theory. Since this solution is centralized and requires global knowledge, based on its basic idea, we then propose a heuristic algorithm that is fully distributed and asynchronous with only local information exchange. Furthermore, we employ layered video coding to encode the video into multiple rates and extend our algorithm to improve the satisfaction of the users with heterogeneous access bandwidth. Simulation results indicate that our distributed algorithm significantly outperforms other different conventional ad hoc scheduling strategy gains in both of the single rate and multirate scenarios.

- *M. Zhang, L. Sun, and S. Yang are with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, P.R. China. E-mail: zhangmeng00@mails.tsinghua.edu.cn, {sunlf, yangshq}@tsinghua.edu.cn.*
- *Y. Xiong is with Microsoft Research Asia, Beijing 100080, P.R. China. E-mail: yqx@microsoft.com.*
- *Q. Zhang is with the Department of Computer Science, Hong Kong University of Science and Technology, Hong Kong. E-mail: qianzh@cs.ust.hk.*

Manuscript received 2 Nov. 2007; revised 23 Feb. 2008; accepted 10 Mar. 2008; published online 24 Apr. 2008.
Recommended for acceptance by A. Boukerche.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-2007-11-0404.
Digital Object Identifier no. 10.1109/TPDS.2008.69.

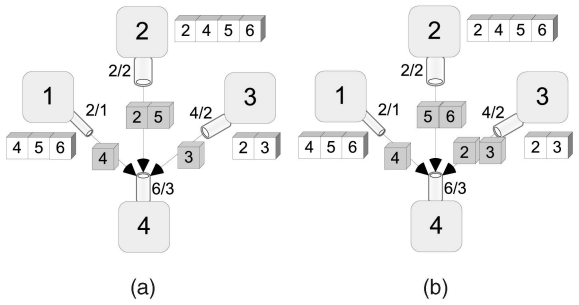


Fig. 1. Illustration of the block scheduling problem (I). (a) LRF scheduling. (b) Optimal scheduling.

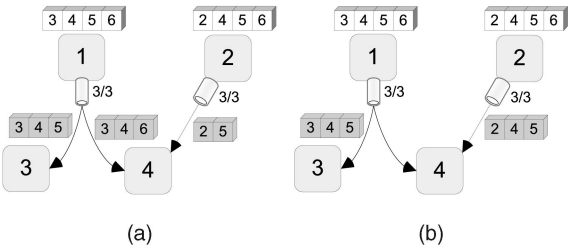


Fig. 2. Illustration of block scheduling problem (II). (a) LRF scheduling. (b) Optimal scheduling.

The remainder of this paper is organized as follows: Section 2 presents our model for the block scheduling problem and gives the global optimal scheduling solution. Section 3 describes our heuristic distributed algorithm. Then, in Section 4, we employ layered video coding and extend our algorithm for multiple streaming rates. In Section 5, we conduct simulations to evaluate the performance of our algorithm. The related work of this paper is briefly reviewed in Section 6. Finally, we conclude this paper and give our future work in Section 7.

2 Block Scheduling: Problem Statement and Formulation

First of all, we briefly review the data-driven protocol here. The idea of data-driven peer-to-peer streaming system is very similar to that of Bit-Torrent protocol [9]. In such protocol, each node independently finds its neighbors in the overlay so that an unstructured random overlay mesh is formed. The media streaming is divided into blocks with the equal size, each of which has a unique sequence number. Every node has a *sliding window*, which contains all the up-to-date blocks on the node and goes forward continuously at the speed of streaming rate. We call the front part of the sliding window *exchanging window*. The blocks in the exchanging window are the ones before the playback deadline, and only these blocks is requested if they are not received. The unavailable blocks beyond playback deadline will be no more requested. The blocks that have been played are buffered in the sliding window, and they can be requested by other nodes. Every node periodically pushes all its neighbors a bit vector called buffer map in which each bit represents the availability of a block in its sliding window to announce what blocks it holds. Due to the announcement of the neighbors, each node periodically sends requests to its neighbors for the desired blocks in its exchanging window. We call the time between two requests a *request period*, or period for short, typically 1-6 seconds. Then, each node decides from which neighbor to ask for which blocks at the beginning of each request period. When a block does not arrive after its request is issued for a while and is still in the exchanging window, it should be requested in the following period again.

In following sections, we first intuitively explain what we optimize in data-driven streaming and then formulate this problem. Our basic approach is comprehensive. We define a priority for every desired block of each node due to the block importance, such as the block rarity, the block emergency, and block layer if layered video coding is used. Our goal is to maximize the average priority sum of all streaming blocks that are delivered to each node in one request period under heterogeneous bandwidth constraints.

2.1 Block Scheduling Problem

In this section, we introduce the block scheduling problem in data-driven P2P streaming. Figs. 1 and 2 give intuitive examples of *block scheduling problem*, BSP for short. The two numbers beside the pipe of each node represent the maximum blocks that can be downloaded and uploaded in each request period, respectively, denoting the inbound and outbound bandwidth constraints of each node. The blocks close to each node illustrate what blocks the node currently holds. We compare the LRF scheduling strategy used in [6] and optimal scheduling in these examples. In Fig. 1a, node 4 asks for blocks from node 1, 2, and 3. In the LRF scheduling strategy used in [6], a block that has the minimum number of holders among the neighbors is requested first. If multiple neighbors hold this block, it is assigned to the one with the maximum surplus bandwidth in turn. As illustrated in Fig. 1a, using LRF, block 3 has only one holder, so it is assigned to node 3. Then, the surplus upload bandwidth of node 3 is reduced to 1. After that, block 2 is assigned to node 2 since the surplus bandwidth of node 2 is larger than node 1 and the surplus bandwidth of node 2 becomes 1. Next, LRF strategy assigns block 4 to node 1 whose surplus bandwidth then descends to 0. Finally, after node 2 gets block 5, no more blocks can be further assigned. Fig. 1a shows the scheduling result using LRF. Four blocks are delivered. On the other hand, one optimal scheduling solution is illustrated in Fig. 1b, and five blocks can be delivered with a gain of 25 percent compared to LRF. In fact, using LRF method usually cannot derive the maximum throughput. As shown in Fig. 1a, the upload bandwidth at node 3 and download bandwidth at node 4 are not fully utilized. Fig. 2a shows another scenario that node 3 and 4 may competitively request blocks from node 1, that is, their requests are congested at node 1. However, node 4 has more options. The optimal way is that node 4 requests blocks from node 2, while node 3 requests from node 1, as shown in Fig. 2b. In fact, compared to LRF, random strategies used in [4] would be even worse. Moreover, the real situation is more complex because the bandwidth bottlenecks are not only at the last mile but also different blocks have different importance. As a consequence, more intelligent scheduling algorithms should be

TABLE 1
Notations

Notation	Description
N	Set of all nodes in the overlay except the source node 0, that is, $N = \{1, \cdots, \|N\|\}$ node
r	The streaming rate
$I_i, i = 0, \cdots, \|N\|$	The inbound bandwidth capacity of node i
$O_i, i = 0, \cdots, \|N\|$	The outbound bandwidth capacity of node i
$E_{ik}, i = 0, \cdots, \|N\|$	The maximal end-to-end available bandwidth between node i and k
$h_{ij} \in \{0, 1\}$	"$h_{ij} = 1$" denotes node i holds block j; "$h_{ij} = 0$", otherwise
NBR_i	Set of neighbors of node i
τ	The request period
W_T	The exchanging window size measured in seconds
C_i	The current clock time at node i
d_j^i	Play out deadline of block j at node i
L	Number of encoded layers
$r_l, l = 1, \cdots, L$	The cumulative rate from layer 1 to layer l, blocks per second
$\Lambda(r)$	$\Lambda(r) = max\{l : r_l \le r, i \in \{1, 2, \cdots, L\}\}$, the highest layer that can be achieved with rate r
l_j	The layer of block j
D_i^s, D_i^m	Set of all desired blocks in the current exchanging window of node i for single rate and multi-rate scenario respectively. $D_i^s = \{j : h_{ij} = 0, C_i < d_j^i < C_i + W_T\}$, $D_i^m = \{j : h_{ij} = 0, C_i < d_j^i < C_i + W_T, l_j \le \Lambda(I_i)\}$
$P_{ij}^s, P_{i,j}^m$	The block priority for single rate and multi-rate scenario respectively

developed to improve the throughput of data-driven protocol under bandwidth constraints.

2.2 Model

To maximize the throughput of the system, our approach is to maximize the number of blocks that are requested successfully under bandwidth constraints as much as possible within every period.

First, we define some notations that are used in the formulation. We let N denote the set of all receiver nodes in the overlay. Let I_i, $O_i (i = 0, \ldots, |N|)$ represent the inbound and outbound bandwidth of node i, and let $E_{ik} (i, k = 0, \ldots, |N|)$ represent the maximal end-to-end available bandwidth from node i to k. Since it is assumed that all the blocks have equal length, we let I_i, O_i, and E_{ik} be measured in blocks per second for convenience. Meanwhile, we use D_i to denote the set of all the desired blocks of node i in its current exchanging window. Let $h_{ij} \in \{0, 1\}$ denote whether node i holds block j. We assume that the size of the exchanging window is W_T measured in seconds. We let C_i and d_j^i denote the current clock on node i and the playback time, i.e., the deadline of block j on node i, respectively. Any desired block j of node i should satisfy $C_i < d_j^i < C_i + W_T$, that is, in exchanging window. Table 1 summarizes the notations in the rest of this paper. In data-driven protocol, different blocks have different significance. For instance, the blocks that have fewer suppliers should be requested preemptively so that they can be spread more quickly. Accordingly, defining different priority for blocks is important. Two properties have been considered in our priority definition: since the previous empirical study has shown that "rarest first" is a very efficient strategy in data dissemination [9], [13], [14], the rarity property is considered first. While as streaming application has real-time constraint, the second one we considered is the emergency property. A block in danger of being delayed beyond the deadline should be more preemptive than the one just entering the exchanging window. Consequently, we define the priority P_{ij}^s of block $j \in D_i$ for node $i \in N$ for the single rate scenario. The priority for the multirate scenario is defined in Section 4):

$$P_{ij}^s = \alpha P_R\Big(\sum_{k \in NBR(i)} h_k^j\Big) + (1 - \alpha) P_E\Big(d_j^i - C_i\Big). \quad (1)$$

Here, α satisfies $0 \le \alpha \le 1$. The function $P_R(*)$ in the first item represents the rarity property, and $\sum_{k \in NBR_i} h_k^j$ is the number of node i's neighbors that hold block j. The function $P_E(*)$ in the second item denotes the emergency property, and $d_j^i - C_i$ is the remaining time of block j until the playback deadline. Both $P_R(*)$ and $P_E(*)$ should be monotonously nonincreasing. For priority parameter, we set $P_R(R) = 10^{8-R}$, when $R = 1, \ldots, 8$, otherwise, $P_R(R) = 1$. And, we hope that each request period in exchanging window has different priority in terms of their remaining time until playback deadline, so we define $P_E(T) = 10^{8 - \lfloor T * \tau / W_T \rfloor}$, when $\lfloor T * \tau / W_T \rfloor \le 8$, otherwise, $P_E(T) = 1$. We give the proper value of α by a simple simulation approach in Section 5.

Then, we define the *decision variable* x_{kj}^i to denote whether node $i \in N$ requests block $j \in D_i$ from its neighbor $k \in NBR_i$:

$$x_{kj}^i = \begin{cases} 1, & \text{node } i \text{ requests block } j \text{ from neighbor } k, \\ 0, & \text{otherwise.} \end{cases} \quad (2)$$

Our target is to maximize the average priority sum of each node, so we have the following optimization:

$$\begin{aligned} &\max \frac{1}{|N|} \sum_{i \in N} \sum_{j \in D_i} \sum_{k \in NBR_i} P_{ij} h_{kj} x_{kj}^i, \\ &\text{s.t.} \\ &\text{(a)} \sum_{k \in NBR_i} x_{kj}^i \le 1, \forall i \in N, j \in D_i, \\ &\text{(b)} \sum_{j \in D_i} \sum_{k \in NBR_i} x_{kj}^i \le \tau I_i, \forall i \in N, \\ &\text{(c)} \sum_{i \in NBR_k} \sum_{j \in D_i} x_{kj}^i \le \tau O_k, \forall k \in N \cup \{0\}, \\ &\text{(d)} \sum_{j \in D_i} x_{kj}^i \le \tau E_{ki}, \forall i \in N, k \in NBR_i, \\ &\text{(e)} \ x_{kj}^i \in \{0, 1\}, \forall i \in N, k \in NBR_i, j \in D_i. \end{aligned} \quad (3)$$

In (3), $D_i = D_i^s$, for all $i \in N$ and $P_{ij} = P_{ij}^s$, for all $i \in N$, $j \in D_i$. Constraint (a) guarantees that each block should be fetched from at most one neighbor so that no duplicate blocks are requested. Constraints (b) and (c) ensure the block scheduling satisfy the inbound and outbound bandwidth capacity limitations, respectively. Constraint (d) ensures the maximal end-to-end available bandwidth limitation. Last, constraint (e) indicates this optimization would be a 0-1 programming problem. We call formulation (3) a global block scheduling problem (global BSP for short).

TABLE 2
Transformation Rules

a)	Put two virtual vertices in set V: s (the ***source vertex***) and t (the ***sink vertex***);
b)	$\forall\, i \in N$, insert a vertex r_i to V, called ***receiver vertex***;
c)	$\forall\, i \in N \cup \{0\}$, insert a vertex s_i to V, called ***sender vertex***;
d)	If node $k \in NBR_i$, insert a vertex n_{ki} to V, called ***neighbor vertex***;
e)	If block $j \in D_i$ (the desired block j is in the current exchanging windows of node i), then insert a vertex b_{ij} to V, called ***block vertex***;
f)	Arcs between source vertex and sender vertices (outbound bandwidth capacity constraints): insert an arc (s, s_k) to A, where $k \in N \cup \{0\}$. The capacity of this arc is τO_k (that is the maximal blocks which can be sent out from node k in one period), and the unit cost is 0;
g)	Arcs between receiver vertices and sink vertex (inbound bandwidth capacity constraints): insert an arc (r_i, t) to A, where $i \in N$. The capacity of the arc is τI_i, and the unit cost is 0;
h)	Arcs between sender vertices and neighbor vertices (end-to-end available bandwidth constraints): if $s_k \in V$ and $n_{ki} \in V$, insert an arc (s_k, n_{ki}) to A, where the capacity is $\tau E_k i$ (the maximal blocks that can be transmitted through path from node k to i in one period);
i)	Arcs between neighbor vertices and block vertices (representing blocks availability): if $k \in NBR_i$, $h_{kj} = 1$, $n_{ki} \in V$ and $b_{ij} \in V$, that is, node k holds block j and node i has not received block j, then insert an arc (n_{ki}, b_{ij}) to A, where the unit cost is 0 and the capacity is 1;
j)	Arcs between block vertices and receiver vertices (for blocks priority and duplicate avoidance): if $b_{ij} \in V$, then insert an arc (b_{ij}, r_i) to A, where the unit cost is $-P_{ij}/\lvert N\rvert$ and the capacity is 1.
k)	Insert an assistant arc (t, s) to A, where the unit cost is 0 and the capacity is $+\infty$.

2.3 Solution

In this section, we show that the global BSP can be transformed into an equivalent min-cost flow problem that can be solved within polynomial time. The min-cost flow problem is briefly depicted as follows [15]: Let $G = (V, A)$ be a directed network defined by a set V of n vertices and a set A of m directed arcs. Each arc $(i, j) \in A$ has an associated cost c_{ij}, denoting the cost per unit flow over that arc. A lower and an upper bound of capacity $l(i, j)$ and $u(i, j)$ is associated to the arc (i, j) to denote the minimum and maximum amount that can flow through the arc. Let $f(i, j)$ denote the flow amount on arc (i, j), which is the decision variables. The min-cost flow problem is an optimization model formulated as follows:

$$\begin{aligned}
&\min \textstyle\sum_{(i,j)\in A} c(i,j) f(i,j),\\
&\text{s.t.}\\
&\text{(a)}\ \textstyle\sum_{j:(i,j)\in A} f(i,j) - \sum_{j:(j,i)\in A} f(i,j) = b(i), \forall i \in V, \quad (4)\\
&\text{(b)}\ l(i,j) \le f(i,j) \le u(i,j), \forall (i,j) \in A,\\
&\text{where}\ \textstyle\sum_{i=1}^{n} b(i) = 0 \ \text{and}\ f(i,j) \in Z^{+}.
\end{aligned}$$

Such min-cost flow problem can be solved in polynomial time, and by double scaling algorithm [15], the time complexity for min-cost flow problem is bounded by $O(nm(\log\log U)\log(nC))$, where U and C are the largest magnitude of arc capacity and cost, respectively. In our model, we let $b(i) = 0$, for all $i \in V$ and $l(i, j) = 0$, for all arcs. We can start the transformation with the rules in Table 2, where rules (a) $\sim$ (e) and (f) $\sim$ (k), respectively, give the vertex and arc meanings.

Applying these rules, we can transform global BSP (3) into a corresponding min-cost flow problem, and we have the following theorem.

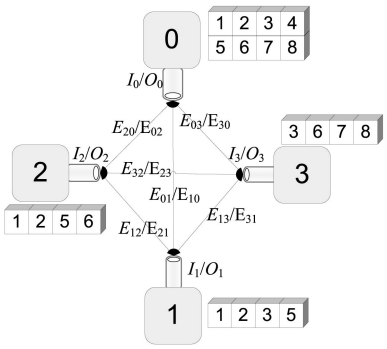


Fig. 3. A global block scheduling problem.

Theorem 1. *The optimal flow amount $f(n_{ki}, b_{ij})^*$ of the corresponding min-cost flow problem on arcs (n_{ki}, b_{ij}) ($\forall i \in N$, $j \in D_i$, and $k \in NBR_i$) is also an optimal solution to the global BSP.*

The proof of the theorem is in the Appendix. Figs. 3 and 4 show a sample of global BSP with four nodes and its min-cost flow modeling, respectively. In Fig. 4, the two numbers close to an arc represent the capacity and per unit flow cost of the arc. Rather than describe the general model formally, we merely describe the model ingredients for these figures. In data-driven streaming, we decompose a node into its three roles: a send, a receiver, and a neighbor. We model each sender k as a vertex s_k, each receiver i as a vertex r_i, and each neighbor k of node i as a vertex n_{ik}. Further, we model a desired block j for node i as a vertex b_{ij}. Besides, we add two virtual vertices: a source vertex s and a sink vertex t. The decision variables for this problem are whether to request block j from neighbor k of node i, which we represent by an arc from vertex n_{ik} to vertex b_{ij} if block j is a desired by node i. These arcs are capacitated by 1, and their per unit flow cost is 0. And, we insert arc from vertex n_{ik} to b_{ij} to indicate that neighbor k of node i holds block j. To avoid duplicate blocks, we add arc capacitated by 1 from b_{ij} to r_i and set the per unit flow cost as the priority of block j for node i multiplied a constant $-1/|N|$. To satisfy the outbound bandwidth constraint of node k, we add arc between vertex s and vertex s_k whose capacity is τO_k. And,

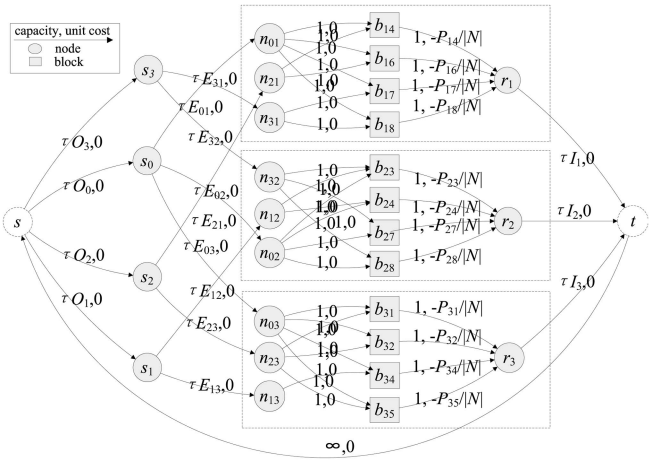


Fig. 4. Model as a min cost flow problem.

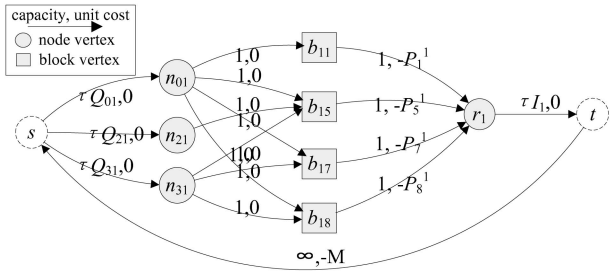


Fig. 5. An example for min-cost flow problem of the corresponding local BSP.

for the maximum end-to-end available bandwidth from neighbor k to node i, we insert arc from vertex s_k to n_{ik} with capacity τE_{ik}. Finally, to incorporate the inbound bandwidth constraint of node i, we introduce arc between r_i and t with capacity τI_i. To guarantee maximum number of blocks are delivered, we insert uncapacitated arc from vertex t to s.

3 Heuristic Distributed Algorithm

We give the algorithm to do global optimal scheduling in Section 2. However, requiring global knowledge, such as block availability, bandwidth information, and request synchronization make the algorithm not scalable. In this section, based on the basic idea of the global optimal solution, we present the heuristic practical algorithm, which is fully distributed and asynchronous. In our distributed algorithm, each node decides from which neighbor to fetch which blocks at the beginning of its request period. As the request period is relatively short, such as 3 seconds, our scheduling algorithm should make a decision as rapidly as possible. Therefore, in the distributed algorithm, we do a local optimal block scheduling on each node based on the current knowledge of the block availability among the neighbors. The local optimal block scheduling can also be modeled as a min-cost flow problem. As shown in Fig. 4, the sub-min-cost flow problem in the each rectangle is just the local optimal block scheduling.

However, one problem to do local scheduling is that each node does not know the optimal flow amount on arcs (s_k, n_{ki}) $(\leq O_k)$. In other words, we should estimate the proper upper bound of the bandwidth from each neighbor. For simplicity, here, we use a purely heuristic way for each node to estimate the maximum rate at which each neighbor can send blocks. Our approach is to use the historical traffic from each neighbor to do this. More formally, let Q_{ki} denote the estimated maximum rate at which neighbor $k \in NBR_i$ can deliver to node i. Of course, Q_{ki} should not exceed O_k. We let $g_{ki}^{(p)}$ denote the total number of blocks received by node i from neighbor k in the pth period. In each request period, we use the average traffic received by node i in the previous P periods to estimate Q_{ki} in the $(p+1)$th period:

$$Q_{ki} = \gamma \cdot \left(\sum_{\omega=p-P+1}^{p} g_{ki}^{(\omega)} \right) / P\tau. \tag{5}$$

Parameter $\gamma(>1)$ is a constant called aggressive coefficient. A small number of γ may lead to a waste of bandwidth. On the other hand, augmenting this parameter tends to a sufficient utilization of the outbound bandwidth of the neighbor nodes. However, more request congestion may happen at some nodes, that is, the block requests from neighbors may be beyond outbound bandwidth. The impact of this parameter will be investigated in Section 5. We simply set the initial value of estimated rate from neighbor i as $r/|NBR_i|$:

TABLE 3
Heuristic Distributed Algorithm

a)	Node i estimates the bandwidth $Q_{ki}^{(m+1)}$ that its neighbor k can allocate it in the $(m+1)^{th}$ period with the traffic received from that neighbor in the previous P periods, as shown in (5);
b)	Based on $Q_{ki}^{(m+1)}$, node i performs the block scheduling (6) using min-cost network flow model. The results $x_{kj}^i \in \{0,1\}$ represent whether node i should request block j from neighbor k;
c)	Send requests to every neighbor.

$$\begin{aligned} &\max \sum\nolimits_{j\in D_i} \sum\nolimits_{k\in NBR_i} P_{ij} h_{kj} x_{kj}^i \\ &\text{s.t.} \\ &1.\ \sum_{k\in NBR_i} x_{kj}^i \leq 1, \forall j \in D_i, \\ &2.\ \sum_{j\in D_i} \sum_{k\in NBR_i} x_{kj}^i \leq \tau I_i, \forall i \in N \\ &3.\ \sum_{j\in D_i} x_{kj}^i \leq \tau Q_{ki}, \forall i \in N, k \in NBR_i, \\ &4.\ x_{kj}^i \in \{0,1\}, \forall k \in NBR_i, j \in D_i. \end{aligned} \tag{6}$$

Similar to global BSP, the local BSP (6) can also be transformed to an equivalent min-cost flow problem by inserting a virtual source node and a virtual sink node with an assistant arc between them to the sub-min-cost flow problem in global BSP. Since it is much easier than the global BSP, we only give an example here. Fig. 5 shows the corresponding min-cost flow problem of the local BSP on node 1 for the topology illustrated in Fig. 3. The flow amount on arcs $(n_{ki}, b_{ij}) \in \{0,1\}$ is the value of x_{kj}^i, for all $i \in N$, and $k \in NBR_i$.

We summarize our distributed algorithm. In the beginning of each request period, do the steps in Table 3. Our distributed algorithm is heuristic, and we will examine its performance and the gap between the distributed algorithm and the global optimal solution by simulation in Section 5.

4 Extending for Multiple Streaming Rates

A lot of measurement studies in peer-to-peer overlay networks [16] reveal that the bottleneck bandwidth between the end hosts exhibits extremely heterogeneity. To deal with the users heterogeneity in streaming multicast applications, numerous solutions has been proposed for both IP multicast [17] and overlay multicast [18], [19]. Their basic way is to encode the source video into multiple layers using layered video coding, and each receiver subscribes an appropriate number of layers due to its bandwidth capacity. In this section, we will show that by simply modifying the block priority definition, data-driven protocol can be extended to combine with layered coding to tackle the heterogeneity

issue. Similar to single rate scenario, we divide each layer encoded into blocks. It should be noted that, in layered video coding, video is encoded into a base layer and several enhanced layers, and a higher layer can only be decoded if all lower layers are available and we call this the layer dependency. Therefore, each block has an additional important property, i.e., the layer property.

We define some additional notations, all of which are summarized in Table 1. We let L represent the number of layers that the video is encoded into and let r_l (where $l=1,\ldots,L$) denote the cumulative rate from layer 1 to layer l. And, we define $\Lambda(r) = max\{l : r_l \leq r, i \in \{1,2,\ldots,L\}\}$, so $\Lambda(I_i)$ represents the maximum layer that node i with inbound bandwidth I_i can achieve. We assume that each node only try to request blocks whose layers are equal to or lower than $\Lambda(I_i)$. Let l_j represent the layer of block j. Therefore, we definition the set of node i's desired blocks in current exchanging window for multirate scenario as $D_i^m = \{j : h_{ij} = 0, C_i < d_j^i < C_i + W_T, l_j \leq \Lambda(I_i)\}$. Then, we define block j's priority value P_{ij}^m of node i for the multirate scenario as follows:

$$P_{ij}^m = \beta P_{ij}^s + (1-\beta)\theta P_L(l_j), \quad \text{where } \beta = \left(d_j^i - C_i\right)/W_T. \tag{7}$$

P_{ij}^s is defined as in (1). Function P_L represents the layer property of block j and satisfies $\Pi_L(l_{j_1}) \gg \Pi_L(l_{j_2})$ when $l_{j_1} < l_{j_2}$, for any block j_1 and j_2 so as to guarantee the layer dependency requirement. Parameter $0 \leq \beta \leq 1$ represents the current position block j in the exchanging window. We let θ have relatively large value, that is, $\theta \gg 1$. In this priority definition, when a block just entered the exchanging window (β is large), the first item has more weight in order to provide more block diversity to the system; meanwhile, when a block is to be played back soon (β is small), the second item contributes more in the priority to ensure the blocks at lower layer are requested preemptively. Although this block priority definition is just a simple linear combination of the two properties, it can guarantee the following key requirements: 1) A lower layer block, especially when it is to be played back soon, has much higher priority than any other upper layer blocks since θ is large. 2) A block with fewer holders has a higher priority than the one with more holders in the same layer and the same position in exchanging window.

5 Performance Evaluation

5.1 Simulation Configuration

As aforementioned, there are two key steps, i.e., the overlay construction and the block scheduling, in data-driven peer-to-peer streaming, and we focus on the block scheduling step. For a fair comparison, all the experiments use the same simple algorithm for overlay construction: each node independently selects its neighbors randomly so that a random graph is organized. And, our simulation ensures that each node has the same set of neighbors at any simulation time for every method. Moreover, to evaluate the performance, we define a metric—*delivery ratio* formally here. *The delivery ratio of a node is represented by the number of blocks that arrive at the node before playback deadline over the total number of blocks encoded in the stream*. Since the total number of blocks in the stream is constant that only relies on the encoding and packetization, the delivery ratio of a node can represent the throughput from the source to this node. The delivery ratio of the whole session is measured as the average delivery ratio among all nodes, also representing the throughput of the session. For the underlying topology, we use the random model of GT-ITM [20] to generate a topology with 2,000 routers and set delays proportional to the distance metric of the resulting topology within [5 ms, 300 ms].

In our experiment, we implement a discrete event-driven peer-to-peer simulator[1] and use Goldberg's "CS2" library [21] to solve min-cost network flow problem. As suggested in [5] and [22], the request period should be several seconds. In all the experiments, we hence fix the request period to 3 seconds. And, the default group size of the whole session is 1,000 nodes. Previous study [22] has shown that there is a sweet range of neighbor count or peer degree roughly between 6 to 14 in data-driven/swarming-based streaming where the delivered quality to the majority of peers is high, and actually, it is the *overlay construction* issue. Therefore, in our simulation, each node randomly selects 14 other nodes as its neighbors. Each block has the same size of 1,250 bytes, i.e., 10 Kbits. Each node estimates the bandwidth allocated from a neighbor with the traffic received from it in the past five periods, namely, $M = 5$. Moreover, the default aggressive coefficient used is $\gamma = 1.5$. We set the default exchanging window size to 10 seconds and the sliding window to 1 minute.

Our experiments are driven by real-world traces obtained from the real-deployed peer-to-peer streaming system—GridMedia [1], [23]. This system has been online to broadcast programs for CCTV[2] [24] since January 2005. In 28 January 2006, GridMedia system supported over 220,000 users simultaneously online from about 70 countries to watch the CCTV Gala Evening for Spring Festival at a streaming rate of 300 Kbps only by one server and about 200 megabits per second (Mbps) server bandwidth is consumed. This is one of the records of peer-to-peer streaming system until 2006. The traces on that day mainly include the arrival and leave time of different nodes. The cumulative distribution of the user online time is shown in Fig. 6. Hence, we can employ the realistic arrival/leave patterns in the traces to simulate the churn of the participating nodes. In our experiment, each run of the simulation is driven by the same part of traces on that night, i.e., the 30-minute traces. As our default group size is set to 1,000, new node joining request is refused in our simulation, when the total online nodes exceed 1,000. Besides, in all of our experiments, we set the outbound bandwidth of the source node to 2 Mbps.

For user outbound/upstream and inbound/downstream bandwidth, as [14], we adopt the measurement results derived from actual Gnutella nodes in [16]. As in [14], we discretize this CDF (i.e., the cumulative distributed function) into *clusters*, and the bandwidth of each cluster follows a Gaussian distribution. The mean bandwidth of each cluster is

1. The simulator is an open source project and its basic component is available online for free downloading: http://media.cs.tsinghua.edu.cn/~zhangm.

2. CCTV—China Central TeleVision is the largest TV station in China. CCTV Online TV: http://www.cctv.com/p2p/index.htm.

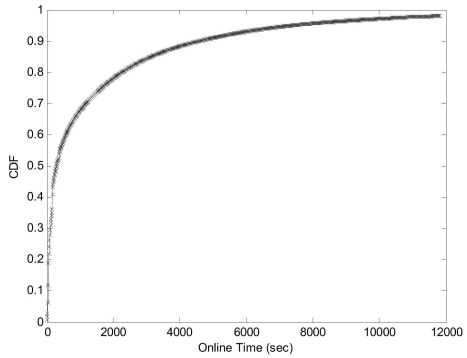


Fig. 6. CDF of user online time obtained from a real-world peer-to-peer streaming system [1].

shown in Table 4, and the standard deviation of a cluster is 10 percent of its mean. Since in our single rate scenario, the streaming rate we used is up to 500 Kbps, and the nodes with inbound bandwidth lower than that rate are unlikely to participate such a high-rate session, we exclude those nodes. In this section, we assume that the first three clusters are DSL/Cable users and the fourth cluster is Ethernet users. The fraction of each DSL/Cable cluster is shown in Table 4, and the ethernet users are altered in different simulation and at most 15 percent among all the users.

5.2 Performance Comparison for Single Rate

Then, we show the performance of our proposed algorithms, including the global optimal and distributed algorithm, and give the comparison with the following conventional ad hoc methods in block scheduling:

- *Pure random method*. Each node will assign each desired block randomly to a neighbor that holds that block. Pai et al. [4] uses this simple strategy.
- *LRF method*. As Section 2 depicted, a block that has the minimum owners among the neighbors will be requested first. DONet [6] employs this strategy.
- *Round-robin (RR) method*. All the desired packets will be assigned to one neighbor in a prescribed order in a round-robin way. If the block is only available at one sender, it is assigned to that sender. Otherwise, it is assigned to a sender that has the maximum surplus available bandwidth. When a block is assigned to a neighbor, the surplus bandwidth of that neighbor will be recalculated by subtracting the amount the block consumes. These steps are repeated till there is no surplus bandwidth or no blocks can be assigned.

Besides, we also give the comparison with Narada [25] to provide the throughput of the traditional single tree-based protocol. The simulator we use for Narada is an myns simulator downloaded from that in [26], which implements the Narada protocol.

TABLE 4
Bandwidth Distribution Used in Simulation

Type	Inbound (kbps)	Outbound (kbps)	Fraction
DSL/Cable	784	128	0.2 among DSL/Cable nodes
DSL/Cable	1500	384	0.5 among DSL/Cable nodes
DSL/Cable	3000	1000	0.3 among DSL/Cable nodes
Ethernet	10000	5000	altered (0∼0.15 among all)

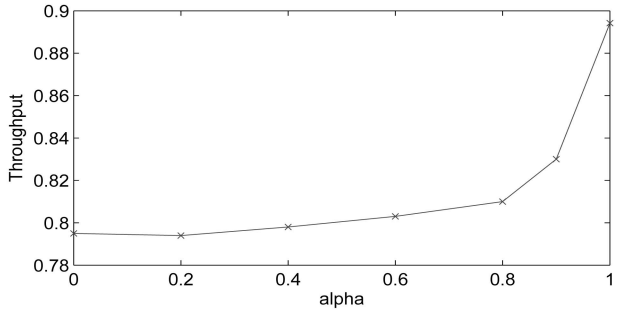


Fig. 7. Deliver ratio for different value of α.

We first use a simulation-based approach to show that the proper value of α in priority definition (1) should be 1. We set all nodes to DSL/Cable nodes and set the streaming rate to 500 Kbps. Fig. 7 shows the delivery ratio under different value of $\alpha \in [0, 1]$. It is somewhat surprising that the emergency property has much weaker effect on improving the throughput compared to rarity property. Some similar results are also obtained in recent literature [27]. An intuitive explanation is that requesting more emergent blocks will incur more extra bandwidth consumed for future blocks. Hence, it has little help to the throughput enhancement. However, since the throughput cannot completely reflect the final playback quality, we will investigate the impact of emergency property on the playback quality for future work. In this section, we set $\alpha = 1$ in all of our experiments.

In Fig. 8, we study the performance of each method when all the nodes are DSL/Cable nodes. This scenario is frequent. For example, in the online classes of some distant education institute in China, such as CRTVU [28], most of the students access Internet through DSL from their home. Therefore, in this figure, we assume that bandwidths of the users are distributed as those of the DSL/Cable nodes in Table 4, and the bottlenecks are only at the last mile. Moreover, the exchanging window size and the sliding window is set to 10 seconds and 1 minute, respectively. We see that when the streaming rate is 250 Kbps, all the

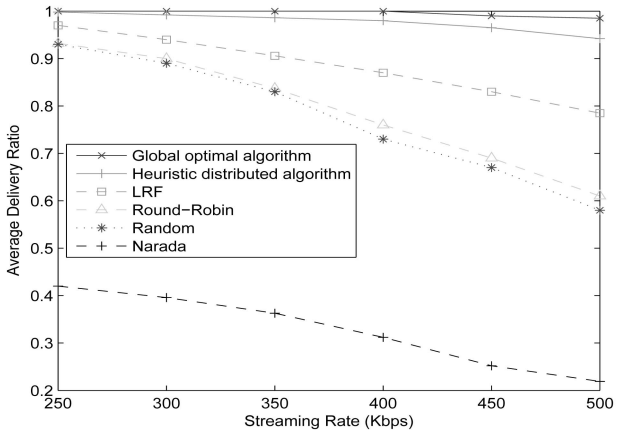


Fig. 8. Delivery ratio for different value of streaming rate. All are DSL/Cable nodes with 10-second exchanging window and 1-minute sliding window, and bottlenecks are only at the last mile. Group size is 1,000.

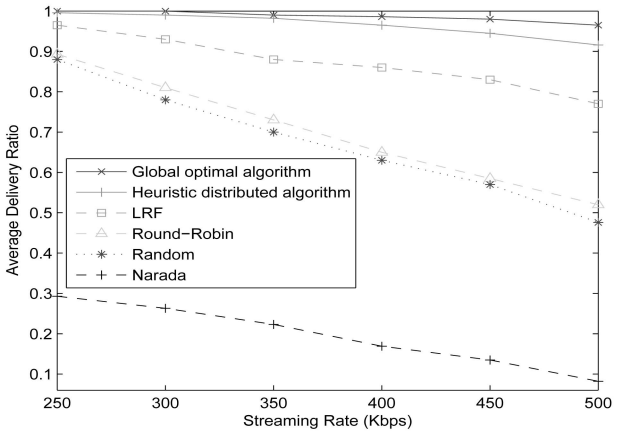


Fig. 9. Delivery ratio for different value of streaming rate. All are DSL/Cable users with 10-second exchanging window and 1-minute sliding window and maximal end-to-end available bandwidth 0-150 Kbps. Group size is 1,000.

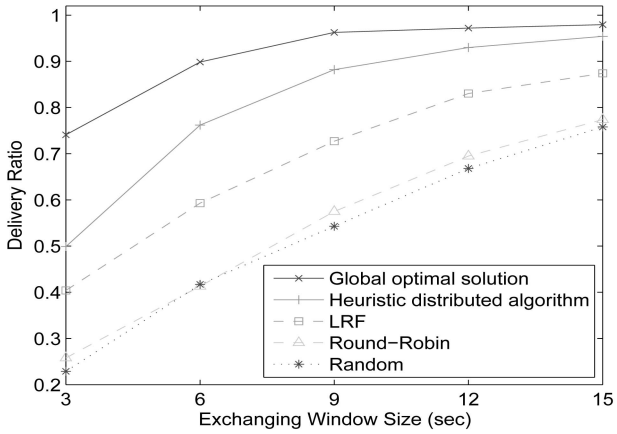


Fig. 10. Delivery ratio for different exchanging window size. All are DSL/Cable nodes with 1-minute sliding window. Streaming rate is 500 Kbps and group size is 1,000.

methods except Narada have very high delivery ratio usually above 90 percent. As the streaming rate increases, the delivery ratio the global optimal solution keeps around 100 percent. Even when the streaming rate reaches 500 Kbps, its delivery ratio still remains above 97 percent. This reveals that the network capacity is sufficient to support the multicast session with 250-500 Kbps. We note that the performance of the three compared methods (LRF, Round-Robin, and Random) goes down fast with the increase of the streaming rate; however, the delivery ratio of our proposed heuristic distributed algorithm is fairly good. At the rate of 500 Kbps, the distributed algorithm outperforms the LRF, Round-Robin, and Random methods by gains of 21 percent, 52 percent, and 62 percent, respectively. The gap between the global optimal solution and the heuristic distributed algorithm is 9 percent. We can also see that the delivery ratio of Narada protocol is low because the traditional single tree-based protocol cannot effectively utilize the outbound bandwidth of all the peers.

Then, we investigate the influence of end-to-end bandwidth constraints. Based on the settings in Fig. 8, we further add the constraint that the maximum end-to-end available bandwidth is between 0 and 150 Kbps in Fig. 9. It is interesting that the end-to-end bandwidth constraints do not have much influence on the delivery ratio among the data-driven protocols. As shown in the figure, the performance of all the methods does not drops much compared to the results in Fig. 8. This is because the data-driven protocol has the inherent ability to allocate the traffic to each neighbor and make full utilize of all bandwidths from every neighbor. Hence, t is very robust to the bandwidth heterogeneity between nodes. The gap between the global optimal solution and the heuristic distributed algorithm is 12 percent. At the rate of 500 Kbps, compared to LRF, Round-Robin, and Random methods, the gains in delivery ratio of the proposed distributed algorithm are 19 percent, 58 percent, and 72 percent, respectively. However, we see that the delivery ratio of the Narada protocol is much poorer compared to that in Fig. 8 since the maximum throughput of the single tree protocol cannot excess 150 Kbps.

The impact of the exchanging window size is shown in Fig. 10. We set the streaming rate to 500 Kbps, and all the other configurations are the same, as in Fig. 8. It is observed that increasing the exchanging window size can significantly improve the delivery ratio. This is because when the exchanging window size is larger, there are more chances to rerequest the blocks that are not arrived due to bandwidth limitation or packet loss. However, the users need to wait more time for the start-up and the watching delay also gets larger. Our proposed distributed algorithm and the global optimal solution can achieve a higher delivery ratio with a relatively smaller exchanging window. As shown, the delivery ratio with 9-second exchanging window of our proposed distributed algorithm reaches 88.2 percent, which is even higher than that of the LRF method with 15-second exchanging window.

Next, we study the impact of Ethernet user percentage. As illustrated in Fig. 11, we vary the percentage of Ethernet users from 0 to 15 percent whose bandwidths are shown in Table 4. And, we assume that the left nodes are all DSL/Cable ones, and their fractions of each bandwidth type are also as listed in Table 4. As shown in Fig. 11, when there are no Ethernet users, that is, all are DSL/Cable nodes, our proposed distributed algorithm has a significant improvement compared to LRF,

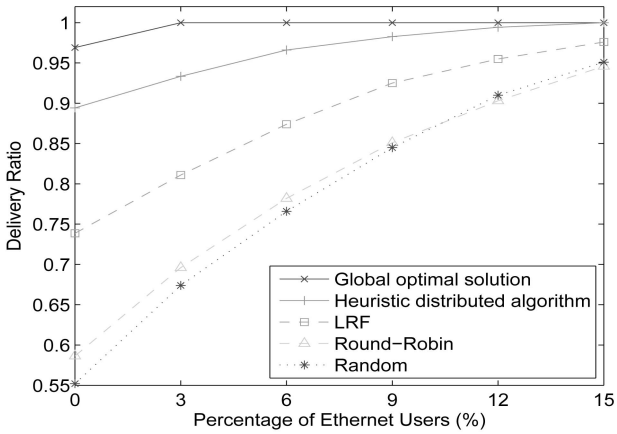


Fig. 11. Delivery ratio for different percentage of Ethernet users. Streaming rate is 500 Kbps, and group size is 1,000.

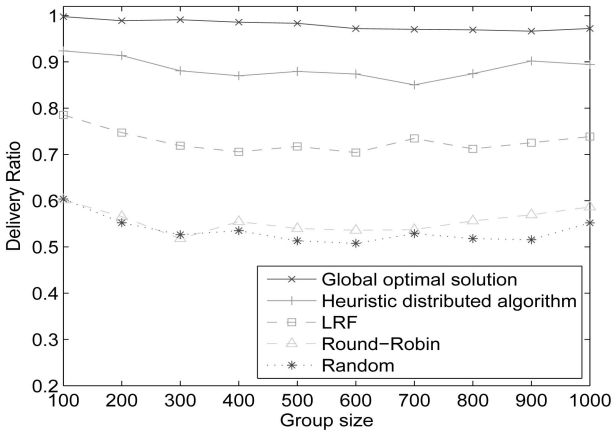


Fig. 12. Delivery ratio for different group size. All are DSL users with 10-second exchanging window and 1-minute sliding window. Streaming rate is 500 Kbps.

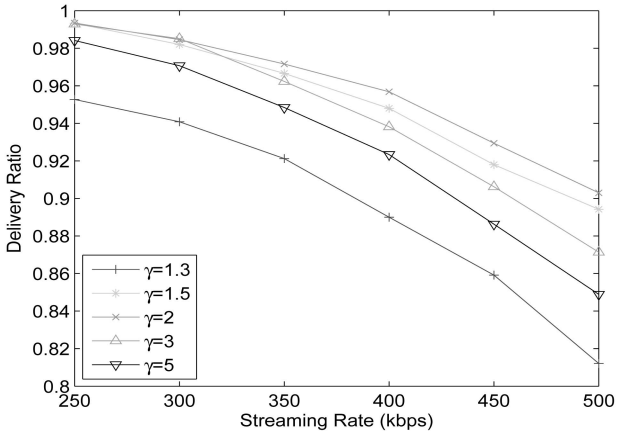


Fig. 13. Delivery ratio for different value of γ. All are DSL users with 10-second exchanging window and 1-minute sliding window. Streaming rate is 500 Kbps.

Round-Robin, and Random methods. This is because more intelligent scheduling in our proposed algorithm can utilize the bandwidth resources much more efficiently especially when the resources are very stringent. With the increasing of Ethernet users, the bandwidth resources get more abundant, and we note that the gap between the compared methods gets smaller. However, on the other hand, if the bandwidth resources were plentiful, we could promote the streaming rate and take advantage of the optimized scheduling of our proposed algorithm to support a service with better quality. We will show in the Section 5.3 that with the assist of layered coding, we can adapt the streaming rate for different types of nodes, and our proposed algorithm can achieve higher throughput under the same stringent limitation of bandwidth resources.

Fig. 12 shows the delivery ratio with respect to different group size. Here, we set the streaming rate at 500 Kbps, and all nodes are DSL/Cable ones. The bandwidth bottlenecks are only at the last mile. We can observe that all the curves are flat, and the performance of all the methods basically remains almost the same when the group sizes increase. This indicates that the group size has little impact on the performance of data-driven/swarming-based protocol, therefore, data-driven protocol has very good scalability. It is noted that our proposed distributed algorithm always has a gain of 20 percent-60 percent to the three conventional methods.

Then, we check the impact of the aggressive coefficient γ in our proposed distributed algorithm. We set the streaming rate to 500 Kbps, and all nodes are DSL/Cable ones. As shown in Fig. 13, we observe that there is a trade-off for the value of γ. When γ is small (below 1.5) or large (greater than 3), the delivery ratio is relatively poor. In fact, at the first request period, the estimated bandwidth from each neighbor is always set to a small number. From the second period, the estimated bandwidth will be γ times the coming traffic rate, and this results in that the estimated bandwidth converges from a small value to the real available bandwidth. The impact of this parameter is comprehensible. Small γ will decrease the convergent speed, resulting in the low utilization of the bandwidth. On the other hand, although large γ leads to fast convergent speed, however, it causes overmuch estimated bandwidth, and hence, a lot of requested packets may fail to be fetched. Besides, large γ can induce request congestion at some hot nodes, typically, the nodes close to the source. By simulation, we obtain a sweet range of γ, i.e., 1.5-3. Within this range, the average delivery ratio keeps high. In the future work, we would like to study how to adjust this value according to the network condition.

Fig. 14 shows the cumulative distribution function of the packet arrival delay of the four distributed scheduling methods. The packet arrival delay here is between the time when packet is sent out from the source node and the time when the packet finally arrives at a node after several hops. We can see that the proposed heuristic method has less delay than the other three methods. Since the heuristic method performs a local optimal block scheduling, it can successfully obtain its required blocks with much higher probability compared to other three naive methods, that is, a block can be successfully requested in fewer periods. Thus, our proposed heuristic method is superior to others in delay performance.

Finally, we conduct the comparison on PlanetLab. With the GridMedia [23] code, we implement all the four distributed packet scheduling methods. The main configuration for the experiment is the same as that in Fig. 8.

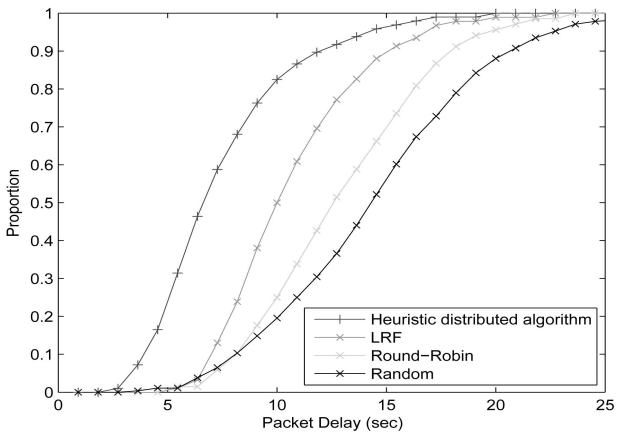


Fig. 14. The packet arrival delay CDF of the four methods.

TABLE 5
Special Nodes Used in PlanetLab Experiment

Special node	URL
P2P Tracking node	planetlab1.csail.mit.edu
Source node	planetlab1.cs.cornell.edu
Log collecting server	thu1.6planetlab.edu.cn
Command nodes[a]	thu1.6planetlab.edu.cn

[a] *Command nodes are used to control other nodes to join and leave.*

And, the streaming rate we use is 400 Kbps. We also use the GridMedia trace to drive our experiment. We uploaded our program to 200 nodes at 200 different sites on PlanetLab. Some special nodes in our experiment are listed in Table 5. The experiment lasts for half an hour. As shown in Fig. 15, the delivery ratio of the proposed heuristic algorithm is around 96 percent. The delivery ratios of LRF, Round-Robin, and Random method are about 87 percent, 75 percent, and 71 percent, respectively. Our algorithm outperforms the other methods in Internet environment.

5.3 Performance Comparison for Multiple Rates

In this section, we check the performance of each method when we encode the video into multiple rates using layered video coding. With the assistance of layered coding, the video rate can adapt to different bandwidth capacity, and all types of users can be supported in one session. As a consequence, in terms of the fractions shown in Table 4, we add an additional cluster of low-capacitated DSL/Cable users whose inbound and outbound bandwidth are 384 Kbps and 128 Kbps, respectively, with a fraction of 0.1. And, the fractions of the other DSL/Cable clusters among the rest DSL/Cable users are as shown in Table 4. We assume that the percentage of Ethernet users is 10 percent. The group size is set to 1,000. As aforementioned, the bandwidth of each cluster follows a Gaussian distribution, and the standard deviation of a cluster is 10 percent of its mean. For the priority in (7), we set θ as a large value 1,000 and define function $P_L(l) = 10^{2(L-l)}$ to ensure the lower layers have much larger priority than the upper layers. And, the compared methods include the following five ones:

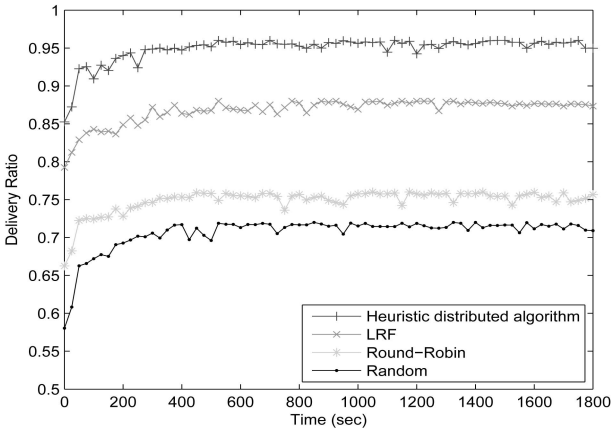


Fig. 15. PlanetLab experiment: the delivery ratio with respect to the elapsed time.

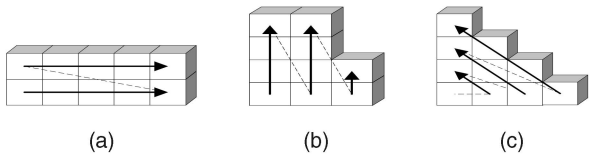


Fig. 16. Three round-robin strategies. (a) Conservative. (b) Aggressive. (c) Trade-off.

- *Random method and LRF method*. As explained in Section 5.2.
- *RR method*. The round robin method used here is basically as explained in Section 5.2. However, for the scenario of multirate with layered coding, the round-robin method is usually classified into three different ways with respect to the block ordering sequence s illustrated in Fig. 16. Fig. 16a shows the conservative block ordering: it always requests blocks of lower layers first. On the contrary, aggressive block ordering scheme requests blocks of all layers with lowest sequence number (or time stamp) preemptively, as illustrated in Fig. 16b. Fig. 16c uses a zigzag ordering, which is a trade-off between the two extreme schemes. Since the first two schemes evidently have its own limitations [5]. We only compare with the third scheme (we call it RR-trade-off for short).

As mentioned previously, each node i only requests the blocks whose layers are not beyond its capacity, i.e., any block $j \in D_i^m$ should satisfy $l_j \leq \Lambda(I_i)$. To evaluate the performance under multirate scenario, we define the delivery ratio at layer l of the whole session as the average delivery ratio at layer l among all the nodes that can achieve layer l. We first encode the video into 10 layers and set the rate of each layer as 100 Kbps. Hence, the steaming rate can be up to 1 Mbps. We assume that the bandwidth bottlenecks are only at the last mile here. Actually, in our configuration, the bandwidth resources are very stringent. Due to the fractions of the nodes with different bandwidth capacity, the total outbound bandwidth can be computed as $1{,}000 \times (0.1 \times 128 + (0.2 \times 128 + 0.5 \times 384 + 0.3 \times 1{,}000) * 0.8 + 0.1 \times 5{,}000) = 926{,}880$. And, the total bandwidth needed can be computed as the sum of the streaming rate needed for different types of nodes: $1{,}000 \times (0.1 \times 300 + (0.2 \times 700 + 0.5 \times 1{,}000 + 0.3 \times 1{,}000) \times 0.8 + 0.1 \times 1{,}000) = 882{,}000$. Note that they are very close to each other.

Fig. 17 gives the delivery ratio at each layer. We note that the global optimal solution has the best performance, and the delivery ratio in all layers is nearly 1. This demonstrates that the generated topologies have sufficient capacity to support all the nodes to receive all layers that they can achieve. The performance of distributed algorithm is fairly good. Most of the delivery ratio in lower layers has nearly 1 and most in higher layers is also above 0.9. For the RR-trade-off method, we use zigzag ordering with slope of 0.1. And, it can be seen that RR-trade-off has much more better delivery ratio at lower layers than higher layers. But, the delivery ratio at all layers is not so good as the proposed distributed algorithm. We note that the LRF strategy has even higher delivery ratio than the round-robin scheme. However, the curve is flat. This means it cannot tackle the layer dependency problem, that is, many

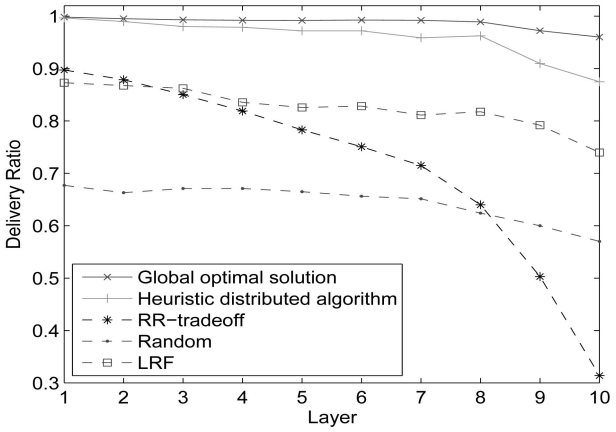


Fig. 17. The video is encoded into 10 layers. The group size is 1,000.

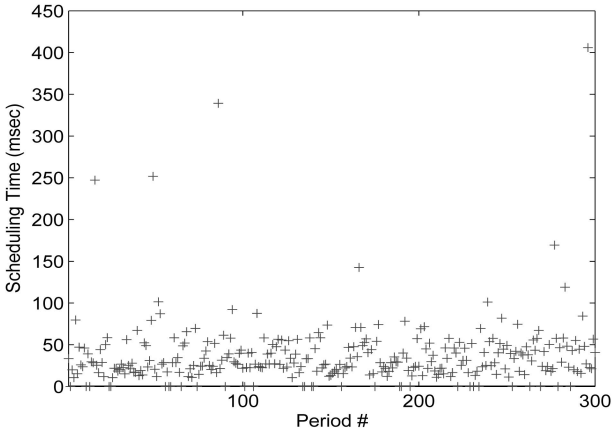


Fig. 19. Computing overhead. Streaming rate is 500 Kbps.

nodes cannot watch even the base layer, although more blocks of higher layers are propagated. Finally, the random strategy has the poorest performance. As shown in Fig. 17, our distributed method outperforms other strategies much with a gain of 10 percent-50 percent in most layers. In Fig. 18, the video is encoded into five layers, each of which is 200 Kbps. We observe that the performance of each layer is similar to that in Fig. 17. Our proposed distributed algorithm is still the best among all the distributed methods. We note that the delivery ratio of all the method is a little higher than that in Fig. 17. This is because, compared to 10 layers, encoding the video into five layers is more coarse, and thus, many nodes will request a lower rate of streaming since the larger mismatch between the streaming rate and the inbound bandwidth. Therefore, more residual available bandwidth can be used to support transmission at lower layers.

5.4 Overhead

Finally, we examine the overhead of our proposed distributed algorithm, including the computing and control overhead. Fig. 19 shows the computing overhead—the time consumed for block scheduling in every period on a typical node that schedules blocks among its 14 neighbors. Our experiment is done on a Pentium III 600 MHz machine. The streaming rate here is 500 Kbps, and the request period is 3 seconds. Since the block size is 1,250 bytes, there are 150 blocks needing scheduling in a period. In Fig. 19, we note that most of the scheduling can be finished in 80 ms and very few cases cost more than 100 ms. Since the request period is 3 seconds, this computing overhead is low and practically acceptable. Furthermore, for the control overhead, it mainly includes the neighbor maintenance packets, buffer map packets, and request packets. As shown in Fig. 20, we plot the control overhead with respect to the different neighbor numbers and different group size for our proposed method, as well as LRF and Random method. We see that our proposed heuristic algorithm has a little lower control overhead than the other methods. This is because the proposed algorithm can successfully obtain a packet in fewer periods compared to naive scheduling methods. Therefore, fewer request packets are needed to send. Besides, it can be observed that the control overhead has little relationship with the group size because each node only communicates with its own neighbors, which demonstrates the good scalability of our algorithm.

6 Related Work

Due to the difficulty of Internet-wide deployment of IP multicast, overlay multicast especially peer-to-peer based multicast is regarded as a very promising alternative to support live streaming applications. Traditional overlay multicast can be classified into three categories in terms of the tree building approaches [29]: tree-first, such as YOID

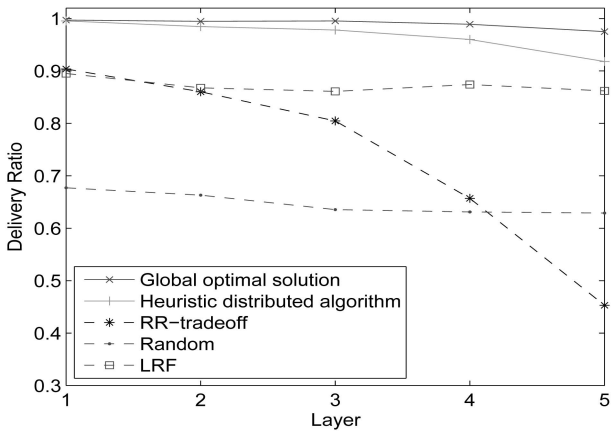


Fig. 18. The video is encoded into five layers. The group size is 1,000.

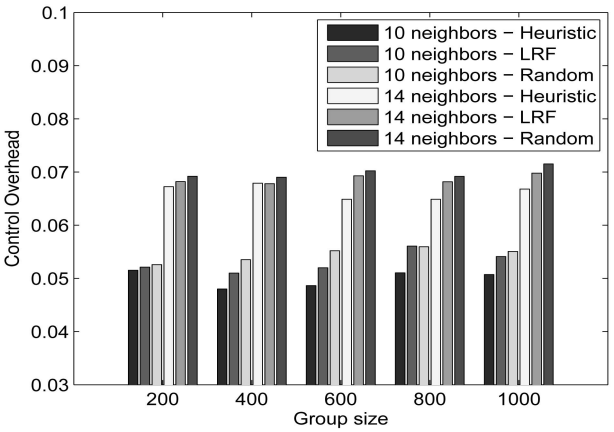


Fig. 20. Control overhead. Streaming rate is 500 Kbps.

[30]; mesh-first, such as Narada [25]; implicit protocol, such as SCRIBE [31]. To utilize the upload bandwidth of end hosts more efficiently, some protocols based on multiple trees are also proposed, such as SplitStream [32]. Rather than building trees, recently proposed data-driven/swarming-based protocols organize the nodes into an unstructured overlay network, i.e., the overlay construction. Meanwhile, each node independently fetches the absent data from its own neighbors according to the notification of data availability, i.e., the scheduling, very similar to the mechanism used in some file sharing systems, such as BitTorrent [9] and Bullet' [13]. These data-driven protocols include Chainsaw [4], DONet [6], PALS [5], and PRIME [8]. Most recent result [33] shows that this type of protocol outperforms the traditional multitree approaches much in many aspects. To improve the performance of data-driven protocol, much efforts are made on the overlay construction [11], [12], [10]. Venkataraman and Francis [11] discuss how to construct an efficient random graph under a heterogeneous environment, and Jiang and Nahrstedt [12] adds a QoS-awareness feature to unstructured overlay. Whereas, the scheduling issue has not been well addressed yet. Most of the proposed methods are simply strategy-based and ad hoc. Pai et al. [4] uses a pure random way to deal with the scheduling. DONet [6] employs an LRF strategy as the block scheduling method, i.e., greedy method, and selects suppliers with the highest bandwidth and enough available time first. PRIME [8] encodes the video into Multiple Description Coding (MDC) and employs joint diffusion and swarming techniques. Similar result shows that the scheduling using rarity factor is better than the random sender selection strategy. However, they do not compare with optimal scheduling method as proposed in this paper. Actually, the problem of doing optimal block scheduling in data-driven/swarming-based peer-to-peer streaming has not been adequately studied in the literature.

Besides, in layered peer-to-peer streaming, Cui and Nahrstedt [18] proposes a peer-to-peer streaming solution to address the on-demand media distribution problem, and it optimally allocates the desired layers among multiple senders. LION [19] employs a multipath-based method to improve the throughput of overlay network using network coding, which is a very different way from data-driven/swarming approaches. PALS [5] is an adaptive streaming mechanism from multiple senders to a single receiver using layered coding, which is actually a swarming based or data-driven protocol. However, PALS mainly focuses on the scenario of streaming from multiple senders to a single receiver to cope with the network dynamics such as bandwidth variations and sender participation. It does not aim to improve the throughput of data-driven streaming and does not evaluate its performance under an overlay mesh as well.

7 Conclusion and Future Work

In this paper, we study the scheduling problem in the data-driven/swarming based protocol in peer-to-peer streaming. The contributions of this paper are twofold. First, to the best of our knowledge, we are the first to theoretically address the scheduling problem in data-driven protocol. Second, we give the optimal scheduling algorithm under different bandwidth constraints, as well as a distributed heuristic algorithm, which can be practically applied in real system and outperforms conventional ad hoc strategies by about 10 percent-70 percent in both single rate and multirate scenario. For future work, we will study how to maximize the blocks delivered over a horizon of several periods, taking into account the interdependence between the periods, and will analyze the gap between the global optimal solution and the proposed distributed algorithm. We also would like to study how to adapt the parameter γ in terms of the network condition in our proposed distributed algorithm. Besides, based on our current results, we are interested in combing the video coding technique to our algorithm to further improve the user perceived quality.

Appendix

Proof of 1

Our proof is simple. Applying the flow balance requirement on each node (i.e., $\sum_{j:(i,j)\in A} f(i,j) - \sum_{j:(j,i)\in A} f(i,j) = b(i) = 0$, $\forall i \in V$), we can use the flow amount $f(n_{ki}, b_{ij})$ on arcs (n_{ki}, b_{ij}), (where $i \in N$, $j \in D_i$, and $k \in NBR_i$) to represent the flow amount on all the other arcs. Therefore, due to the flow balance on vertices:

1. $b_{ij} (\forall i \in N, \forall j \in D_i)$.
2. $r_i (\forall i \in N)$.
3. $n_{ik} (\forall k \in N, i \in NBR_k)$.
4. $s_k (\forall k \in N)$, we have

$$
\begin{aligned}
&1.\quad f(b_{ij}, r_i) = \sum_{k \in NBR_i} f(n_{ik}, b_{ij}),\\
&2.\quad f(r_i, t) = \sum_{j \in D_i} f(b_{ij}, r_i) = \sum_{j \in D_i} \sum_{k \in NBR_i} f(n_{ik}, b_{ij}),\\
&3.\quad f(s_k, n_{ki}) = \sum_{j \in D_i} f(n_{ik}, b_{ij}),\\
&4.\quad f(s, s_k) = \sum_{i \in N} f(s_k, n_{ki}) = \sum_{i \in N} \sum_{j \in D_i} f(n_{ik}, b_{ij}).
\end{aligned} \tag{8}
$$

Due to the capacity constraints on each arcs

1. (b_{ij}, r_i), $\forall i \in N$, $\forall j \in D_i$.
2. (r_i, t), $\forall i \in N$.
3. (s, s_k), $\forall k \in N \cup \{0\}$.
4. (s_k, n_{ki}), $\forall i \in N$, and $\forall k \in NBR_i$, we have the following constraints:

$$
\begin{aligned}
&1.\quad f(b_{ij}, r_i) = \sum_{k \in NBR_i} f(n_{ik}, b_{ij}) \le 1\\
&2.\quad f(r_i, t) = \sum_{j \in D_i} \sum_{k \in NBR_i} f(n_{ik}, b_{ij}) \le \tau I_i\\
&3.\quad f(s, s_k) = \sum_{i \in N} \sum_{j \in D_i} f(n_{ik}, b_{ij}) \le \tau O_3\\
&4.\quad f(s_k, n_{ki}) = \sum_{j \in D_i} f(n_{ik}, b_{ij}) \le \tau E_{ki}.
\end{aligned} \tag{9}
$$

Due to the capacity constraint on arcs $(n_{ki}, b_{i,j})$, $\forall i \in N$, $\forall j \in D_i$, $\forall k \in NBR_i$, and all the flow amount is integer, we have

$$
f(n_{ki}, b_{i,j}) \in \{0, 1\}. \tag{10}
$$

The objective of the min-cost network to minimize the sum of the cost among all arcs. Since only arcs (b_{ij}, r_i), $\forall i \in N$, and $\forall j \in D_i$ have nonzero unit cost, we have the objective function of the min-cost flow problem:

$$\begin{aligned} &\min \sum_{(i,j)\in A} c(i,j)f(i,j) \\ &= \min \sum_{\forall i\in N} \sum_{\forall j\in D_i} (-P_{ij}) f(b_{ij}, r_i) \\ &= \min \sum_{\forall i\in N} \sum_{\forall j\in D_i} \sum_{k\in NBR_i} (-P_{ij}) f(n_{ik}, b_{ij}), \end{aligned} \tag{11}$$

$$\Leftrightarrow$$

$$\frac{1}{|N|} \max \sum_{\forall i\in N} \sum_{\forall j\in D_i} \sum_{k\in NBR_i} P_{ij} f(n_{ik}, b_{ij}). \tag{12}$$

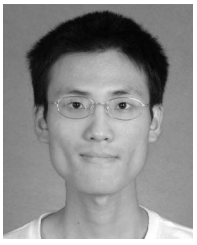
From (12) and (9), we observe that the flow amounts on arcs $(n_{ki}, b_{i,j})$, $\forall i \in N$, $\forall j \in D_i$, $\forall k \in NBR_i$ have the same constraints as x_{kj}^i in global BSP (3). And, (10) demonstrates that the min-cost flow problem has an equivalent objective function as global BSP (3). Therefore, we prove Theorem 1. □

ACKNOWLEDGMENTS

The authors thank the following institutions for their support: National Basic Research Program of China under Grant 2006CB303103, NSFC under Grants 60503063 and 60773158, the RGC under contracts CERG 622407 and N_HKUST609/07, the NSFC Oversea Young Investigator Grant under Grant 60629203, the 863 Program under Grant 2006AA01Z321, and the Key Project of Guangzhou Municipal Government Guangdong/Hong Kong Critical Technology grant 2006Z1-D6131.

REFERENCES

[1] Gridmedia, http://www.gridmedia.com.cn/, 2006.
[2] Pplive, http://www.pplive.com/, 2008.
[3] Zatto, http://www.zattoo.com/, 2008.
[4] V. Pai et al., "Chainsaw: Eliminating Trees from Overlay Multicast," *Proc. IEEE INFOCOM '05,* Feb. 2005.
[5] V. Agarwal and R. Rejaie, "Adaptive Multi-Source Streaming in Heterogeneous Peer-to-Peer Networks," *Proc. Multimedia Computing and Networking (MMCN '05),* Jan. 2005.
[6] X. Zhang, J. Liu, B. Li, and T.-S.P. Yum, "Coolstreaming/Donet: A Data-Driven Overlay Network for Efficient Media Streaming," *Proc. IEEE INFOCOM '05,* Mar. 2005.
[7] M. Zhang, J.-G. Luo, L. Zhao, and S.-Q. Yang, "A Peer-to-Peer Network for Live Media Streaming Using a Push-Pull Approach," *Proc. ACM Multimedia,* Nov. 2005.
[8] N. Magharei and R. Rejaie, "Prime: Peer-to-Peer Receiver-Driven Mesh-Based Streaming," *Proc. IEEE INFOCOM '07,* May 2007.
[9] B. Cohen, http://bitconjuer.com, 2008.
[10] V. Venkataraman and P. Francis, "Chunkyspread: Multi-Tree Unstructured End System Multicast," *Proc. Int'l Workshop Peer-to-Peer Systems (IPTPS '06),* Feb. 2006.
[11] V. Venkataraman and P. Francis, "On Heterogeneous Overlay Construction and Random Node Selection in Unstructured P2P Networks," *Proc. IEEE INFOCOM '06,* Apr. 2006.
[12] J. Jiang and K. Nahrstedt, "Randpeer: Membership Management for QoS Sensitive Peer-to-Peer Applications," *Proc. IEEE INFOCOM '06,* Apr. 2006.
[13] D. Kostic et al., "Maintaining High Bandwidth under Dynamic Network Conditions," *Proc. Usenix Ann. Technical Conf.,* 2005.
[14] A.R. Bharambe, C. Herley, and V.N. Padmanabhan, "Analyzing and Improving a Bittorrent Networks Performance Mechanisms," *Proc. IEEE INFOCOM '06,* Apr. 2006.
[15] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin, *Network Flows: Theory, Algorithms, and Applications.* Prentice Hall.
[16] S. Saroiu, P. Gummadi, and S. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems," *Proc. ACM/SPIE Multimedia Computing and Networking (MMCN),* 2002.
[17] S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-Driven Layered Multicast," *Proc. ACM SIGCOMM '96,* Sept. 1996.
[18] Y. Cui and K. Nahrstedt, "Layered Peer-to-Peer Streaming," *Proc. Int'l Workshop Network and Operating Systems Support for Digital Audio and Video (NOSSDAV),* 2003.
[19] J. Zhao, F. Yang, Q. Zhang, Z. Zhang, and F. Zhang, "Lion: Layered Overlay Multicast with Network Coding," *IEEE Trans. Multimedia,* vol. 8, no. 5, Oct. 2006.
[20] K.C. Ellen, W. Zegura, and S. Bhattacharjee, "How to Model an Internetwork," *Proc. IEEE INFOCOM,* 1996.
[21] A. Goldberg, *Andrew Goldberg's Network Optimization Library,* http://www.avglab.com/andrew/soft.html, 2008.
[22] N. Magharei and R. Rejaie, "Understanding Mesh Based Peer-to-Peer Streaming," *Proc. ACM Int'l Workshop Network and Operating Systems Support for Digital Audio and Video (NOSSDAV),* 2006.
[23] M. Zhang, L. Zhao, Y. Tang, J. Luo, and S. Yang, "Large-Scale Live Media Streaming over Peer-to-Peer Networks through Global Internet," *Proc. ACM Workshop Advances in Peer-to-Peer Multimedia Streaming (P2PMMS '05),* pp. 21-28, Nov. 2005.
[24] CCTV Online TV, http://www.cctv.com/p2p/index.htm, 2007.
[25] Y. Chu, S. Rao, and H. Zhang, "A Case for End System Multicast," *Proc. ACM Sigmetrics '00,* June 2000.
[26] http://www.cs.umd.edu/~suman/research/myns/index.html, 2002.
[27] D. Li, Y. Cui, K. Xu, and J. Wu, "Segment-Sending Schedule in Data-Driven Overlay Network," *Proc. IEEE Int'l Conf. Comm. (ICC),* 2006.
[28] China Central Radio and TV University, http://www.crtvu.edu.cn/, 2006.
[29] S. Banerjee and B. Bhattacharjee, *A Comparative Study of Application Layer Multicast Protocols,* http://www.cs.wisc.edu/~suman/pubs.html, 2002.
[30] P. Francis, *Yoid: Extending the Internet Multicast Architecture,* white paper, http://www.icir.org/yoid/, 2006.
[31] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron, "Scribe: A Large-Scale and Decentralized Application-Level Multicast Infrastructure," *IEEE J. Selected Areas in Comm.,* vol. 20, no. 8, 2002.
[32] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A.I.T. Rowstron, and A. Singhr, "Splitstream: High-Bandwidth Multicast in Cooperative Environments," *Proc. Symp. Operating Systems Principles (SOSP '03),* Oct. 2003.
[33] N. Magharei, R. Rejaie, and Y. Guo, "Mesh or Multiple-Tree: A Comparative Study of P2P Live Streaming Services," to appear in *Proc. IEEE INFOCOM '07,* May 2007.



Meng Zhang received the BEng degree in computer science and technology from Tsinghua University, Beijing, in 2004. He is currently a PhD candidate in the Department of Computer Science and Technology, Tsinghua University. He is also the key designer and developer of GridMedia, one of the earliest very large scale peer-to-peer live streaming system in the world. His research interests include multimedia networking, particularly in QoS issue of peer-to-peer streaming, peer-to-peer video-on-demand, multimedia streaming on overlay networks, etc. He has published more than 10 papers in multimedia networking. He is a student member of the IEEE.

Yongqiang Xiong received the BS, MS, and PhD degrees, in computer science from Tsinghua University, Beijing, in 1996, 1998, and 2001, respectively. He is with the Wireless and Networking Group, Microsoft Research Asia, as a researcher. He has published about 20 referred papers and served as a TPC member or reviewers for the international key conferences and leading journals in wireless and networking area. His research interests include peer-to-peer networking, routing protocols for both MANETs and overlay networks, and network security. He is a member of the IEEE.

Qian Zhang received the BS, MS, and PhD degrees from Wuhan University, China, in 1994, 1996, and 1999, respectively. She joined the Hong Kong University of Science and Technology in September 2005 as an associate professor. Before that, she was in Microsoft Research, Asia, Beijing, from July 1999, where she was the research manager of the Wireless and Networking Group. She has published about 150 refereed papers in international leading journals and key conferences proceedings on wireless/Internet multimedia networking, wireless communications and networking, and overlay networking. She is the inventor of about 30 pending patents. Her current research interests include wireless communications, IP networking, multimedia, P2P overlay, and wireless security. She also participated in many activities in the IETF ROHC (Robust Header Compression) WG Group for TCP/IP header compression. She is an associate editor for the *IEEE Transactions on Wireless Communications*, *IEEE Transactions on Multimedia*, *IEEE Transactions on Vehicular Technologies*, *Computer Networks*, and *Computer Communications*. She also served as a guest editor for *IEEE JSAC*, *IEEE Wireless Communications*, *Computer Networks*, and *ACM/Springer MONET*. She has been involved in the organization committee for many important IEEE conferences, including ICC, GLOBECOM, WCNC, INFOCOM, etc. She has received TR 100 (MIT Technology Review) world's top young innovator award. She also received the Best Asia Pacific (AP) Young Researcher Award elected by IEEE Communication Society in 2004. She received the Best Paper Award in Multimedia Technical Committee (MMTC) of the IEEE Communication Society and Best Paper Award in QShine 2006. She received the Oversea Young Investigator Award from the National Natural Science Foundation of China (NSFC) in 2006. She is the vicechair of the Multimedia Communication Technical Committee of the IEEE Communications Society. She is a senior member of the IEEE.

Lifeng Sun received the BS and PhD degrees in system engineering from the National University of Defense Technology, Changsha, Hunan, China, in 1995 and 2000, respectively. His professional interests include ad hoc network streaming, peer-to-peer video streaming, interactive multiview video, and distributed video coding. He has published more than 50 papers in the above domain. He was on the postdoctor research of the Department of Computer Science and Technology, Tsinghua University from 2001 to 2003. He is currently on the Faculty of the Department of Computer Science and Technology, Tsinghua University. He is the secretary general of Multimedia Committee of China Computer Federation, vice director of Tsinghua-Microsoft Media and Network Key Lab, Ministry of Education (MOE). He serviced as a TPC member of the Pacific-Rim Conference on Multimedia (PCM) 2005, Pacific-Rim Conference on Multimedia (PCM) 2006, IEEE International MultiMedia Modelling Conference (MMM) 2007 and IEEE International MultiMedia Modelling Conference (MMM) 2006. He reviewed papers for IEEE ICME 2006 and IEEE CSVT special issue on multiview video coding. He is a member of the IEEE.

Shiqiang Yang received the BS and MS degrees in computer science from Tsinghua University, Beijing, in 1977 and 1983, respectively. He is a chief professor at Tsinghua University with the Department of Computer Science and Technology. From 1980 to 1992, he worked as an assistant professor at Tsinghua University, Beijing. From 1992 to 1994, he visited the City University of Hong Kong and was a research assistant at the Department of Manufacture Engineering. As the associate header of the Department of Computer Science and Technology, Tsinghua University, since 1994, he served as the associate professor from 1994 to 1999 and then as the professor since 1999. He is currently the president of the Multimedia Committee of China Computer Federation, Beijing, and the codirector of Microsoft-Tsinghua Multimedia Joint Lab, Tsinghua University. His research interests mainly include multimedia application, video procession, streaming media, and embedded multimedia. In the recent three years, he has published more than 60 papers in international conferences and journals, as well as more than 10 proposals within MPEG and AVS standardization. He is a member of the IEEE.