Minimizing Streaming Delay in Homogeneous Peer-to-Peer Networks

Joe Wenjie Jiang*, Shaoquan Zhang†, Minghua Chen†, Mung Chiang*
*Princeton University, NJ, USA. {wenjiej, chiangm}@princeton.edu
†The Chinese University of Hong Kong, Hong Kong. {zsq008, minghua}@ie.cuhk.edu.hk

***Abstract*—Two questions on the theory of content distribution capacity are addressed in this paper: What is the worst user delay performance bound in a chunk-based P2P streaming systems under peer fanout degree constraint? Can we achieve both the minimum delay and the maximum streaming rate simultaneously? In the homogeneous user scenario, we propose a tree-based algorithm called *Inverse Waterfilling*, which schedules the chunk transmission following an optimal transmitting structure, under fanout degree bound. We show that the algorithm guarantees the delay bound for each chunk of the stream and maintains the maximum streaming rate at the same time.**

I. Introduction

Peer-to-peer (P2P) technology, already widely used for file-sharing applications, has the potential to reduce server and network load for video streaming applications, by allowing consumers to download live video content from each other [1]. Existing P2P streaming applications, however, suffer from low-quality video, periodic hiccups, and high delay [2]. This makes it difficult for service providers to leverage the technology directly in commercial offerings.

There has been a number of work on theoretical foundations to understand the design of P2P systems. Many of these work focus on maximizing the *streaming rate*, a metric that determines the video quality peers experience, subject to the peer capacities [3], [4], [5]. Recently there are also work on maximizing the streaming rate with bounded *node degree* and *node out-degree* taken into account [6], [7], [5]. Node degree is defined as the total number of neighbors of a peer. Similarly, node out-degree is defined as the total number of neighbors of a peer to whom the peer maintains out-going connections. Since a peer is typically a home PC with limited system resources, it is necessary to bound the number of connections (or out-going connections) a node maintains, thus bounding its node degree (or out-degree) [6].

Compared to rate, delay performance of P2P systems is much less understood. Earlier work on delay focused on fluid-based P2P content distribution model [7], and derived algorithms to minimize the worst tree depth. Others [8], [9] studied the average user delay under various settings.

In a chunk-based P2P streaming where a stream is modeled as an infinite sequence of video chunks, Liu [10] [11] investigated the fundamental limits of the minimum delay in realtime P2P streaming and proposed a simple snow-ball algorithm to approach the delay bound, though no degree bound is guaranteed. A graph labeling algorithm is proposed in [12] to minimize the initial buffering delay, which is a generalized version of the problem studied in [10]. G. Bianchi [13] extended the result of [10] by adding the restrictions on the number of neighbors each peer can have. But their work only studied the simplest case with one or two chunks and cannot extend to an infinite stream. So far no algorithms are known to achieve the bound with bounded degree constraint [14].

In this paper, we study the delay performance for homogenous chunk-based P2P streaming systems, where all peers have the same upload capacity. We also take practical considerations into account and bound the node out-degree across the system. We seek to answer the following fundamental questions:

- how to achieve the minimum worst user delay for continuous stream under node out-degree bound?
- can we achieve the maximum streaming rate and minimum worst user delay simultaneously?

Our results characterize the minimum worst user delay for an *infinite* continuous stream under arbitrary node out-degree bound, which can be extended to the same node in-degree bound. We further show it is possible to achieve both the maximum streaming rate and minimum worst user delay by packing a *finite* number of multicast trees, by our proposed Inverse Water-Filling algorithm. We also show that the marginal return of allowing large peer out-degree is diminishing, and a proper range of out-degree gives close-to-optimal delay.

II. System Model and Problem Statement

We first present a mathematical model for the peer-to-peer streaming systems and the problem to solve.

Consider a peer-to-peer streaming network with one source and N participating peers. The peer set is denoted by $\mathcal{N} = \{0, 1, \ldots, N-1\}$. The source is denoted by S. It generates a continuous stream of video chunks at a constant rate, and delivers the chunks to all the peers. A video chunk is the smallest unit of data exchange in the network. Let B be the chunk size. Let C_v denote the upload capacity for node v, which we assume to be the only bottleneck in the network. In this paper, we consider homogeneous P2P networks, where all peers including the source have the same upload capacity, *i.e.*, $C_v = C$ for all v in $\mathcal{N}$. Without loss of generality, we assume each peer has one unit capacity, *i.e.*, it takes one unit

This work has been in part supported by NSF CNS-0519880 and CNS-0720570 at the Princeton EDGE Lab. This work is also supported by Hong Kong RGC GRF grants 411008 and 411209, a Hong Kong UGC AoE grant (Project No. AoE/E-02/08), and two gift grants from Microsoft and Cisco.

of time for any peer to transmit one single chunk, and the source generates a new chunk every time unit.

When a peer transmits a chunk to one of its neighbor peers, the total delay is the sum of the transmission time of that chunk, i.e., B/C, plus the propagation delay. For P2P networks where the peer upload capacity is several hundred kbps, when peers are close to each other, transmission time is order of magnitude larger than the propagation delay. Based on this observation, we assume the propagation delay is zero in our study.

We define peer out-degree to be the total number of neighbors a peer can deliver chunks to. It has been argued that the peer out-degree should be bounded in order to reduce the overhead in maintaining active connections among peers [7]. In this paper, we consider the case where the out-degree of all peers are uniformly bounded by M, which is called the peer out-degree bound.

For a particular user and a particular chunk in the P2P streaming system, the chunk *playback delay* is defined as the latency between the chunk generation time at the source and the receiving time at the user. Further the *max-user-delay* for a particular chunk is defined as the maximum chunk playback delay among all users.

A fundamental question that asks the performance bound of a P2P streaming system is, *given a streaming rate, what is the minimum max-user-delay for each chunk and how to achieve that?*

III. Delay Optimization

In this section, we propose an optimal multi-tree construction algorithm in homogeneous P2P streaming networks, where peers are constrained by an out-degree bound M. We show that our algorithm achieves rate and delay optimality simultaneously.

A. Principle of Accelerating Chunk Propagation

We start with a simple case in which we only need to deliver one *single* chunk to all peers, which is also studied in [13]. The principle of single chunk transmission later serves as a building block when it comes to the case of continuous stream.

To ensure that a chunk quickly spans the entire network, one intuitive observation is that once a peer receives a chunk, it replicates the chunk to M peers which haven't received the chunk. Take the case $M = 4, N = 18$ for example, one way to span the chunk over the whole network is as shown in Figure 1. Suppose at time 0, the source generates a chunk. We consider the case where the source transmits the chunk to one peer, and does not participate in chunk replication afterwards (because it needs to transmit new chunks, as will be discussed later). After peer 0 gets the chunk from the source, it replicates the chunk to its 4 neighbor peers. Similarly, once other peers receive the chunk, they will transmit it to their neighbor peers. The replication process continues until the chunk spans the entire system. It has been shown in [13] that the number of new peers that receive the chunk at time t follows a M-step fibonacci sequence as follows:

$$F_M(t) = \begin{cases} 0 & t \le 0 \\ 1 & t = 1 \\ \sum_{k=1}^{M} F_M(t-k) & t \ge 2 \end{cases} \qquad (1)$$

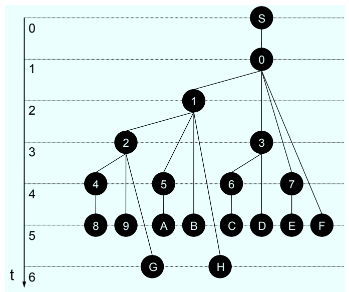


Fig. 1. Single chunk transmission for M=4, N=18.

It turns out that the scheme described in Figure 1 achieves the minimum max-user-delay. The following theorem gives a way to characterize the minimum max-user-delay for single chunk, denoted by D^*.

Theorem 1. *[13] In a chunk-based P2P streaming system where all peers have unit upload capacity and out-degree bound M, the time for a single chunk to span the entire system is lower-bounded by*

$$D^* = \min\{t : S_M(t) \ge N\} \qquad (2)$$

where

$$S_M(t) = \begin{cases} 0 & t \le 0 \\ \sum_{j=1}^{t} F_M(j) & i \ge 1 \end{cases} \qquad (3)$$

B. Optimal Delay and Out-Degree

From Theorem 1, optimal max-user-delay D^* is a function of peer number N and out-degree bound M. Knowing how N and M impact the D^* can not only give us deeper insights into the delay optimization problem but also guide the design of real P2P applications. Let's first derive the relationship. The exact expression of $S_M(t)$ is as follows [13]:

$$S_M(t) = \sum_{i=1}^{M} \frac{x_i}{(x_i - 1)Q_M(x_i)} \cdot x_i^t - \frac{1}{M-1}, \qquad (4)$$

where $x_i, i \in (1, M)$ are the M roots of the following equation:

$$x^M - x^{M-1} - x^{M-2} - \cdots - x - 1 = 0, \qquad (5)$$

and $Q_M(x)$ is as follows:

$$Q_M(x) = -1 + 2(M-1)x + \sum_{i=2}^{M-1} (M-i-2)x^i. \qquad (6)$$

According to Theorem 1, we plot the D^* under different N and M in Figure 2.

One observation from Figure 2 is as follows. Large M can help in reducing the delay of the chunk. But the impact of M on D^* is marginal when $M > 7$. On the other hand large M introduces communication overhead. So a proper node out-degree should be less than 8.

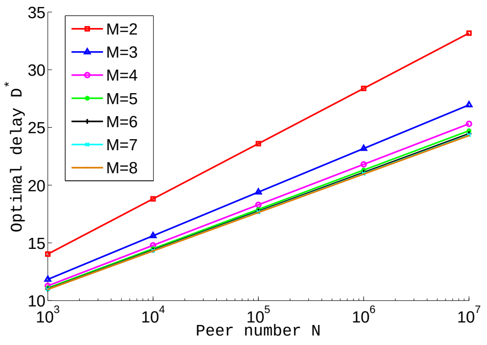


Fig. 2. Optimal delay D^* vs peer number N and out-degree bound M.

To understand it better, we approximate the optimal delay D^*. One important property about x_i in the expression of $S_M(t)$ is that only one root has module larger than 1. This root takes real value and tends to be 2 very quickly when M becomes large. Let's denote this largest-module root by x. We plot x with out-degree bound M in Figure 3.

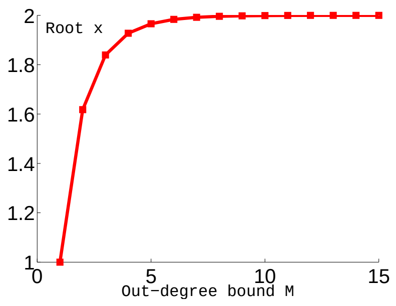


Fig. 3. The root x vs out-degree bound M.

When $M > 7$, the impact on the value of the root is negligible. Ignoring the influence of roots with module smaller than 1 when N is large and $M > 7$, we can approximate $S_M(t)$ as follows:

$$S_M(t) \approx \frac{2^{t+1}}{3} - \frac{1}{M-1}. \tag{7}$$

According to the definition of D^*:

$$D^* = \log_2\left(N + \frac{1}{M}\right) + \log_2 1.5. \tag{8}$$

From (8), we see that the impact of M on D^* is negligible for large N. The intuition is: the extra number of peers receiving a chunk each time unit due to a larger M is very small compared to the total number of peers N. Therefore, the node out-degree constraint should be less than 8 when N is large.

C. Achieving Optimal Delay for Continuous Stream

For live streaming, chunks are generated at a constant rate [1]. Under our problem setting the maximum streaming rate is one chunk per time unit [4]. Given any out-degree bound M, we have discussed how to achieve the minimum max-user-delay, for one single chunk in Section III-A. It remains a question whether the optimal delay D^* is attainable while at the same time achieving the maximum streaming rate.

The following observation interprets the challenge in achieving minimum delay for multiple chunks. Consider the tree shown in Figure 1. Source generates a chunk at time $t = 0$ and uses the tree shown in Figure 1 to distribute the chunk. Now suppose a new chunk is generated at time $t = 1$. Following the old tree for the new chunk is not possible: the upload capacity of non-leaf peers will be reserved for one or multiple consecutive time slots, preventing them from being reused in the same manner. While one can consider switching leaf nodes and internal nodes, the out-degree constraint also presents challenge: requiring the total out-degree across multiple trees be no greater than M. As a whole, the algorithm should schedule the chunk transmission in a way so that each peer transmits one chunk every time unit and the out-degree bound is obeyed.

We next present an algorithm that constructs a multi-tree, *i.e.*, a set of multicast trees, that achieves the minimum delay D^* for *every* chunk, without violating the out-degree constraint M across all trees. The optimality of our algorithm consists of three building blocks. We first show it is a necessary condition that every tree follows an identical structure. The question of the tree structure is answered in Lemma 1. We second show the minimum number of trees needed on the multi-tree, which is answered in Lemma 2. We finally propose the Inverse Waterfilling (IWF) algorithm to construct the multi-tree, following the tree structure and the number of trees stated in the previous two results. Combining three steps give us an algorithm that achieves both rate and delay optimality.

	chunk 4k+1	chunk 4k+2	chunk 4k+3	chunk 4k+4
t=0	S			
t=1	0	S		
t=2	01	8	S	
t=3	0123	89	C	S
t=4	01234567	89AB	CD	F
t=5	12	89AB3456	CD7E	FG
t=6		23	CD7E4569	FGAB
t=7			45	FGAB67DE
t=8				EG

Fig. 4. Multi-tree topology (active sets) for $M = 4, N = 18, D^* = 6$.

1) Multi-tree Structure: We first use an example to illustrate the multi-tree structure. We also start to introduce notations that simplify the tree presentation like in Figure 1. Take the case of $M = 4, N = 18$ for example. We show that we only need to construct $M = 4$ trees, which are used for chunk $4k+1, 4k+2, 4k+3, 4k+4$ where $k = 0, 1, \ldots, \infty$, respectively. Let $A_d^{(l)}$ denote the set of active peers, *i.e.*, those that are transmitting, on d-th layer of l-th tree, where $1 \le l \le M, 0 \le d \le D^* - 1$. Similarly, let $R_d^{(l)}$ denote the set of receivers which receive chunks from $A_d^{(l)}$. Apparently $|A_d^{(l)}| = |R_d^{(l)}|$. For example, in Figure 1, $A_3^{(1)} = \{0,1,2,3\}, R_3^{(1)} = \{4,5,6,7\}$.

[1]In practice, streaming rate depends on codec and video content, and hence, is not strictly constant.

Lemma 1. *In a delay-optimal multi-tree,*

$$|A_d^{(l)}| = \begin{cases} F_M(d+1) & d \le D^* - 2 \\ N - S_M(D^*-2) & d = D^* - 1 \end{cases} \tag{9}$$

Proof: To achieve minimum delay, the size of each active set, equals the size of its corresponding receiver set, and hence follows the M-step Fibonacci number. The active set at layer $D^* - 1$, is the number of remainder peers. ∎

Receiver sets can be derived from active sets as follows:

$$R_d^{(l)} = \begin{cases} A_{d+1}^{(l)} \backslash A_d^{(l)} & 0 \le d \le D^* - 3 \\ X, A_{d+1}^{(l)} \backslash A_d^{(l)} \subset X \subset \{n : n \in \mathscr{N} - A_d^{(l)}\} & d = D^* - 2 \\ \mathscr{N} \backslash \bigcup_{d=0}^{D^*-2} R_d^{(l)} & d = D^* - 1 \end{cases} \tag{10}$$

As shown in Equation 10, a receiver set consists of *new* peers that appear in the active set of the next time slot, except for the last two layers.

Tree structures are characterized by active sets[2]. For example, consider the tree shown in Figure 1 ($M = 4, N = 18$). Figure 4 is one possible configuration of active sets. Row t denotes the start of time slot t. The source generates a new chunk every unit time. Column $l, (l = 1, \ldots, M)$ denotes the multicast tree for chunk l. Entry (t, l) in the table denotes the active set for tree l at time t, *i.e.*, $A_{t-l+1}^{(l)}$. The size of active set at each layer follows the Fibonacci sequence, so every chunk achieves the minimum delay. Observe the following two invariants for this example, which we later show are also key conditions to construct an optimal multi-tree topology:

- Each peer appears no more than M times on the table, *i.e.*, the out-degree constraint is satisfied.
- All active sets at the same time are disjoint, *i.e.*, no peer is transmitting more than one chunk simultaneously.

It is not difficult to see that if the two conditions hold for all M trees, we can repeatedly use them for infinitely many continuous chunks, thus achieving minimum streaming delay.

2) Number of Multi-tree: Before presenting the tree construction algorithm, we need to determine how many different trees are needed.

Lemma 2. *In a chunk-based P2P streaming system where all peers have unit upload capacity and out-degree constraint M, the number of different trees necessary to achieve minimum delay is:*

$$\delta(N) = \begin{cases} M & N \ge 2^M \\ \lceil \log N \rceil & 1 < N < 2^M \\ 1 & N = 1 \end{cases} \tag{11}$$

Proof: (i) When $N \ge 2^M$, in order to achieve minimum delay, a peer's maximum out-degree required is M, as shown by Theorem 1. The streaming rate (*e.g.*, one unit) requires all but one peer contribute their upload capacities. Let every peer exhausts out-degree M, the total out-degree across all trees is $(N - 1 + 1) \cdot M$. Each tree has a total in-degree of N, because there are N receivers. Thus, the number of trees needed, is $N \cdot M / N = M$. (ii) When $1 < N < 2^M$, in order to achieve minimum delay, a peer only needs an out-degree of $\lceil \log N \rceil < M$. Follow the same calculation as (i), the number of trees needed is $\lceil \log N \rceil$. (iii) When N=1, constructing the single tree is trivial. ∎

[2]Note that there are many ways to construct the exact parent-child connections given a specific configuration of active sets and receiver sets, all of which satisfy the delay and out-degree requirements.

3) Inverse Waterfilling (IWF) Algorithm: We next present an algorithm called Inverse Waterfilling (IWF) to construct a multi-tree that achieves the minimum delay.

Lemma 1 characterizes the size of each active set, and it remains to fill in the active sets with qualified nodes and build the multi-tree using the active sets. The basic idea of the IWF algorithm is to iteratively select qualified peers to fill in each position. For example, Figure 4 shows one possible node placement.

To initialize, each node i maintains a degree budget $B_i = \min(M, \lceil \log N \rceil)$, which is the maximum out-degree used on the multi-tree. Once a node is placed on a position, *i.e.*, a particular layer of a tree, it consumes one degree and its budget is decremented by one. By Theorem 1, we know that once a node is first time placed at a position, it is used for one or multiple subsequent time slots. Let $d_a^{(l)}$ denote the total degree consumption for a position $a \in A_d^{(l)}$,*i.e.*, the number of time slots reserved for the node on tree l. We can compute $d_a^{(l)}$ by Theorem 1. Take the example of Figure 4, for the position $a \in A_2^{(1)}$ that node 1 takes, $d_a^{(1)} = 3$. Note that node 1 also appears in the last layer, but it is not mandotary by Theorem 1: we can use node 7 instead of node 1. Hence, $d_a^{(l)} = 3$ does not count the degree consumption at the last layer. All positions at the last layer have degree consumption of 1.

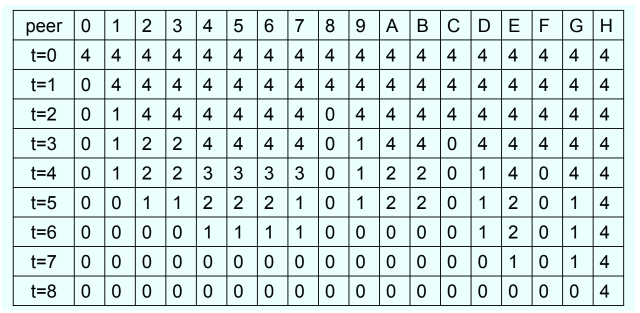
The algorithm starts at time $t = 0$. At a particular time t, we fill in the active sets $A_d^{(l)}$, for $l = 1, \ldots, \delta(N)$ respectively. Once a node is selected for a position, it is also selected for the corresponding positions in the following active sets. The node selection should qualify the following rules: (i) the node budget is greater than or equal to the degree consumption for that position, (ii) the node has not been selected on the same layer so active sets are disjoint, (iii) a node with lower budget is prioritized, breaking ties arbitrarily, *e.g.*, by node ID. Condition (i) is required so that the tree achieves minimum delay, (ii) avoids scheduling conflict since each peer only transmits one chunk at a time, and (iii) is key to the correctness of the algorithm, which also inspires the name of the algorithm. We formally present the algorithm in Table I.

Figure 5 shows the degree budget B_i at each step of the IWF algorithm. Note that if we break ties by node ID, then the last node is never used. Therefore, the total number of degrees spent, including the source, is $(N - 1 + 1)\delta(N)$, equals the total out-degree of the multi-tree. The correctness of the algorithm is shown by the following theorem.

Theorem 2. *In a homogeneous P2P network where each node has unit upload capacity and out-degree constraint M, the multi-tree built by the Inverse Water-Filling (IWF) algorithm achieves minimum max-user-delay D^* for every chunk. At the*

```
Inverse Water-Filling (IWF)
Initialize $B_i = \min(M, \lceil \log N \rceil)$, for $i = 1, \ldots, N$.
for $t = 0$ to $D^* + \delta(N) - 2$ do
    for $l = 1$ to $\delta(N)$ do
        for every $a \in A^{(l)}_{t-l+1}$
            if $a$ is not filled
                For position $a$, select node $i$ according to rules (i)-(iii)
                Mark all $d^{(l)}_a$ positions filled with node $i$
                $B_i \leftarrow B_i - d^{(l)}_a$
            end if
        end for
end for
```

TABLE I
INVERSE WATERFILLING



peer	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	G	H
t=0	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
t=1	0	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
t=2	0	1	4	4	4	4	4	4	0	4	4	4	4	4	4	4	4	4
t=3	0	1	2	2	4	4	4	4	0	1	4	4	0	4	4	4	4	4
t=4	0	1	2	2	3	3	3	3	0	1	2	2	0	1	4	0	4	4
t=5	0	0	1	1	2	2	2	1	0	1	2	2	0	1	2	0	1	4
t=6	0	0	0	0	1	1	1	1	0	0	0	0	0	1	2	0	1	4
t=7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	4
t=8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4

Fig. 5. Degree budget B_i in constructing multi-tree for $M = 4, N = 18$.

same time, the maximum streaming rate is also achieved.

Proof: We first show that the IWF algorithm achieves optimal delay, and then prove that optimal streaming rate is also achieved simultaneously.

The proof of delay optimality proceeds in three steps: (A) we show that the IWF algorithm always terminates successfully, (B) minimum delay is achieved for every chunk on the multi-tree, (C) minimum delay is achieved for infinitely many continuous chunks.

(A). To show that the IWF algorithm always terminates, it suffices to prove that we can always find a peer that qualifies rules (i)-(iii). We briefly outline the proof sketch here, and the full version can be found at [15]. Observe the following invariant: at any step of the algorithm, the total degree consumption for unfilled positions on the multi-tree, equals the total degree budget except the last peer. Now suppose we try to fill a position $a \in A^{(l)}_d$ but cannot find any qualified peers, there are only two cases: (a) For $\forall i \in \mathcal{N}$ that has not been selected in active sets on the same layer, $0 < B_i < d^{(l)}_a$, (b) otherwise, $B_i = 0$. We prove by contradiction that both cases do not exist. For case (a), suppose there exists an unselected peer i such that $B_i = k$. By assumption $k < d^{(l)}_a$, we can find a position a' on the same layer that has already been filled, and its degree consumption $d^{l'}_{a'} = k$ (such that $l' < l$). Let i' be the selected node that fills position a'. Claim that $B_{i'} = 0$, otherwise we should use i to fill position a', according to rule (iii). In other words, for *all* nodes that fill positions with degree consumption k, their budgets are zero. We show in [15] that it violates the invariant we stated earlier, by observing the the number of zero-budget nodes. For case (b), the total degree consumption for unfilled positions on the layer, where each node has degree consumption $\geq d^{(l)}_a$, exceeds the total leftover degree budgets because $m < d^{(l)}_a$. Again, this contradicts the invariant we stated earlier.
(B). This claim is Lemma 1.
(C). For infinitely continuous stream, we can reuse the multi-tree repeatedly. To show that such scheduling works for infinitely many chunks, it suffices to show that any active set is disjoint with active sets that are $\delta(N)$ time slots earlier. To see this, note that a node appears in active sets for $\delta(N)$ consecutive time slots on the multi-tree. Therefore, there are no nodes that also appear in an active set that is $\delta(N)$ time slots later on the same multi-tree. Finally, combining (A)-(C) together completes the proof of delay optimality.

For the rate optimality, the maximum streaming rate for a homogeneous system is $\min(1, \frac{N+1}{N}) = 1$. [7]. In our problem, the source generates a new chunk every time unit, which realizes the upper-bound of streaming rate. ■

Thus we show that delay and rate optimality can be achieved simultaneously in the region of homogeneity. A next step is to address the two questions we post in the introduction in a more general system with heterogeneous node upload capacities.

REFERENCES

[1] C. Huang, J. Li, and K. W. Ross, "Can Internet Video-on-Demand be Profitable?," in *Proc. ACM SIGCOMM*, 2007.
[2] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross, "A Measurement Study of a Large-Scale P2P IPTV System," *IEEE Transactions on Multimedia*, 2007.
[3] R. Kumar, Y. Liu, and K. Ross, "Stochastic fluid theory for P2P streaming systems," in *Proc. IEEE INFOCOM*, 2007.
[4] P. A. C. J. Li and C. Zhang, "Mutualcast: an effcient mechanism for content distribution in a p2p network," in *Acm Sigcomm Asia Workshop*, 2005.
[5] S. Liu, M. Chen, S. Sengupta, M. Chiang, J. Li, and P. A. Chou, "P2p streaming capacity under node degree bound," in *Proc. International Conference on Distributed Computing Systems*, 2010.
[6] M. Chen, M. Chiang, P. A. Chou, J. Li, S. Liu, and S. Sengupta, "Peer-to-peer streaming capacity: Survey and recent results," in *the 47th Annual Allerton Conference on Communication*, 2009.
[7] S. Liu, R. Zhang-Shen, W. Jiang, J. Rexford, and M. Chiang, "Performance bounds for peer-assisted live streaming," in *Proc. ACM SIGMETRICS*, 2008.
[8] F. Huang, B. Ravindran, and A. Vullikanti, "An approximation algorithm for minimum-delay peer-to-peer streaming," in *Ninth International Conference on Peer-to-Peer Computing*, 2009.
[9] Y. Wu, Y. C. Hu, J. Li, and P. A. Chou, "The delay region for p2p file transfer," in *2009 IEEE International Symposium on Information Theory*, 2009.
[10] Y. Liu, "On the minimum delay peer-to-peer video streaming: How realtime can it be?," in *ACM Multimedia*, pp. 127–136, 2007.
[11] Y. Liu, "Delay bounds of chunk-based peer-to-peer video streaming," *IEEE/ACM Transactions on Networking*, 2009.
[12] C. Feng, B. Li, and B. Li, "Understanding the performance gap between pull-based mesh streaming protocols and fundamental limits," in *Proc. IEEE INFOCOM*, 2009.
[13] G. Bianchi, N. Blefari-Melazzi, L. Bracciale, F. L. Piccolo, and S. Salsano, "Fundamental delay bounds in peer-to-peer chunk-based real-time streaming systems," *CoRR*, vol. abs/0902.1394, 2009.
[14] Y. Wu, Y. C. Hu, J. Li, and P. A. Chou, "The delay region for P2P file transfer," in *Proc. IEEE International Symposium on Information Theory*, 2009.
[15] W. Jiang, S. Zhang, M. Chen, and M. Chiang, "Minimizing streaming delay in homogeneous peer-to-peer networks." Princeton University Technical Report, 2010.