On Minimizing Average End-to-End Delay in P2P Live Streaming Systems

Fei Huang[1], Maleq Khan[2], and Binoy Ravindran[1]

[1] Dept. of Electrical & Computer Engineering
[2] Network Dynamics and Simulation Science Laboratory
Virginia Tech, Blacksburg, VA 24061, USA
huangf@vt.edu,maleq@vbi.vt.edu,binoy@vt.edu

Abstract. In this paper, we devise a streaming scheme which optimizes the bandwidth allocation to achieve the minimum average end-to-end P2P streaming delay. We first develop a generic analytical framework to model the minimum average delay P2P streaming problem, called the MADPS problem. We then present *iStream* to solve the MADPS problem. The core part of iStream is a fast approximation algorithm, called *iStream-APX*, based on primal-dual schema. We prove that the performance of iStream-APX is bounded by a ratio of $1+\omega$, where ω is an adjustable input parameter. Furthermore, we show that the flexibility of ω provides a trade-off between the approximation factor and the running time of iStream.

1 Introduction

In the recent decade, P2P live media streaming applications have exhibited growing popularity, such as IPTV, VOIP, and video conferencing. By enabling efficient cooperation among end-users, P2P live streaming can distribute thousands of channels to millions of viewers simultaneously [1]. In these classes of applications, the delivery of real-time video content imposes rigorous constraints on the end-to-end delay. Obtaining assurances on meeting such delay constraints for multiple channels is a challenging problem, especially in highly dynamic and heterogeneous P2P network environments. The long playback latency has negatively affected the extensive commercial deployment of P2P systems. For example, IPTV deployment from commercial service providers is far below the industry expectation [2]. Motivated by these, in this paper, we focus on minimizing average end-to-end streaming delay in P2P networks.

Recently, layered coding has emerged as a viable solution for delivering real-time streaming content [3]. This technique not only provides an adaptive support for different downloading capacities on peers, but also allows IPTV service providers to deliver live content at diverse video definitions from the same coding process. For example, viewers may pay general fees for a standard service, or extra fees for 1080HD video or even 3D video. Unlike traditional IPTV service where viewers only download the multimedia content, under the P2P paradigm, substantial bandwidth may exist in viewers who pays only for a standard service, while HD viewers may instead suffer bad streaming service due to the bandwidth deficit among them. To maximize the bandwidth utilization, we should enable peer cooperation among viewers of different service qualities. Toward that, the HD content can be forwarded through peers with standard service, but only the HD viewers receive the authorization key for viewing HD content. This raises a fundamental question: how to optimally distribute the video content and conduct sub-stream scheduling among peers with diverse service qualities, while achieving the minimum average end-to-end P2P streaming (or MADPS) delay. We call this problem, the MADPS problem.

Minimizing streaming delays for P2P live systems is not a trivial problem. This is due to the heterogeneous bandwidth requirements and network dynamics of P2P systems. Previous theoretical works on designing P2P live streaming usually assume a homogeneous service quality [4, 5]. Thus, obtaining optimal solutions to this problem for large-scale networks is expensive in terms of algorithmic computational costs [6]. Approximate or heuristic solutions with scalable costs are therefore highly desirable. In this paper, we focus on approximate algorithms because we target time-critical P2P applications (e.g., video conferencing, or cloud computing), for which assured bounds on end-to-end delays are more desirable than heuristic (or empirically-established) gains in end-to-end delays. In addition, the analytical foundation that is necessary for developing approximate algorithms can contribute to a greater understanding of the problem and can provide deeper insights on designing efficient algorithms, be they approximate or heuristic. We take the first such steps toward this. The paper is theory-oriented.

For a feasible solution, we start with the assumption of a static network—i.e., no churn. In this way, we can devise a framework which is analytically achievable. The method will be most suitable for the scenario where a service provider deploys a set-top box at viewers' homes. In that case, even when a viewer turns off the TV, the set-top box can still contribute its bandwidth to other viewers. For this scenario, we first develop an analytical model that formulates the MADPS problem as an optimization problem. Then we propose an algorithm called *iStream* to solve MADPS problem. Inspired by the primal-dual schema, we develop an approximation algorithm as the core of iStream, called *iStream-APX* for optimally utilizing the bandwidth among peers subscribing to different video qualities, while achieving the minimum average streaming delay. We show that iStream-APX's performance in terms of delay is bounded by a factor of $1 + \omega$, where ω is an input parameter. iStream's running time is also bounded. We show that there exists a trade-off between iStream-APX's approximation factor ω and its running time. The approximation factor is adjustable in the range of $(1, n]$, where n is the number of peers in the network. This trade-off allows users to flexibly tune the performance bound according to running time requirements.

Thus, the paper's contribution is an approximation algorithm for the MADPS problem with bounded performance and running time (which can be traded-off, one for gains in the other), and its adaptive distributed version to operate in high-churn networks. iStream is the first approximation-based solution for the MADPS problem, and we are not aware of any other past efforts on approximating the MADPS problem.

The rest of the paper is organized as follows. Section 2 overviews past and related works. In Section 3, we describe our network model and formulate the MADPS problem. Section 4 presents our proposed approximation algorithm and derives its performance. Section 5 concludes the paper.

2 Related Work

Theoretical works on the minimum delay P2P streaming problem are limited, though recently a growing number of studies have focused on P2P live streaming [4–9]. Due to the lack of formal theoretical bounds, intuitions and heuristics have driven the design of P2P schemes so far [4, 7]. For example, Ren *et al.* [4] propose a heuristic to reduce the delay on mesh topology, where peers select their parents based on the metric of link capacity divided by communication delay. In this algorithm, peers located at the edge of mesh may only download the data without uploading, which may lead to low bandwidth utilization in P2P networks. Thus, when the total uploading capacity is close to the downloading capacity in the P2P community, some peers may not be able to receive a live streaming.

In our previous work [5], we developed an approximation algorithm to minimize the maximum P2P streaming delay by clustering and filtering methods with an approximation bound of $O(\sqrt{\log n})$. The minimum delay P2P streaming problem (or MDPS) presented in [5] focuses on minimizing the maximum end-to-end streaming delay. The MDPS problem is significantly different from the problem of minimizing the average end-to-end delay problem which we focus here. For example, the simulation results in [5] show that minimizing the maximum delay does not necessarily minimize the average end-to-end delay. Furthermore, the work in [5] assumes a network model with a symmetric graph and satisfying the triangle inequality. In contrast, in this paper, we remove those assumptions in modeling the minimum average delay P2P streaming problem.

The MADPS problem that we focus has some similarity with the minimum-cost multi-commodity flow problem (or MCMF) [10, 11]. iStream is inspired by the primal-dual schema from Garg and Konemann [10]. However, previous approximation solutions to the MCMF problem generally assume flow conservation on nodes—i.e., incoming commodities and outgoing commodities are exactly equal in amount. This is not true in P2P streaming, where peers can reproduce whatever commodities they receive—i.e., flow conservation does not hold. In addition, the MCMF problem considers only the capacities on edges, whereas in P2P streaming, the capacities actually exist on nodes instead of edges. This distinction (for the MADPS problem) further requires optimal flow scheduling among edges departing from the same node. All these differences make the MADPS problem more complex than the MCMF problem. Our work tackles these complexities and achieves a solution with near-optimal performance bound.

3 Problem Formulation

In this section, we formally state the minimum average end-to-end delay P2P streaming (MADPS) problem and present the problem in linear programming (LP) framework.

3.1 Preliminaries and Modeling

We model an overlay network as a directed graph $G = (V, E)$, where V is the set of vertices representing peer nodes, and E is the set of overlay edges representing directed overlay links. Let n represent the number of peers in the network, i.e. $n = |V|$. Each overlay link $(i, j) \in E$ is associated with a communication delay l_{ij}. In the rest of this paper, we define the length of edge (i, j) as l_{ij}, $\forall (i, j) \in E$. For every peer $i \in V$, we define an upload capacity of C_i units/second and a download capacity of I_i units/second. For ease of presentation, we define *unit* as the minimum flow size in P2P streaming, which may vary in different applications [12, 13].

We consider a peer-to-peer streaming session to originate from a single source node S to a set of receivers R, where $V = \{S\} \cup R$. Peers may receive the streaming data from the source node directly or indirectly from multiple P2P paths. In practical applications, receivers may pay for services of different streaming qualities, e.g., 720i/p and 1080i/p, which leads to different streaming rates correspondingly. Suppose peer j selects a service that has a constant streaming rate of d_j units/second. We denote f_{ij} as the rate at which peer i streams to peer j. If peer j receives the aggregated non-identical streams at d_j units/second from its parents, we call peer j as *fully served* [4]. Mathematically, the fully served requirement of peer j can be expressed as $\sum_{i: i \in L_j} f_{ij} = d_j$, where L_j is the set of parents of peer j. We assume that a fully served peer can smoothly play back the streaming content at its original rate of d_j units/second [4].

We call the stream from the source to one receiver j as the *P2P unicast flow* to j. Each P2P unicast flow U_j may consist of streams from multiple P2P paths, called *fractional flows* [6]. Each fractional flow $p \in U_j$ has the arrival latency $l(p)$ from the source to receiver, i.e., *end-to-end delay*, where $l(p) = \sum_{(i,j) \in p} l_{ij}$. We define the average end-to-end delay of the unicast flow U_j as the weighted average of end-to-end latencies of all its fractional flows, where the weight is the portion of fractional flow rate to the total streaming rate. Denote $f(p)$ as the streaming rate of fractional flow p. For viewer j, the weighted average of end-to-end latencies can be expressed by

$$\frac{1}{d_j} \sum_{p \in U_j} l(p) f(p).$$

To stream multimedia content to multiple receivers, we can envision multiple unicast flows from the source to receivers. Thus, the *average end-to-end delay in P2P streaming* is defined as the weighted average latency of all fractional flows to all receivers, which can be described by

$$\frac{1}{\sum_{j \in R} d_j} \sum_{p \in P} l(p) f(p), \tag{1}$$

where $P = \bigcup_{j \in R} U_j$. Since the term $\sum_{j \in R} d_j$ has no effect on the optimal solution, i.e., the solution that minimizes (1) also minimizes $\sum_{p \in P} l(p)f(p)$, we will focus on minimizing $\sum_{p \in P} l(p)f(p)$. It is easy see that removal of the term $\sum_{j \in R} d_j$ also preserves the approximation factor. For ease of presentation, we simply refer to $\sum_{p \in P} l(p)f(p)$ as *the cumulative delay* in the later sections.

To help understand the concept of average end-to-end delay, we use the term: "envision" in the above paragraph. In reality, there exists only one stream through each edge (i, j) instead of multiple fractional flows and peer j can reproduce any part of the stream content it receives and send it to other peers. Therefore, the actual data rate on an edge (i, j) is $\max_{t \in R} \sum_{p \in P_{ij}^t} f(p)$, where P_{ij}^t is the set of fractional flows through edge (i, j) to receiver t.

Next we provide a formal description of the problem.

3.2 MADPS Problem

Definition 1 ***Minimum Average End-to-End Delay P2P Streaming Problem (MADPS problem):*** *Given the capacity and data rate constraints that are mentioned in this section, the MADPS problem is to devise a streaming scheme which minimizes the maximum average end-to-end streaming delay with all receivers fully served.*

There is no known efficient algorithm with a practically-feasible running time to solve this problem optimally. Therefore, we are motivated to develop a near-optimal approximation algorithm with significantly smaller running time.

To ensure a solution exists to the MADPS problem, it is reasonable to assume the total bandwidth resources in P2P networks is sufficient to support the full services on all the viewers. Hence, we deduct the bandwidth requirement in Corollary 1.

Corollary 1 *If the instance of MADPS problem has a solution, then the sum of the upload capacities, including source and receivers, must be no less than the sum of fully served streaming rates at all receivers, i.e.,*

$$\sum_{i \in V} C_i \geq \sum_{j \in R} d_j. \tag{2}$$

In addition, we presume that the download capacity $I_i \geq d_i, \forall i \in V$ for a smooth playback at the receiver.

4 Approximation Algorithm

In this section we devise an approximation algorithm to find the near-optimal solution with provable bounds on the worst-case performance and running time.

4.1 Overview of Techniques

There are two fundamental techniques used in this work, including *primal-dual schema* and *binary search* based on the result of primal-dual schema.

First, we describe *primal-dual schema* [10, 14]. Given a linear programming problem, also referred to as a *primal problem*, we can convert it to a *dual problem*. Due to space limitation, we do not present the detailed mechanics of this conversion here, which can be found at [14]. Primal and dual problems are in a "mirror" relation. If one problem is a maximization problem, the other problem is a minimization problem, and vise versa. Suppose we have a primal problem: $\max \mathbf{c}^{\mathrm{T}}\mathbf{x}$, and the corresponding dual problem:$\min \mathbf{b}^{\mathrm{T}}\mathbf{y}$. According to the weak duality theorem, if $\mathbf{X}$ and $\mathbf{Y}$ are feasible solutions for the primal and dual problems respectively, it follows that $\mathbf{c}^{\mathrm{T}}\mathbf{X} \leq \mathbf{b}^{\mathrm{T}}\mathbf{Y}$. Moreover, the primal and dual problems share the same optimum, denoted by OPT. Given an approximation factor ρ, ρ bounds $\frac{\mathrm{OPT}}{\mathbf{c}^{\mathrm{T}}\mathbf{x}}$. Since any feasible solution to the dual also provides an upper bound on OPT, the approximation factor can be established by comparing the primal and dual solutions. In light of this, the primal-dual schema starts with a feasible solution for dual problem and relax the conditions for primal problem. Then, iStream iteratively improves the feasibility of primal conditions and the optimality of the dual solution. iStream winds up with feasible solutions for both primal and dual problems. So, the gap between them makes the approximation factor.

In detail, iStream employs the primal-dual schema to solve the delay-bounded maximum streaming rate problem (DBMSR problem) defined as follows.

Definition 2 ***Delay-bounded Maximum Streaming Rate problem (DBMSR problem):*** *Given a bound L on the average delay, i.e., $\sum_{p \in P} l(p) f(p) \leq L$, the DBMSR problem is to devise a streaming scheme which maximizes λ, where $\sum_{p \in P^t} f(p) \geq \lambda d_t, \forall t \in R$.*

In the next step, we can do a binary search on L to find the smallest λ that satisfies $\lambda \geq 1$. Towards that purpose, a reasonable initial value of L should be set in the range of $[\sum_{j \in R} d_j \cdot \min_{p \in P} l(p), \sum_{j \in R} d_j \cdot \max_{p \in P} l(p)]$. The result of this procedure leads to a near-optimal solution for MADPS problem.

In the rest of this section, we formulate the DBMSR problem by primal-dual schema. Then, we discuss the details of iStream and derive its performance bound.

4.2 Formulation about Primal and Dual

We refer to DBMSR problem as the primal problem here, or simply called primal. According to its definition, we formulate the primal as following.

Primal:

$$\max\ \lambda \tag{3}$$

subject to

$$\sum_{p \in P_{ij}^t} f(p) \leq \sum_{p \in P_{ij}^j} f(p), \qquad \forall (i,j) \in E, \forall t \in R \tag{4}$$

$$\sum_{j:(i,j)\in E} \sum_{p \in P_{ij}^j} f(p) \leq C_i, \qquad \forall i \in V \tag{5}$$

$$\sum_{p \in P^t} f(p) \geq \lambda d_t, \qquad \forall t \in R \tag{6}$$

$$\sum_{p \in P} l(p) f(p) \leq L, \tag{7}$$

$$f(p) \geq 0, \lambda \geq 0. \tag{8}$$

Equation (4) presents the fact that the amount of fractional flow through edge (i,j) to any viewer will always be bounded by the total fractional flow sent to node j, i.e. $\sum_{p \in P_{ij}^j} f(p) = \max_{t \in R} \sum_{p \in P_{ij}^t} f(p)$. Because we attempt to utilize the bandwidth from peers scribing to the standard video quality, it is possible to see the amount of fractional flow to j from all incoming edges of j exceeds viewer j's demand, i.e., $\sum_{i:(i,j)\in E} \sum_{p \in P_{ij}^j} f(p) \geq d_j$. Equation (5) ensures no conflicts in terms of the uploading capacities. In terms of the downloading capacities, which can be written as $\sum_{j:(j,i)\in E} \sum_{p \in P_{ji}^i} f(p) \leq I_i, \forall i \in V$, we assume $I_i \geq \max_{j \in R} d_j$, which is practical with the wide deployment of high-speed internet. Since the actual flow sent to or relayed by node i cannot be larger than the maximum service demand, expressed by $\max_{j \in R} d_j$, it is reasonable to remove the constraints on the downloading capacities in the LP expression without affecting the optimal solutions. Equation (6) means the objective of DBMSR problem is to maximize the minimum demand on nodes. Equation (7) puts a bound L on the cumulative delay. As stated in Section 4.1, we can conduct a binary search on L until λ is very close to 1 to achieve a solution to the MADPS problem.

Next, we convert the primal to its dual problem, or simply called dual.

Dual:

$$\min \sum_{i \in V} C_i w_i + \varphi L \tag{9}$$

subject to

$$\sum_t d_t z_t \geq 1, \qquad \forall t \in R \tag{10}$$

$$\sum_{(i,j)\in p, i \neq i'} s_{ij}^t + w_{i'} + \varphi l(p) \geq z_t, \qquad (i',t) \in p, \forall t \in R,$$

$$\forall p \in P^t \tag{11}$$

$$s_{ij}^t \geq 0, w_i \geq 0, z_t \geq 0, \varphi \geq 0, \tag{12}$$

where i' is the peer one hop away from the viewer t on routed path.

Generally, there is no direct physical meaning to the dual problem because it comes from a mechanical conversion of the primal problem. To help the analysis on iStream, we hereby assign a logical explanation to the dual after investigating its formulation. We envision each edge (i, j) has multiple copies $(i, j)^1, (i, j)^2, \cdots, (i, j)^{|R|}$, where any copy $(i, j)^t$ exclusively represents to the usage of edge (i, j) for flows to viewer t. Each edge $(i, j)^t$ is associated with a length metric s^t_{ij}, and each node i is associated with a length metric w_i. Thus, we view $\sum_{(i,j)\in p, i\neq i'} s^t_{ij} + w_{i'} + \varphi l(p)$ as the length function associated with flow path p, where φ is the weight associated with the delay metric $l(p)$. According to Equation (11), z_t can be comprehended as the shortest length to node t based on the length function.

4.3 Approximation Algorithm

iStream-APX is the core part of iStream, which is built with approximation algorithm. iStream-APX proceeds in phases. Each phase is completed by $|R|$ iterations with each iteration satisfy the demand of one viewer. Due to the constraints from LP conditions, each iteration may be completed by multiple steps. Inside each step, we route such amount of fractional flows that can ensure the constraints are not violated. At the end of all phases, iStream-APX will rescale all the flows to ensure a feasible solution to the primal. We express the k^{th} step in the t^{th} iteration of m^{th} phase by (m, t, k). The initial status is marked by $(0, 0, 0)$, or simply (0).

We start the algorithm with the following initial settings on length metrics.

$$w_i(0) = \delta/C_i, \qquad \forall i \in V \tag{13}$$

$$s^t_{ij}(0) = w_i, \qquad \forall (i, j) \in E, \forall t \in R \tag{14}$$

$$\varphi(0) = \delta/L, \tag{15}$$

where δ is an input parameter. The proper assignment of it will be discussed in Section 4.4.

Throughout the execution of algorithm iStream-APX, it dynamically updates the length metrics, which are used to built the flowing path. Let $w_i(m, t, k)$, $s^t_{ij}(m, t, k)$, $\varphi(m, t, k)$ be the length metrics at the end of step (m, t, k). At step (m, t, k), iStream-APX first computes the shortest path p^* from S to viewer t in terms of the length function $\sum_{(i,j)\in p, i\neq i'} s^t_{ij}(m, t, k-1) + w_{i'}(m, t, k-1) + \varphi(m, t, k-1)l(p)$, where $(i', t) \in p, p \in P^t$. Then, it finds the minimum capacity $C_{\min}$ on nodes along the shortest path, which can be expressed by $C_{\min} = \min_{i\in p^*}\{C_i\}$. Since the previous steps may already route some flows to the viewer, let γ_t be the residual amount of demands unsatisfied on node t, and $x(p) = \min\{\gamma_t, C_{\min}\}$. Next, we route $x(p)/\eta$ amount of flow to t, where $\eta = l(p)x(p)/L$ if $l(p)x(p) > L$; otherwise, $\eta = 1$. So the length bound L and the capacities on the path are not violated in each step. At the end of this step, we update the length metrics as well as the residual demands according to Equations (16)-(19).

$$w_i(m,t,k) = w_i(m,t,k-1) \cdot [1 + \epsilon \cdot f(m,t,k)/C_i] , \quad \forall i \in p^* \setminus \{t\} \tag{16}$$

$$s_{ij}^t(m,t,k) = w_i(m,t,k), \forall i \in p^* \setminus \{t\}, \forall (i,j) \in E, \forall t \in R \tag{17}$$

$$\varphi(m,t,k) = \varphi(m,t,k-1) \cdot \prod_{j \in p^* \cap R} [1 + \epsilon \cdot L_j(m,t,k)/L] , \tag{18}$$

$$\gamma_i(m,t,k) = \gamma_i(m,t,k-1) - f(m,t,k), \forall i \in p^* \setminus \{t\} \tag{19}$$

where $f(m,t,k)$ is the amount of flow routed in current procedure (m,t,k) and $L_j(m,t,k)$ means the cumulative delay of the routed flow through node j which is on the path p^* at step (m,t,k). Mathematically, it can be expressed by $L_j(m,t,k) = l(p_j^*)x(p_j^*)$, where p_j^* is the segmental path from S to j on path $p*$. We can observe in each step for every capacity-saturated node i on the routing path, all the length metrics regarding i increase by a factor of $1+\epsilon$. Since the assignments of s_{ij}^t are identical in Equation (17), we simply use s_i to represent all s_{ij}^t.

We repeat the steps until the demand of viewer t is fully satisfied. Then we call the end of iteration t, and start the iteration for next viewers which has positive residual demand in the current phase. After the last step of a phase, all viewers have no residual demands, i.e., $\gamma_t = 0, \forall t \in R$. Then, we start a new round of phase $m+1$ after resetting the residual demands equal to viewer's actual demands, i.e., $\gamma_t = d_t, \forall t \in R$. The whole procedure completes as soon as $W(m,t,k) \geq 1$. Obviously, the cumulative flows routed in all phases may strongly violate the capacity and average delay constraints. Define $F(p)$ as the cumulative flows routed in all phases through path p. To obtain a feasible solution to the primal problem, we need to scale down each $F(p)$ by a factor of $\log_{1+\epsilon} 1/\delta$. We will justify the correctness of this scaling down factor in Section 4.4.

We continue a binary search on L by repeating iStream-APX until λ tends to 1, denoted as $\lambda \to 1$. The result of the binary search will provide a near-optimal solution to MADPS problem. The detailed procedures about the approximation algorithm are presented in [15].

4.4 Algorithm Analysis

In this section, we formally analyze the algorithm and prove the approximation factor. To facilitate the analysis, we make some definitions. Let $W = \sum_{i \in V} C_i w_i + \varphi L$ be the metric minimized by the dual. Let ζ_t be the shortest length from S to t, i.e.,

$$\zeta_t = \min_{p \in P^t} \sum_{(i,j) \in p, i \neq i'} s_{ij}^t + w_{i'} + \varphi l(p). \tag{20}$$

Here ζ_t actually represents and interprets the meaning of z_t. Besides, we define

$$\alpha = \sum_t \Big(d_t \zeta_t \Big). \tag{21}$$

Lemma 1 *Denote the optimal solution to the dual by $OPT(W)$. When $OPT(W)$ is obtained, α is 1.*

Proof. We prove this lemma by contradiction. As we know, α represents $\sum_t d_t z_t$ in the dual. Let $W = W'$ when $\alpha = 1$. For the sake of contradiction, we assume $W' > \mathrm{OPT}(W)$, where $\mathrm{OPT}(W)$ is achieved when $\alpha = \alpha^* > 1$. Then, we scale down α^* to 1. Towards that, we can divide all the s^t_{ij} and φ by a factor of $\sum_t d_t z_t$. As a result, w_i will proportionally scale down the same factor. Consequently, it leads to an update on W with a new value W', where $W' = \mathrm{OPT}(W)/\sum_t d_t z_t$. According to the assumption, W' should be larger than $\mathrm{OPT}(W)$. However, because $\sum_t d_t z_t > 1$, we have $W' = \mathrm{OPT}(W)/\sum_t d_t z_t < \mathrm{OPT}(W)$, which contradicts the assumption. Thus, the lemma follows.

Define β as the minimum value of W/α, i.e., $\beta = \min W/\alpha$. We conclude the following theorem.

Theorem 1 *The optimal solution to the dual, denoted as $OPT(W)$, is equivalently to the optimal solution β under the same constraints in the dual.*

Proof. From the definition of β,we know that $\beta = \min W/\alpha$. Suppose β is achieved when $\alpha = \alpha^* > 1$. We can always proportionally scale down all the s^t_{ij} and φ by multiplying a factor of $1/\alpha^*$. As a result, $\alpha = 1$. Since W will scale down with the same factor, W/α will keep the optimal value β. That is to say we can always find the optimal solution β with $\alpha = 1$.

According to Lemma 1, it follows that $\alpha = 1$ when $\mathrm{OPT}(W)$ is achieved. Therefore, we can conclude the problem of finding $\mathrm{OPT}(W)$ for the dual is equivalently to solving the optimization problem for W/α. This completes the proof.

In iStream-APX, we update the length metrics s_i, w_i, φ on the routing path. In terms of that, we can conclude the following.

Lemma 2 *w_i increases at least by a factor of $1 + \epsilon$ for every C_i units of flow through node $i, \forall i \in V$.*

Proof. Due to space limitation, we do not prove this lemma here. The detailed proof can be found at [15].

Corollary 2 *s_i increases at least by a factor of $1 + \epsilon$ for every C_i units of flow through node $i, \forall i \in V$.*

Corollary 3

$$\log_{1+\epsilon} \frac{\varphi(m)}{\varphi(0)} \geq \sum_{p \ inP} l(p) f(p)/L,$$

where $f(p)$ represents the cumulative amount of flows through path p at the end of phase m.

Given the assumption that the total bandwidth resources in P2P networks is sufficient to support the full services on all the viewers, we can do a binary search on L so as to find the smallest λ that satisfies $\lambda \geq 1$. According to the weak-duality theorem, it follows that $\beta \geq \lambda \geq 1$.

Lemma 3 *Given $\beta \geq 1$, we have*

$$\beta \leq \frac{\epsilon(M-1)}{(1-\epsilon)\ln\frac{1-\epsilon}{(|V|+1)\delta}}.$$

Proof. We start the proof by analyzing the change on W on each step. At the end of this analysis, we will carry out the cumulative increment on W when algorithms stops.

Let $p(m,t,k)$ be the shortest path found at procedure (m,t,k), and $f(m,t,k)$ be the quantity of flow routed through path $p(m,t,k)$. Because in our algorithm we assign $s_i = w_i$ for any procedure (m,t,k), we can simplify the length function as

$$\sum_{(i,j)\in p, i\neq i'} s_i^t + w_{i'} + \varphi l(p) = \sum_{(i,j)\in p} (w_i + \varphi l_{ij}), \tag{22}$$

where $(i',t) \in p$. Consequently, we can carry out the following.

Since the objective is to find the cumulative increment, we can think of the change on length metrics w_i and φ regarding node i at procedure (m,t,k), where $i \neq t$, will hold until procedure $(m,i,0)$ without loss on the final cumulative increment on W.

$$\begin{aligned}
&W(m,t,k) - W(m,t,k-1)\\
=&C_{i'}\cdot\Big(w_{i'}(m,t,k) - w_{i'}(m,t,k-1)\Big)+\\
&+\Big(\varphi(m,t,k) - \varphi(m,t,k-1)\Big)\cdot L\\
\leq&\sum_{i\in p(m,t,k)\setminus\{t\}}\Big(C_i\cdot w_i(m,t,k-1)\epsilon f(m,t,k)/C_i\Big)+\\
&+\Big(\varphi(m,t,k-1)\epsilon L(m,t,k)/L\Big)\cdot L\\
=&\epsilon\cdot\Big[\sum_{i\in p(m,t,k)\setminus\{t\}}\Big(w_i(m,t,k-1)f(m,t,k)\Big)+\\
&+\varphi(m,t,k-1)L(m,t,k)\Big].
\end{aligned}$$

Let K_{mt} be the number of steps in a given iteration t of phase m, $\zeta_t(m,t,k)$ be the shortest path at the end of procedure (m,t,k), and $l(m,t,k)$ be the cumulative latency on path $p(m,t,k)$. We have

$$
\begin{aligned}
&W(m,t+1,0)-W(m,t,0)\\
\leq&\epsilon\cdot\sum_{k=1}^{K_{mt}}\Big[\sum_{i\in p(m,t,k)\setminus\{t\}}\Big(w_i(m,t,k-1)f(m,t,k)\Big)+\\
&+\varphi(m,t,k-1)L(m,t,k)\Big]\\
=&\epsilon\cdot\sum_{k=1}^{K_{mt}}\Big[f(m,t,k)\cdot\sum_{i\in p(m,t,k)\setminus\{t\}}\Big(w_i(m,t,k-1)\Big)+\\
&+\varphi(m,t,k-1)l(m,t,k)\Big]\\
=&\epsilon\cdot\sum_{k=1}^{K_{mt}}\Big[f(m,t,k)\cdot\sum_{(i,j)\in p(m,t,k)}\Big(w_i(m,t,k-1)+\\
&+\varphi(m,t,k-1)l_{ij}\Big)\Big]\\
=&\epsilon\cdot\sum_{k=1}^{K_{mt}}f(m,t,k)\cdot\zeta_t(m,t,k-1)\\
\leq&\epsilon\cdot d_t\zeta_t(m,t,k).
\end{aligned}
$$

For brevity on notations, we define $W(m)$ as the value of W at the end of phase m, and make a similar definition for $\alpha(m)$. Then, it follows that

$$
\begin{aligned}
&W(m)-W(m-1)\\
=&W(m,|R|,K_{m|R|})-W(m,0,0)\\
\leq&\epsilon\cdot\sum_{t=1}^{|R|}\Big(d_t\zeta_t(m,t,K_{m|R|})\Big)\\
\leq&\epsilon\alpha(m).
\end{aligned}
\tag{23}
$$

Combining the property of $W(m)/\alpha(m)\geq\beta$ with Equation (23), we can carry out

$$W(m)\leq\frac{W(m-1)}{1-\epsilon/\beta}.$$

In light of the initial settings, $w_i(0)=\delta/C_i$ and $\varphi(0)=\delta/L$. Thus, we obtain $W(0)=(|V|+1)\delta$.

Given $m\geq 1$ and $\beta\geq 1$, it follows that

$$W(m)\leq\frac{(|V|+1)\delta}{(1-\epsilon/\beta)^m}$$

$$
\begin{aligned}
&= \frac{(|V|+1)\delta}{1-\epsilon/\beta}(1+\frac{\epsilon}{\beta-\epsilon})^{m-1} \\
&\leq \frac{(|V|+1)\delta}{1-\epsilon/\beta} e^{\frac{\epsilon(m-1)}{\beta-\epsilon}} \\
&\leq \frac{(|V|+1)\delta}{1-\epsilon} e^{\frac{\epsilon(m-1)}{(1-\epsilon)\beta}}.
\end{aligned}
$$

Let the last phase in the algorithm be numbered by M. It follows that $1 \leq W(M) \leq \frac{(|V|+1)\delta}{1-\epsilon} e^{\frac{\epsilon(M-1)}{(1-\epsilon)\beta}}$. Hence, we carry out

$$
\beta \leq \frac{\epsilon(M-1)}{(1-\epsilon)\ln\frac{1-\epsilon}{(|V|+1)\delta}}.
$$

Thus, the lemma follows.

Lemma 4 *iStream-APX generates a feasible streaming solution that makes* $\lambda \geq \frac{M-1}{\log_{1+\epsilon} 1/\delta}$.

Proof. At the end of the $(M-1)^{\text{th}}$ phase, $W(M-1) \leq 1$ for all node i. Thus, we deduct $s_i(M-1) = w_i(M-1) \leq 1/C_i$.

From Lemma 2 and Corollary 2, we know w_i and s_i increase at least by a factor of $1+\epsilon$ for every C_i units of flow through node i. Denoting the total flow through node i as F_i, we can carry out

$$
\begin{aligned}
F_i &\leq C_i \log_{1+\epsilon} \frac{w_i(M-1)}{w_i(0)} \\
&\leq C_i \log_{1+\epsilon} \frac{1/C_i}{\delta/C_i} \\
&= C_i \log_{1+\epsilon} \frac{1}{\delta}.
\end{aligned}
$$

Therefore, dividing all the flows through node i by a scaling factor of $\log_{1+\epsilon}\frac{1}{\delta}$, we obtain feasible flows through i without violating its uploading capacity C_i.

Applying the scaling factor, we can get feasible flows received by t of a total value $(M-1)d_t/\log_{1+\epsilon}\frac{1}{\delta}$ units. Accordingly, a feasible λ will follow

$$
\begin{aligned}
\lambda &\geq \frac{(M-1)d_t/\log_{1+\epsilon}\frac{1}{\delta}}{d_t} \\
&= \frac{(M-1)}{\log_{1+\epsilon}\frac{1}{\delta}}.
\end{aligned}
$$

Theorem 2 *The result of iStream-APX follows the property of* $\sum_{p\in P} l(p)f(p) \leq L$.

Proof. According to Corollary 3, in our procedure every time we route every flow with a cumulative delay of L, we increase φ by at least a factor of $1+\epsilon$.

Because $W(M-1) < 1$, we deduct that $\varphi(M-1) < 1/L$. Thus, in the first $M-1$ phases, the cumulative delay is at most $L \cdot \log_{1+\epsilon} \frac{\varphi(M-1)}{\varphi(0)} = L \cdot \log_{1+\epsilon} \frac{1}{\delta}$, i.e., $\sum_{p \in P} l(p) f(p) \leq L \cdot \log_{1+\epsilon} \frac{1}{\delta}$.

In the final procedure of the algorithm, we scale down all the flows proportionally by a scaling factor. Thus, applying the scaling factor of $\log_{1+\epsilon} \frac{1}{\delta}$, we have

$$\begin{aligned} \sum_{p \in P} l(p) f(p) &\leq \frac{L \log_{1+\epsilon} \frac{1}{\delta}}{\log_{1+\epsilon} \frac{1}{\delta}} \\ &= L. \end{aligned}$$

The theorem follows.

Theorem 3 *The approximation factor, denoted as ρ, is $1+\omega$.*

Proof. From Lemma 4, we have a feasible solution $\lambda = \frac{M-1}{\log_{1+\epsilon} \frac{1}{\delta}}$. It follows that

$$\begin{aligned} \frac{\beta}{\lambda} &= \frac{\beta \log_{1+\epsilon} \frac{1}{\delta}}{(M-1)} \\ &= \frac{\epsilon \ln \frac{1}{\delta}}{(1-\epsilon) \ln(1+\epsilon) \ln \frac{1-\epsilon}{(|V|+1)\delta}}. \end{aligned}$$

Let $\delta = \left(\frac{1-\epsilon}{|V|+1}\right)^{1/\epsilon}$. We have

$$\begin{aligned} \frac{\beta}{\lambda} &\leq \frac{\epsilon \ln \frac{1}{\delta}}{(1-\epsilon) \ln(1+\epsilon) \ln \frac{1-\epsilon}{(|V|+1)\delta}} \\ &= \frac{\epsilon}{(1-\epsilon)^2 \ln(1+\epsilon)} \\ &\leq \frac{\epsilon}{(1-\epsilon)^2 (\epsilon - \epsilon^2/2)} \\ &\leq (1-\epsilon)^{-3}. \end{aligned}$$

According to the strong duality theorem, if the dual has the optimal solution β, the primal also has an optimal value, denoted as $\mathrm{OPT}(\lambda)$, such that $\mathrm{OPT}(\lambda) = \beta$. Therefore, the approximation factor ρ can be obtained by

$$\begin{aligned} \rho &= \max \frac{\mathrm{OPT}(\lambda)}{\lambda} \\ &= \max \frac{\beta}{\lambda}. \end{aligned}$$

Now, we make an assignment of $\omega = (1-\epsilon)^{-3} - 1$. We have $\rho = 1 + \omega$. Thus, the proof is complete.

4.5 Running Time

In this section, we analyze the bound on running time. We define maximum binary search bound on L as $\Gamma = \sum_{j \in R} d_j \cdot \max_{p \in P} l(p)$.

Theorem 4 *Suppose the shortest path algorithm employed will consume a running time of Ψ. The running time of iStream is $O(\epsilon^{-2}\Psi|V|\log|V|\log\Gamma)$.*

Proof. According to weak duality theorem, we have $\frac{\beta}{\lambda} \geq 1$, which deduces

$$\frac{\beta}{M-1}\log_{1+\epsilon}\frac{1}{\delta} > 1.$$

So the number of phases $M < 1 + \beta\log_{1+\epsilon}\frac{1}{\delta}$. Because $\delta = \left(\frac{1-\epsilon}{|V|+1}\right)^{1/\epsilon}$, it follows that

$$M = \lceil \frac{\beta}{\epsilon}\log_{1+\epsilon}\frac{|V|+1}{1-\epsilon}\rceil$$

If iStream-APX does not stop within $2\lceil\frac{1}{\epsilon}\log_{1+\epsilon}\frac{|V|+1}{1-\epsilon}\rceil$ phases, we must have $\beta \geq 2$. We know $\mathrm{OPT}(\lambda) = \beta$ and we are pursuing $\mathrm{OPT}(\lambda) = 1$. In the case of $\beta \geq 2$, we break the current call for iStream-APX, and continue the binary search on L. So each call for iStream-APX will have $2\lceil\frac{1}{\epsilon}\log_{1+\epsilon}\frac{|V|+1}{1-\epsilon}\rceil = O(\epsilon^{-2}\log|V|)$ phases.

In order to compute the total running time, we need to calculate the number of steps in each call for iStream-APX. It is easy to see at every step except the the last step in an iteration, we increase either w_i of some node or φ by a factor at least $1+\epsilon$. So the number of steps exceeds the number of iterations by at most

$$|V|\log_{1+\epsilon}\frac{w_i(M-1)}{w_i(0)} = |V|\log_{1+\epsilon}\frac{1}{\delta} = O(\epsilon^{-2}|V|\log|V|). \tag{24}$$

Also, the maximum number of iterations in all phases is $|R| \cdot O(\epsilon^{-2}\log|V|) = O(\epsilon^{-2}|R|\log|V|)$. Combining this with Equation (24), we have the total number of steps in each call for iStream-APX is $O(\epsilon^{-2}(|V|+|R|)\log|V|) = O(\epsilon^{-2}|V|\log|V|)$.

Considering the number of calls for iStream-APX in binary search is bounded by $\log\Gamma$. Consequently, we can carry out the running time of iStream is bounded by $O(\epsilon^{-2}\Psi|V|\log|V|\log\Gamma)$. The theorem follows.

5 Conclusion

We present the design of iStream and derive a near-optimal approximation bound for its core component iStream-APX. To achieve a tractable theoretical analysis, we assume no network dynamics in the first stage of algorithm design. Although the assumption is strong in practical P2P applications, the value of this paper lies in the theoretical framework and analysis, which sheds light on the practical design. To reduce the complexity of the problem, we focus only on minimizing the communication delay. For packet scheduling, there exists a vast array of solutions. The mesh built from our algorithm can adopt any of these scheduling algorithms to yield low-delay streaming.

References

[1] D.-C. Tomozei and L. Massoulie, "Flow control for cost-efficient peer-to-peer streaming," in *INFOCOM, 2010 Proceedings IEEE*, 14-19 2010, pp. 1 –9.

[2] A. Sentinelli, G. Marfia, M. Gerla, L. Kleinrock, and S. Tewari, "Will IPTV ride the peer-to-peer stream?" *Communications Magazine, IEEE*, vol. 45, no. 6, pp. 86–92, June 2007.

[3] Z. Liu, Y. Shen, S. S. Panwar, K. W. Ross, and Y. Wang, "Using layered video to provide incentives in p2p live streaming," in *P2P-TV '07: Proceedings of the 2007 workshop on Peer-to-peer streaming and IP-TV*, 2007.

[4] D. Ren, Y.-T. Li, and S.-H. Chan, "On reducing mesh delay for peer-to-peer live streaming," in *INFOCOM '08*.

[5] F. Huang, B. Ravindran, and V. A. Kumar, "An approximation algorithm for minimum-delay peer-to-peer streaming," in *Peer-to-Peer Computing '09*, 2009.

[6] C. Wu and B. Li, "rstream: Resilient and optimal peer-to-peer streaming with rateless codes," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 19, no. 1, pp. 77–92, Jan. 2008.

[7] G. Bianchi, N. Blefari Melazzi, L. Bracciale, F. Lo Piccolo, and S. Salsano, "Streamline: An optimal distribution algorithm for peer-to-peer real-time streaming," *Parallel and Distributed Systems, IEEE Transactions on*, vol. PP, no. 99, pp. 1 –1, 2010.

[8] Z. Chen, K. Xue, and P. Hong, "A study on reducing chunk scheduling delay for mesh-based P2P live streaming," in *GCC '08*.

[9] Y. Liu, "On the minimum delay peer-to-peer video streaming: how realtime can it be?" in *ACM MULTIMEDIA '07*, 2007.

[10] N. Garg and J. Könemann, "Faster and simpler algorithms for multicommodity flow and other fractional packing problems," *SIAM J. Comput.*, vol. 37, no. 2, pp. 630–652, 2007.

[11] G. Karakostas, "Faster approximation schemes for fractional multicommodity flow problems," *ACM Trans. Algorithms*, vol. 4, no. 1, pp. 1–17, 2008.

[12] X. Hei, C. Liang, J. Liang, Y. Liu, and K. Ross, "A measurement study of a large-scale P2P IPTV system," *Multimedia, IEEE Transactions on*, vol. 9, no. 8, Dec. 2007.

[13] M. Hefeeda and O. Saleh, "Traffic modeling and proportional partial caching for peer-to-peer systems," *Networking, IEEE/ACM Transactions on*, vol. 16, no. 6, pp. 1447–1460, Dec. 2008.

[14] V. V. Vazirani, *Approximation Algorithm.* New York: Springer-Verlag New York, LLC, 2007.

[15] B. R. Fei Huang, Maleq Khan, "Technical report: On minimizing average end-to-end delay in p2p live streaming systems," Tech. Rep., 2010. [Online]. Available: http://staff.vbi.vt.edu/maleq/papers/APX_P2P_average_delay.pdf