

# On the Optimal Scheduling for Media Streaming in Data-driven Overlay Networks\*

Meng ZHANG\*, Yongqiang XIONG<sup>†</sup>, Qian ZHANG<sup>‡</sup>, Shiqiang YANG\*

\* Department of Computer Science and Technology, Tsinghua University, Beijing, China,

zhangmeng00@mails.tsinghua.edu.cn, yangshq@tsinghua.edu.cn

<sup>†</sup> Microsoft Research Asia, Beijing, China, Yongqiang.Xiong@microsoft.com

<sup>‡</sup> Hong Kong University of Science and Technology, Hong Kong, China, qianzh@cs.ust.hk

**Abstract**—The Internet has witnessed a rapid growth in deployment of data-driven overlay network (DON) based streaming applications during recent years. In these applications, each node independently selects some other nodes as its neighbors (i.e. overlay construction), and exchanges streaming data with these neighbors (i.e. data scheduling). This scheme improves the robustness of the system. However, most of the work in the literature focused on the construction problem, and very few addressed its scheduling problem which is also very important for the overall performance. In this paper, we analytically study the scheduling problem in DON 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 maximize the system throughput. Experimental results indicate that our algorithms outperform other schemes and the throughput gain is up to 80%.

## I. INTRODUCTION

During recent years, the Internet has witnessed a rapid growth in deployment of peer-to-peer (P2P) streaming especially the data-driven overlay network (DON) based streaming applications [1]–[4]. DON achieves great success due to its robustness to high churn of participating nodes. Recent exciting reports show that this type of system has the power to enable over 200,000 users simultaneously watching a hot live event with relatively high bit rate (300-500 Kbps) by only one streaming server [5], [6].

The basic idea of DON is rather simple and similar to bit-torrent [7]. The protocol contains two steps. The first step is overlay construction. In this step, each node independently selects its neighbors so as to form an unstructured overlay network, and notifies its neighbors what streaming data it has. The second step is the streaming scheduling. In this step, each node requests the absent data from its neighbors according to the notification of data availability, and sends the data it has to those neighbors who ask for it. Since all nodes are equivalent and independent in DON, this two-step scheme is robust with low maintenance cost for the entire system, and its performance relies on the algorithms in these two steps.

However, in the literature most of the related research works focused on the construction problem of the first step. Researchers proposed different schemes to build unstructured

overlays to improve its efficiency or robustness [8]–[10], but there are few schemes addressing the scheduling problem of the second step except for some empirical studies, furthermore, there’s no theoretical study in DON on optimal streaming scheduling, namely, how each node optimally decides from which neighbor to request which block and optimally allocates its limited outbound bandwidth to every neighbor, in order to maximize the throughput with different bandwidth constraints in such an unstructured overlay network.

In this paper, we present our analytical model and the corresponding solutions to tackle the streaming scheduling problem in DON. 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. This solution is centralized and requires global knowledge, making it impractical in real system. Based on its basic idea, we then propose a heuristic algorithm which is fully distributed and asynchronous with only local information exchange. Simulation results indicate that our distributed algorithm outperforms other different existing methods by about 10%~80% gains in high streaming rate. The remainder of this paper is organized as follows: Section II presents our model for the scheduling problem and gives the global optimal scheduling algorithm. Section III describes our heuristic distributed algorithm. We conduct simulations to evaluate the performance of our algorithm in section IV. Section V introduces the related works. Finally, conclusion and future work are stated in section VI.

## II. MODEL AND SOLUTION

### A. Block Scheduling Problem

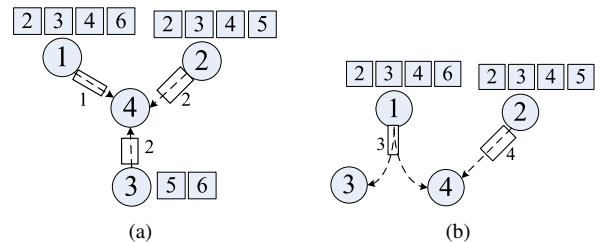


Fig. 1. Illustration of block scheduling problem

In DON, the streaming media is partitioned into blocks, and there is an exchanging window on each node containing the blocks that the node is interested in and sliding forwards

\*Work performed when Meng was an intern at Microsoft Research Asia and supported by the National Natural Science Foundation of China under Grant No. 60432030 and 973 Program under Grant No. 2006CB303103

continuously. Every node periodically notifies each of its neighbors with a bit vector called "buffer map" representing the blocks availability in its exchanging window to declare what blocks it has. Then nodes will request their absent blocks from neighbors according to that declaration.

To maximize the throughput of the system, our approach is to increase the number of blocks that is requested successfully under bandwidth constraints as much as possible within every scheduling period. We illustrate it in Fig. 1(a), the node 4 in the center tries to request blocks from the other three nodes. The numbers beside the pipes represent the outbound bandwidth of that link, indicating the maximal blocks that can be uploaded within one interval. One optimal scheduling is to request block 2 from S1, blocks 3, 4 from node 2 and blocks 5, 6 from node 3, that is, all the blocks can be obtained in one period. Yet, neither random nor rarest-first strategy can give this scheduling. Fig. 1(b) shows the 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 choices. The best way is that node 4 requests blocks from node 2 while node 3 requests from node 1. The real situation is more complicated because different blocks may have different importance and the bottlenecks are not only at the last mile. In general, our aim is to maximize the sum of priorities of all requested blocks in the overlay under different bandwidth constraints.

TABLE I  
NOTATIONS

Notation	Description
$N$	Set of all nodes in the overlay except the source node 0, that is, $N = \{1, \dots,  N \}$ node
$r$	The streaming rate
$I_i, i = 0, \dots,  N $	The inbound bandwidth capacity of node $i$
$O_i, i = 0, \dots,  N $	The outbound bandwidth capacity of node $i$
$E_{ik}, i = 0, \dots,  N $	The end-to-end 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 $i$
$\tau$	The request period
$W_T$	The exchanging window size scaled by time
$C_i$	The current clock time at node $i$
$d_j^i$	Play out deadline of block $j$ at node $i$
$D_i$	Set of all desired blocks in the current exchanging window of node $i$ : for any block $j \in D_i$ , it satisfies $C_i < d_j^i < C_i + W_T, h_{ij} = 0$

## B. Model

First, we define some notations that are used in the formulation as shown in TABLE I. In DON, 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. Hence defining different priority for blocks is important. Two factors have been considered in our priority definition: previous empirical study has shown that "rarest-first" is a very efficient strategy in data dissemination [7], [11], so rarity factor is considered first; while as streaming application has real-time constraint, the second factor we considered is emergency factor: A block in danger of being

delayed beyond the deadline should be of more priority than the one just entering the exchanging window. Consequently, we define the priority  $P_j^i$  of block  $j \in D_i$  for node  $i \in N$ :

$$P_j^i = \beta P_R(\sum_{k \in NBR(i)} h_k^j) + (1 - \beta) P_E(C_i + W_T - d_j^i) \quad (1)$$

Here  $\beta$  satisfies  $0 \leq \beta \leq 1$ . The first part represents the rarity priority, where  $\sum_{k \in NBR(i)} h_k^j$  is the number of neighbors that hold block  $j$ . The second part is the emergency priority, where  $C_i + W_T - d_j^i$  is the remaining time of block  $j$  till the deadline. And we use functions  $P_R(*)$  and  $P_E(*)$  are to obtain the rarity and emergency priority value. Then we define the *determined variable*  $x_{kj}^i$  to denote whether node  $i \in N$  requests block  $j \in D_i$  from its neighbor  $k \in NBR_i$ :

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

We call set  $\{x_{kj}^i\}$  a block scheduling in one period. Our target is to maximize the average priority sum of each node:

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

Constraints a) ~ c) ensure that the block scheduling should satisfy the inbound, outbound, and end-to-end available bandwidth constraints. Constraint d) ensures that each block should be fetched from at most one neighbor so that no duplicate blocks arrive. The 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).

## C. Solution

In this section, we show that the global BSP can be transformed into an equivalent minimum cost flow problem that can be solved within polynomial time. The Min-Cost Flow Problem (MCFP for short) is briefly depicted as follows [12]. 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  $l_{ij}$  and  $u_{ij}$  is associated to the arc  $(i, j)$  to denote the minimum and maximum amount that can flow through the arc. Let  $f_{ij}$  denote the flow amount on arc  $(i, j)$ , which is the decision variables. The MCFP is an optimization model formulated as follows:

$$\begin{aligned} & \min \sum_{(i,j) \in A} c_{ij} f_{ij} \\ & \text{s.t.} \\ & \text{(a)} \quad \sum_{j:(i,j) \in A} f_{ij} - \sum_{j:(j,i) \in A} f_{ij} = b(i), \forall i \in V \\ & \text{(b)} \quad l_{ij} \leq f_{ij} \leq u_{ij}, \forall (i,j) \in A \\ & \text{where } \sum_{i=1}^n b(i) = 0 \text{ and } f_{ij} \in Z^+ \end{aligned} \quad (4)$$

Such min-cost flow problem can be solved in polynomial time. In our model, we let  $b(i) = 0, \forall i \in V$ . We can start the transformation with the rules in TABLE II, where rules a) ~ e) and f) ~ k) respectively give the vertex and arc meanings.

TABLE II  
TRANSFORMATION RULES

a)	Put two virtual vertices in set $V$ : $v_S$ (the <b>source vertex</b> ) and $v_T$ (the <b>sink vertex</b> );
b)	$\forall i \in N$ , insert a vertex $v_i^r$ to $V$ , called <b>receiver vertex</b> ;
c)	$\forall i \in N \cup \{0\}$ , insert a vertex $v_i^s$ to $V$ , called <b>sender vertex</b> ;
d)	If node $k \in NBR_i$ , insert a vertex $v_{ki}^n$ to $V$ , called <b>neighbor vertex</b> ;
e)	If block $j \in D_i$ (the desired block $j$ is in the current exchanging windows of node $i$ ), then insert a vertex $v_{ij}^b$ to $V$ , called <b>block vertex</b> ;
f)	Arcs between source vertex and sender vertices (outbound bandwidth capacity constraints): insert an arc $(v_S, v_i^s)$ to $A$ , where $i \in N \cup \{0\}$ . The capacity of this arc is $\tau O_i$ (that is the maximal blocks which could be sent out from node $i$ in one period) and the unit cost is 0;
g)	Arcs between receiver vertices and sink vertex (inbound bandwidth capacity constraints): insert an arc $(v_i^r, v_T)$ to $A$ , where $i \in N$ . The capacity of the arc is $\tau I_i$ , and the unit cost is 0;
h)	Arcs between sender vertices and neighbor vertices (end-to-end available bandwidth constraints): if $v_k^s \in V$ and $v_{ki}^n \in V$ , insert an arc $(v_k^s, v_{ki}^n)$ to $A$ , where the capacity is $\tau E_{ki}$ (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 $k \in NBR_i, h_{kj} = 1, v_{ki}^n \in V$ and $v_{ij}^b \in V$ , that is, node $k$ holds block $j$ and node $i$ has not received block $j$ , then insert an arc $(v_{ki}^n, v_{ij}^b)$ 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 $v_{ij}^b \in V$ , then insert an arc $(v_{ij}^b, v_i^r)$ to $A$ , where the unit cost is $-P_j^i/ N $ and the capacity is 1.
k)	Insert an assistant arc $(v_T, v_S)$ to $A$ , where the unit cost is a very negative value $-M$ and the capacity is $+\infty$ .

Applying these rules, we can transform global BSP (3) into a corresponding MCFP, and we have the following proposition:

**Proposition 1:** The optimal flow amount  $f(v_{ki}^n, v_{ij}^b)^*$  of the corresponding MCFP on arcs  $(v_{ki}^n, v_{ij}^b)$  ( $\forall i \in N, j \in D_i$  and  $k \in NBR_i$ ) is also an optimal solution to the global BSP.

Due to page limitation, the proof is omitted here. We take an example to illustrate it intuitively. Suppose that there are 4 nodes in the overlay whose available blocks are shown in Fig. 2, and they are neighbors each other. The corresponding equivalent min-cost

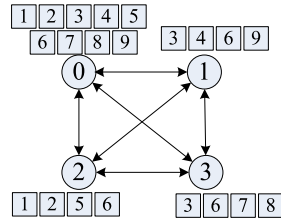


Fig. 2. An example for global BSP

network flow problem is shown in Fig. 3. The left and right values marked on arcs represent the capacity and the cost of unit flow respectively. The flow amount on arcs  $(v_S, v_i^s), i = 0, 1, 2, 3$  and arcs  $(v_i^r, v_T), i = 0, 1, 2, 3$  stands for the number of blocks uploaded and downloaded in one period separately. The flow amount on arcs  $(v_k^s, v_{ki}^n), k, i = 0, 1, 2, 3$  is the number of blocks transmitted from node  $k$  to  $i$  in one period. Applying this algorithm, we can obtain a global optimal block scheduling under different bandwidth constraints.

### III. HEURISTIC DISTRIBUTED ALGORITHM

We give the algorithm to do global optimal scheduling in Section II. However, requiring global knowledge (such as

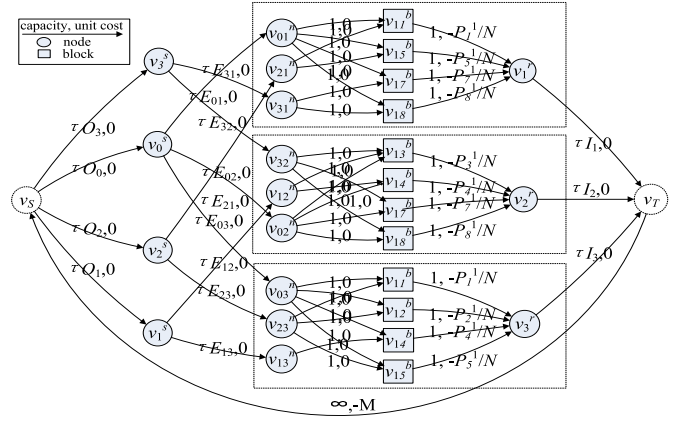


Fig. 3. An example of the equivalent MCFP

block availability, bandwidth information) and request synchronization make the algorithm not scalable. In this section, we design our heuristic fully distributed and asynchronous algorithm.

As shown in Fig. 3, the sub network flow problems in the dashed rectangles can be computed distributedly on each node. Yet, the trouble is that we can't determine the flow amount on arcs  $(v_k^s, v_{ki}^n)$ , that is, every node is not able to predict how much bandwidth its neighbors can allocate to it. If neighbor  $k$  of node  $i$  promises to allocate node  $i$  an amount  $W_k^i$  of bandwidth ( $W_{ki} < E_{ki}, W_{ki} < O_k$ ), then a local block scheduling can be computed on node  $i$  as follow and corresponding min-cost network flow problem can be also naturally derived applying the rules as described in TABLE II.

$$\begin{aligned}
 & \max \sum_{j \in D_i} \sum_{k \in NBR_i} P_j^i h_{kj} x_{kj}^i \\
 & \text{s.t.} \\
 & \text{(a) } \sum_{k \in NBR_i} x_{kj}^i \leq 1, \forall j \in D_i \\
 & \text{(b) } \sum_{j \in D_i} \sum_{k \in NBR_i} x_{kj}^i \leq \tau I_i, \forall i \in N \\
 & \text{(c) } \sum_{j \in D_i} x_{kj}^i \leq \tau W_{ki}, \forall i \in N, k \in NBR_i \\
 & \text{(d) } x_{kj}^i \in \{0, 1\}, \forall k \in NBR_i, j \in D_i
 \end{aligned} \tag{5}$$

TABLE III  
HEURISTIC DISTRIBUTED ALGORITHM

a)	Node $i$ estimates the bandwidth $W_{ki}^{(m+1)}$ that its neighbor $k$ can allocate it in the $(m+1)^{th}$ period with the traffic received from that neighbor in the previous $M$ periods, as shown in equation 5;
b)	Based on $W_{ki}^{(m+1)}$ , 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.

We use a very simple way to estimate the bandwidth that can be allocated to node  $i$  with historical information. Let  $q_{ki}^{(m)}$  denote the total number of blocks arrived at node  $i$  from neighbor  $k$  in the  $m^{th}$  period. In each request, we use the average traffic from node  $k$  to  $i$  in previous  $M$  periods to estimate the bandwidth  $W_{ki}^{(m+1)}$  that neighbor  $k$  can allocate node  $i$  in the  $(m+1)^{th}$  period,

$$W_{ki}^{(m+1)} = \gamma \cdot \left( \sum_{l=m-M+1}^m q_{ki}^{(l)} / M\tau \right) \tag{6}$$

$\gamma (> 1)$  is a constant called aggressive coefficient. With  $\gamma$  increasing, the outbound bandwidth of its neighbors can be

utilized with more possibility; however, the request may be more congested on its neighbors. For a new neighbor  $j$  of node  $i$ , we simply set the initial bandwidth as  $W_{ki} = \gamma \cdot r / |NBR_i|$ . We summarize our distributed algorithm. In the beginning of each request period, do the following steps in TABLE III.

#### IV. PERFORMANCE EVALUATION

##### A. Simulation Configuration

As aforementioned, there are two key steps for supporting streaming applications in overlay networks, and we focus on the streaming 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. Moreover, to evaluate the performance, we define *delivery ratio* to represent the number of blocks that arrive at each node before playback deadline over the total number of blocks encoded. The average delivery ratio indicates the throughput of the whole system and reflects the average continuity quality observed by users.

In our experiment, we implement a discrete event-driven P2P simulator and use "CS2" library [13] to solve min-cost network flow problem. We adopt the method in DONet [1] to reduce the control overhead: dozens of block requests are piggybacked in one request packet, and each node sends out requests to its neighbors periodically. We set the period 2 seconds. In our configuration, the group size of overlay is of 500 nodes and each node has 15 neighbors. Each block has the same size of 1280 bytes (10Kbit). Each node estimates the bandwidth allocated from a neighbor with the traffic received from it in previous 5 periods, namely,  $M = 5$ . For priority parameter, we set  $P_R(a) = 10^8 - a, a = 1, \dots, 8, P_R(a) = 1, a \geq 9$ , and  $\beta = 1$ . Moreover, the aggressive coefficient used is  $\gamma = 1.5$ .

##### B. Performance Comparison

In this subsection, we show the performance of our proposed algorithms (global optimal and distributed algorithm) and perform comparisons with Chainsaw [2], DONet [1] and Round-Robin method: Chainsaw uses a purely random strategy to decide what blocks to request from neighbors; DONet employs a rarest-first strategy as the block scheduling method, and selects suppliers with the most surplus bandwidth and enough available time first; Round-Robin method (used in PALS [3] especially designed for layered streaming) assigns blocks to each supplier proportionally by weighted round-robin strategy due to the bandwidth from the senders, and in our experiment, 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.

In Fig. 4, we investigate the performance of our algorithms when all the nodes are DSL nodes with low bandwidth. There are three types of nodes in terms of inbound bandwidth: 40% users with 512Kbps inbound bandwidth capacity, 30% with 1Mbps and 30% with 2Mbps. And for all users, the outbound bandwidth capacity is half of the inbound bandwidth. In such

configuration, the bottlenecks are set only at the last mile. The exchanging window size is set to 10 seconds. We see that when the streaming rate is low (such as 250Kbps), all the methods have high delivery ratio. However, with the streaming rate increasing, the performance of the three compared methods goes down fast. At the rate of 500Kbps, the global optimal algorithm can still keep a delivery ratio of nearly 99% and the proposed distributed algorithm has a delivery ratio of 95% which outperforms the DONet method and the other two methods by gains of about 20% and 58%.

Based on the setting to Fig. 4, we add the constraint that the end-to-end available bandwidth is between 10Kbps and 150Kbps in Fig. 5. We note that the performance of our algorithms and DONet method do not descend much. However, Chainsaw and Round-Robin method decrease more. This is because our algorithm and DONet method not only employ rarity as a heuristic factor but also are both bandwidth-aware. As shown in Fig. 5, our distributed algorithm has 20% gains compared to DONet method and over 80% gains to the other ones. In Fig. 6, we modified the exchanging window size to 5 seconds. With a smaller window size, the delivery ratio of all methods reduces. As the request period is set to 2 seconds, most of the blocks can only be request repeatedly for two times. Our proposed algorithm outperforms the others because our algorithm can request blocks successfully with much more probability in every period. In Fig. 7, we verify the performance when there are some strong nodes in the overlay and the LAN nodes take 10% over total. The remainders are DSL users with the same proportion for different access bandwidth as in Fig. 4. As expected, the delivery ratio of all the methods improved much since the bandwidth resource gets richer. However, the delivery ratio of our distributed algorithm exceeds the others by no less than 10%. Fig. 8 and Fig. 9 show the delivery ratio under different group size without and with end-to-end available bandwidth constraints respectively, when the streaming rate is 500Kbps. We can see 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 they all have good scalability. However, our proposed distributed algorithm always has a gain of 10%~80% compared to different existent methods.

##### C. Overhead

The control overhead of our algorithm is very low as described in section III. Simulation results show that our scheme adds no more than 2% communication control overhead when the streaming rate is 500Kbps and the neighbor count is 15. Computation overhead is also not high. When the streaming rate is 500Kbps with block size of 1280bytes, the block rate is 50 blocks/s. For a 2-second request period, there are 100 blocks to schedule. Each node with 15 neighbors spends no more than 12 milliseconds for one-period scheduling on a PIII 600MHz PC.

#### V. RELATED WORK

Traditional application layer multicast (ALM) can be classified into three categories in terms of the tree building

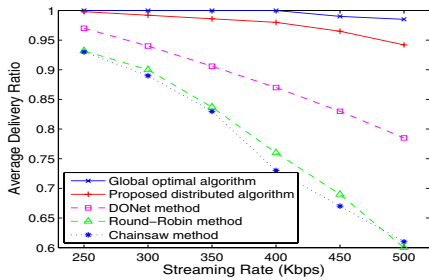


Fig. 4. All are DSL nodes with exchanging window of 10 sec and bottlenecks only at the last mile. Group size is 500

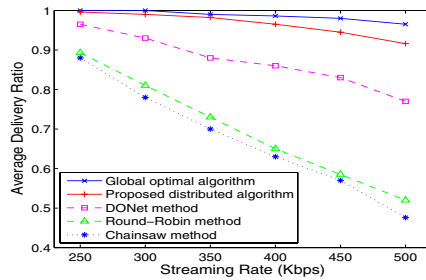


Fig. 5. All are DSL users with exchanging window of 10 sec and end-to-end available bandwidth 10~150Kbps. Group size is 500

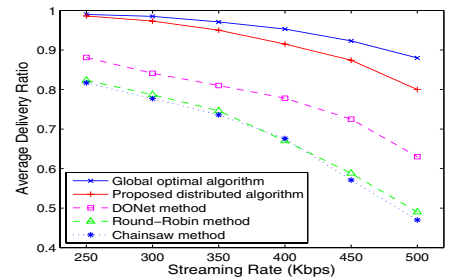


Fig. 6. All are DSL users with exchanging window of 5 sec and bottlenecks only at last mile. Group size is 500

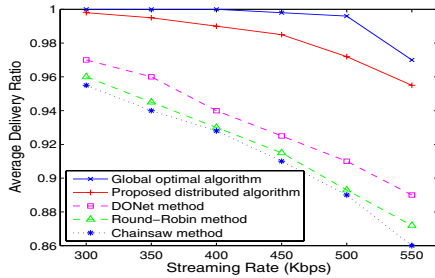


Fig. 7. 10% 10M LAN users with exchanging window of 10 sec and maximum end-to-end available bandwidth 1.5Mbps. Group size is 500

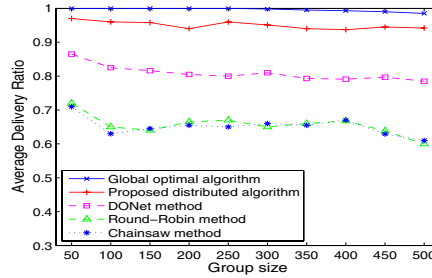


Fig. 8. All are DSL users with exchanging window of 10 sec and bottlenecks only at the last mile. Streaming rate is 500 Kbps.

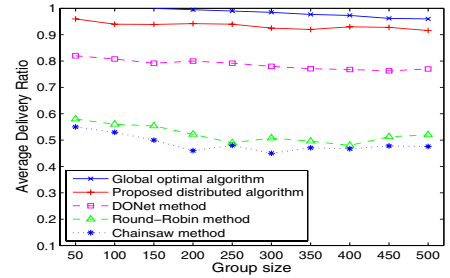


Fig. 9. All are DSL users with exchanging window of 10 sec and end-to-end available bandwidth 10~150Kbps. Streaming rate is 500Kbps

approaches [14]: tree-first, mesh-first, implicit protocol. Rather than building trees, some protocols for building unstructured overlay have also been proposed. In unstructured overlay network, trees are not built and some form of random node selection is employed. These works include [8]–[10]. Whereas, they mainly focus on the overlay construction problem while not addressing the scheduling issue. Data-driven protocol for streaming applications adopts the idea of unstructured overlay. Each node retrieves the absent data from its neighbors according to the notification of data availability. Such works include Chainsaw [2], DONet [1], PALS [3]. Chainsaw uses a purely random strategy to decide what packets to request from neighbors. DONet employs a rarest-first strategy as the packet scheduling method (greedy method), and selects suppliers with the highest bandwidth and enough available time first. PALS (a receiver driven protocol) is designed for layered streaming. It uses a diagonal buffer distribution (snake shape) to request absent blocks, and assigns blocks to each supplier proportionally by weighted round-robin strategy due to the bandwidth measured by TFRC protocol from the senders. However, all these works employ straight forward methods to do block scheduling. How to do optimal block scheduling has not been addressed in the previous literature.

## VI. CONCLUSION AND FUTURE WORK

In this paper, we investigate the scheduling problem in the data-driven overlay (DON) based streaming systems. 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 DON. 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 existent

methods by about 10%~80%. For future work, we will study how to maximize the blocks delivered over a horizon of several periods, taking into account the inter-dependence between the periods. We will also do more experiments on examining the parameter sensitivities in our algorithm by both simulations and real world experiments on Planet-Lab.

## REFERENCES

- [1] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum, "Coolstreaming/donet: A data-driven overlay network for efficient media streaming," in *IEEE INFOCOM 2005*, Miami, US, Mar. 2005.
- [2] V. Pai, K. Kumar, and et al, "Chainsaw: Eliminating trees from overlay multicast," in *IEEE INFOCOM 2005*, Conell, US, Feb. 2005.
- [3] V. Agarwal and R. Rejaie, "Adaptive multi-source streaming in heterogeneous peer-to-peer networks," in *Multimedia Computing and Networking 2005 (MMCN)*, San Jose, CA, USA, Jan. 2005.
- [4] 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," in *ACM Multimedia 2005*, Singapore, Nov. 2005.
- [5] GridMedia, "http://www.gridmedia.com.cn/".
- [6] PPLive, "http://www.pplive.com/".
- [7] B. Cohen, "Bittorrent: http://bitconjurer.com."
- [8] V. Venkataraman and P. Francis., "Chunkyspread: Multi-tree unstructured end system multicast," in *IPTPS 2006*, San Babara, CA, USA, Feb. 2006.
- [9] V. Venkataraman and P. Francis, "On heterogeneous overlay construction and random node selection in unstructured p2p networks," in *IEEE INFOCOM 2006*, Barcelona, Spain, Apr. 2006.
- [10] J. Jiang and K. Nahrstedt, "Randpeer: Membership management for qos sensitive peer-to-peer applications," in *IEEE INFOCOM 2006*, Barcelona, Spain, Apr. 2006.
- [11] D. Kostic and et al, "Maintaining high bandwidth under dynamic network conditions," in *USENIX Annual Technical Conference*, 2005.
- [12] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall.
- [13] A. Goldberg, "Andrew goldberg's network optimization library." [Online]. Available: <http://www.avglab.com/andrew/soft.html>
- [14] S. Banerjee and B. Bhattacharjee, "A comparative study of application layer multicast protocols." [Online]. Available: <http://www.cs.wisc.edu/~suman/pubs.html>