# Quality-Aware Segment Transmission Scheduling in Peer-to-Peer Streaming Systems

Cheng-Hsin Hsu[*]
Deutsche Telekom R&D Lab USA
5050 El Camino Real #221
Los Altos, CA 94022

Mohamed Hefeeda
School of Computing Science
Simon Fraser University
Surrey, BC, Canada

## ABSTRACT

In peer-to-peer (P2P) mesh-based streaming systems, each video sequence is typically divided into segments, which are then streamed from multiple senders to a receiver. The receiver needs to coordinate the senders by specifying a transmission schedule for each of them. We consider the scheduling problem in both live and on-demand P2P streaming systems. We formulate the problem of scheduling segment transmission in order to maximize the perceived video quality of the receiver. We prove that this problem is NP-Complete. We present an integer linear programming (ILP) formulation for this problem, and we optimally solve it using an ILP solver. This optimal solution, however, is computationally expensive and is not suitable for real-time streaming systems. Thus, we propose a polynomial-time approximation algorithm, which yields transmission schedules with analytical guarantees on the worst-case performance. More precisely, we show that the approximation factor is at most 3, compared to the absolutely optimal solution as a benchmark. We implement the proposed approximation and optimal algorithms in a packet-level simulator for P2P streaming systems. We also implement two other scheduling algorithms proposed in the literature and used in popular P2P streaming systems. By simulating large P2P systems and streaming nine real video sequences with diverse visual and motion characteristics, we demonstrate that our proposed approximation algorithm: (i) produces near-optimal perceived video quality, (ii) can run in real time, and (iii) outperforms other algorithms in terms of perceived video quality, smoothness of the rendered videos, and balancing the load across sending peers. For example, our simulation results indicate that the proposed algorithm outperforms heuristic algorithms used in current systems by up to 8 dB in perceived video quality and up to 20% in continuity index.

[*]This work was done while the author was a PhD student at Simon Fraser University.

## Categories and Subject Descriptors

H.4.3 [**Information Systems Applications**]: Communications Applications; C.2.4 [**Computer-Communication Networks**]: Distributed Systems

## General Terms

Design

## Keywords

Optimization, peer-to-peer streaming, video streaming, perceived video quality, transmission scheduling

## 1. INTRODUCTION

As video streaming over the Internet is getting increasingly popular [1], many multimedia objects are distributed using peer-to-peer (P2P) streaming systems [16, 23, 26, 30], which reduce the deployment cost of expensive streaming servers. P2P streaming systems can be built in two ways [17]: (i) tree-based systems in which one or more trees are used to connect peers for transferring content [7, 9, 20], and (ii) mesh-based systems in which each peer connects to a few neighboring peers without an explicit network topology [16, 21, 33]. We consider mesh-based systems, because they incur lower maintenance overhead, adapt better to network dynamics, are easier to implement [3], and lead to better perceived video quality [19].

In mesh-based systems, a video sequence is partitioned into small *segments*, and segments are transmitted from multiple senders to a receiver. The receiving peer *must* coordinate the segment transmission from its senders. More precisely, a receiver runs a *scheduling* algorithm to compose a transmission schedule for its senders, which specifies for each sender the assigned segments and their transmission times.

Composing segment transmission schedules is not an easy task, as P2P streaming systems impose time constraints on segment transmission. Segments arriving at the receiver after their decoding deadlines are essentially useless, because they cannot be rendered to users for improving video quality. Hence, segment scheduling algorithms should strive to maximize the perceived video quality delivered by the on-time segments, which refer to those segments that meet their decoding deadlines. Optimally constructing segment schedules may be computationally expensive [33], and thus existing systems either resort to simple heuristic algorithms [2,21,33], or assign each segment an ad-hoc *utility* and solve the simplified problem of maximizing the system-wide utility [8,32]. These algorithms provide *no* performance guarantees on the number of on-time delivered segments, and may result in playout glitches and degraded perceived video quality. A recent work pointed out that these existing algorithms might work in live streaming systems as peers in these systems share a small scheduling window and are less sensitive to the performance of scheduling algorithms; however, they do not work well in on-demand streaming systems [3].

In this paper, we study the problem of scheduling segment transmission in *both* live and on-demand P2P streaming systems. Our goal is to maximize the perceived video quality by scheduling the segment transmission so that segments that are more critical in terms of video quality are given higher priority to meet their deadlines. We first prove that this segment scheduling problem is NP-Complete. We then present an integer linear programming (ILP) formulation for this problem, and we optimally solve it using an ILP solver. Optimally solving this ILP problem may take long time, which is not suitable for P2P streaming systems that are fairly dynamic. Thus, we propose an efficient approximation algorithm, which constructs transmission schedules with performance *guarantees*. We analytically derive the performance bound and time complexity of the proposed algorithm. We also implement an event-driven simulator to evaluate it using a large P2P network and real video sequences with diverse characteristics. The simulation results show that the proposed algorithm leads to almost optimal perceived video quality, and outperforms heuristic algorithms used in current systems.

The rest of this paper is organized as follows. Sec. 2 summarizes the related work in the literature. In Sec. 3, we describe the considered system model and formulate the segment scheduling problem. We develop our approximation algorithm in Sec. 4, and we analytically analyze it in the same section. We evaluate the proposed algorithm using extensive simulations in Sec. 5. Sec. 6 concludes this paper.

## 2. RELATED WORK

Several commercial P2P streaming systems have been deployed, which implement proprietary scheduling algorithms. A recent measurement study on PPLive [22] reports that users suffer from long start-up delays and playout lags, and suggests that better segment scheduling algorithms are required [12].

Optimally computing segment schedules to maximize the perceived video quality, however, is computationally complex. Therefore, many P2P streaming systems, such as [2, 21, 33], resort to simple heuristics for segment scheduling. The authors of [21] propose to randomly schedule segment transmission. The authors of [33] assume that segments with fewer potential senders are more likely to miss their deadlines, and propose to schedule the segments with fewer potential senders earlier. The authors of [2] describe a weighted round-robin algorithm based on senders' bandwidth. Unlike our algorithm, these *heuristic* algorithms do not provide any performance guarantees on perceived video quality, and do not perform well in on-demand streaming systems [3].

One way to cope with the hardness of the segment scheduling problem is to *simplify* the objective function from the perceived video quality to the sum of *ad-hoc* utility functions [8,32]. The authors of [32] define a utility for each segment as a function of the rarity, which is the number of potential senders of this segment and the urgency, which is the time difference between the current time and the deadline of that segment. They then transform the segment scheduling problem into a min-cost flow problem. We note that although the min-cost flow problem can be optimally solved, the resulting schedules do *not* maximize the perceived video quality, which is the objective of the original problem.

The authors of [8] formulate an optimization problem to maximize the perceived video quality, and they solve it using an iterative descent algorithm. This algorithm, however, is computationally expensive and cannot be used in real-time systems. Therefore, they simplify the original formulation by proposing an ad-hoc utility function for each segment, which defines the multiplication of each segment's R-D (rate-distortion) efficiency, rarity, and urgency as its utility. They then greedily schedule the segments, i.e., they schedule the segments with higher utility values earlier. This greedy algorithm does *not* produce optimal schedules, nor does it provide any guaranteed performance. The works in [8, 32] are different from our work in the sense that we solve the original segment scheduling problem, and we propose efficient approximation algorithm with guaranteed performance.

Several other works are related to the segment scheduling problem, but they do not directly solve it. The authors of [6] propose a P2P system that measures the time required to download the entire video from a number of senders and uses this information to choose senders from a large group of potential senders. The authors of [3] propose using network coding to bypass the scheduling problem among small blocks belonging to the same relatively large segment. However, employing network coding may impose higher processing overhead on peers, which may require special hardware to speed up the decoding process [24], and is not easy to deploy. Finally, several segment scheduling algorithms, such as [15], have been proposed for tree-based systems. They are, however, not applicable to mesh-based systems, in which peers have no knowledge on the global network topology.

## 3. SEGMENT TRANSMISSION SCHEDULING

In this section, we first provide an overview of the P2P system model employed in this paper. We then state the segment transmission problem and show its hardness. Then we mathematically formulate it as an integer linear programming problem. For quick reference, we list all symbols used in the paper in Table 1.

### 3.1 System Model

We consider mesh-based (also known as swarm-based and data-driven) P2P streaming systems, which are widely deployed and used. Examples of such systems include CoolStreaming [33], PPLive [22], UUSee [28], SopCast [25], and TVAnts [27]. In these systems, peers form swarms for exchanging video data. Each swarm contains a subset of the peers, and a peer may participate in multiple swarms. Data availability on peers is propagated through exchanging control messages, such as buffer maps which indicate which video segments peers currently have in their buffers and thus can upload. Using these buffer maps, peers *pull* video segments from each other. More specifically, a receiving peer simultaneously requests segments from different sending peers. This is done by forming a segment transmission schedule by which the receiver specifies for each sender which

**Table 1: List of symbols used in the paper.**

| Symbol | Description |
|---|---|
| $\delta$ | scheduling window size |
| $F$ | frame rate |
| $G$ | number of frames in each segment |
| $M$ | number of senders |
| $N$ | number of segments |
| $T$ | number of time slots |
| $m_i$ | sender $i$ |
| $n_i$ | segment $i$ |
| $t_i$ | transmission time for segment $i$ |
| $u_i$ | whether segment $i$ arrives on-time |
| $b_m$ | bandwidth of sender $m$ |
| $\mathbf{Q}_m$ | schedule for sender $m$ |
| $s_n$ | size of segment $n$ |
| $w_n$ | weight or value of segment $n$ |
| $d_n$ | deadline of segment $n$ |
| $a_{n,m}$ | availability of segment $n$ on sender $m$ |
| $x_{n,m,t}$ | variable for $m$ to transmit $n$ at time $t$ |
| $\mathbf{I}_m$ | set of intervals to color |
| $P$ | rounding factor |
| $U$ | number of variables in Eq. (1) |
| $V$ | number of constraints in Eq. (1) |
| $L$ | number of bits to encode Eq. (1) |
| $\alpha$ | perceived video quality |
| $\beta$ | continuity index |
| $\gamma$ | load balancing factor |

segments to transmit and when. In video streaming systems, the arrival times of segments are critical, as segments arriving after their playback times cannot be rendered to users and are essentially useless.

The problem addressed in this paper is to compute transmission schedules for receivers in order to optimize their perceived video quality. Transmission schedules are computed for recurring time windows, which are usually in the order of seconds. Before the time window ends, the scheduling algorithm must be invoked again to compute the transmission schedule for the next window. The algorithm is also invoked whenever the set of senders or their characteristics change, e.g., if a sender fails or leaves the P2P network. Since the scheduling algorithm is invoked frequently within short periods, it must be computationally efficient and runs in real time.

Our problem formulation and solution employ a *realistic* model for P2P streaming systems. Thus our proposed algorithm can readily be implemented in current mesh-based P2P streaming systems to improve their performance. Particularly, we consider that P2P streaming systems are highly dynamic and peers will join and leave frequently. Thus, we design our algorithm to be light-weight and can be invoked whenever such events occur in order to quickly recompute a new transmission schedule. In addition, the dynamic propagation and replication of the video segments in the P2P streaming system can easily be handled by our algorithm. This is because the segment propagation will trigger peers to update their buffer maps to reflect the availability of the newly acquired segments. When these buffer maps are exchanged among peers in control messages, our scheduling algorithm will account for the new segments in computing new transmission schedules.

It is important to emphasize that our work in this paper focuses on a single, but critical, component of the P2P streaming system, which is the transmission scheduler. We present rigorous design of this component with mathematical formulation, complexity analysis, analytical guarantee on the performance of the computed schedules, and extensive simulations. We are *not* proposing a new, complete, P2P streaming system. We, however, do not impose any assumptions on the other components of the system, e.g., the overlay management, sender-receiver matching, exchanging control messages, churn handling, and incentive schemes. We assume that these components will function according to whatever protocols dictated by the specific P2P streaming system and eventually a set of potential senders will be presented to a receiver for obtaining the video data. Given data availability of each sender, our work is to make the best out of this set of senders for the receiver.

## 3.2 Problem Statement and Hardness

We study the problem of transmitting a video stream from multiple, $M$, senders to a receiver in a P2P streaming system. This stream consists of a series of coded video frames at frame rate $F$ fps (frames per second), where each frame has a decoding deadline. Coded video frames that arrive at the receiver *after* their decoding deadlines are *useless.* To efficiently transmit video frames over the network, multiple consecutive coded frames are aggregated into a *segment*, which is the smallest transmission unit. Segment sizes are flexible and can be chosen based on the structure of a subject video stream. For example, one P2P streaming system may choose to construct a segment for each video frame for finer-grained scheduling, while another system may prefer to create a segment for every GoP (group-of-picture) for lower overhead. We consider a very general P2P streaming system that aggregates $G$ coded frames in each segment, where video frames can have different sizes. We let $N$ be the number of segments in the whole video stream. Since each segment consists of $G$ coded frames, it has a playout time of $G/F$. Furthermore, segments are in different sizes because coded frames in video streams typically vary in sizes. We let $s_n$ kb be the size of segment $n$, where $1 \leq n \leq N$. Segment $n$ has a decoding deadline $d_n = (n-1)G/F$ sec, which is the decoding deadline of the first video frame in that segment.

To generate feasible schedules, the receiver monitors its senders in terms of segment availability and upload bandwidth. We let $a_{n,m}$ be the availability of segment $n$ ($1 \leq n \leq N$) on sender $m$ ($1 \leq m \leq M$). The receiver sets $a_{n,m} = 1$ if sender $m$ has a copy of segment $n$, and $a_{n,m} = 0$ otherwise. Each sender employs bandwidth estimation methods to estimate its upload bandwidth, and divides it among all connected receivers. That is, a receiver keeps track of the upload bandwidth of its senders by querying each sender. We let $b_m$ kbps be the upload bandwidth of sender $m$. With the segment availability and sender bandwidth, the receiver composes a segment schedule for a time window of $\delta$ sec. $\delta$ is a system parameter. The resulting schedule is sent to senders, and senders transmit segments following the schedule. Furthermore, decoding different segments results in different video quality improvements. Let $w_n$ be the *weight* or the *value* of segment $n$, which represents the quality improvement brought by this segment. We consider a general problem in which the definition of $w_n$ is determined by P2P systems. P2P systems with enough computational power may *pre-compute* a perceived video quality value as the $w_n$ value for each segment $n$ ($1 \leq n \leq N$) offline, while P2P systems with limited computational power may assign $w_n$ values heuristically.

Our goal is to maximize the sum of weights of all on-time segments. We exclude late segments as they cannot be used toward video quality enhancement. As mentioned above, the considered problem is general and can take any definition of $w_n$. For computationally powerful P2P systems, we may use perceived video quality metrics, such as PSNR (Peak Signal-to-Noise), to define $w_n$, while in other P2P systems, we may heuristically define $w_n$. In the rest of paper, we use PSNR to define $w_n$ as it is widely used in multimedia systems [29, Sec. 1.5.5]. Nevertheless, the consider problem and our proposed solution are general, and P2P systems may compute the $w_n$ value using their own definitions. More precisely, we let $w_n$ be the average video quality in PSNR of segment $n$. We mention that the $w_n$ weights (or values) are typically pre-computed by video coders, and inserted into coded streams as *meta data*. That is, they are not computed at streaming time, and not by the senders nor the receiver. The computation of PSNR values can be done empirically for higher accuracy or by some rate-distortion (R-D) models for lower overhead. With these notations, we formally describe the considered problem in the following.

Problem 1 (Scheduling). *We consider the segment transmission scheduling problem of video sequences in P2P streaming systems. The problem is to construct an optimal transmission schedule* $\mathbf{Q} = \{<m_i, n_i, t_i>,\ 1 \leq i \leq Q\}$ *for a time window of* $\delta$ *sec, where* $m_i$ *indicates the sender,* $n_i$ *represents the segment,* $t_i$ *is the transmission time, and* $Q$ *is the number of segments in the time window. A segment* $<m_i, n_i, t_i>$ *is said to be on-time if and only if* $t_i + s_{n_i}/r_{m_i} \leq d_{n_i}$. *We let* $u_n = 1$ *if segment* $n$ *arrives on-time at the receiver from any of its senders, and* $u_n = 0$ *otherwise. The objective is to maximize the perceived video quality, which is the sum of weights of all segments that arrive on-time.* □

In the next theorem, we prove that solving this segment scheduling problem is computationally expensive.

Theorem 1 (Hardness). *The scheduling problem defined in Problem 1 is NP-Complete.*

Proof. We reduce an NP-Complete machine scheduling problem [5, Sec. 5.3] to the segment scheduling problem. The machine scheduling problem schedules $J$ jobs on $K$ identical machines, and each job $j$ ($1 \leq j \leq J$) takes time $p_j$ to complete. Each job can be scheduled on one machine and can not be preempted, and each machine can process one job at any time. The goal is to minimize the makespan $C_{\max}$, which is the maximal completion time of all jobs, and the problem is to determine whether there exists a schedule with $C_{\max} \leq c_0$ for a given constant $c_0$.

For each machine scheduling problem, we construct a segment scheduling problem as follows. Let the number of senders $M = K$, and the number of segments $N = J$. For each segment $n$ ($1 \leq n \leq N$), let $s_n = p_j$, $d_n = c_0$, and $w_n = 1$. We let $b_s = 1$ for all $1 \leq s \leq S$. Finally, the problem is to determine whether there exists a schedule that achieves the sum of weights $\sum_{n=1}^{N} u_n w_n \geq N$. Since the segment scheduling problem is constructed and can be verified in polynomial time, and $C_{\max} \leq c_0 \iff \sum_{n=1}^{N} u_n w_n$, it is NP-Complete. □

## 3.3 Problem Formulation

We formulate the segment scheduling problem as a time-indexed integer linear programming (ILP) problem. Time-indexed formulations discretize the time axis into $T$ time slots, where $T$ is large so that the time slots are fine enough to represent any feasible schedule without reducing the value of the objective function. We let $x_{n,m,t}$ be a 0-1 variable for each $n = 1, 2, \ldots, N$, $m = 1, 2, \ldots, M$ and $t = 1, 2, \ldots, T$, where $x_{n,m,t} = 1$ if segment $n$ is scheduled to be transmitted by sender $m$ at time $t$, and $x_{n,m,t} = 0$ otherwise. We note that, according to the definition, while the transmission of a segment often spans over several time slots, only the *first* time slot has an $x$ value of 1.

We formulate the considered segment scheduling problem as:

$$z^* = \max \sum_{m=1}^{M} \sum_{n=1}^{N} \sum_{t=1}^{d_n - \frac{s_n}{b_m}} w_n x_{n,m,t} \tag{1a}$$

$$\text{s.t.} \quad x_{\hat{n},\hat{m},\hat{t}} \leq a_{\hat{n},\hat{m}} \tag{1b}$$

$$\sum_{j=1}^{N} \sum_{k=\hat{t}-\frac{s_j}{b_{\hat{m}}}+1}^{\hat{t}} x_{j,\hat{m},k} \leq 1 \tag{1c}$$

$$\sum_{i=1}^{M} \sum_{j=1}^{d_{\hat{n}} - \frac{s_{\hat{n}}}{b_i}} x_{\hat{n},i,j} \leq 1 \tag{1d}$$

$$x_{\hat{n},\hat{m},\hat{t}} \in \{0,1\},\ \forall\ \hat{m} = 1, 2, \ldots, M,$$
$$\hat{n} = 1, 2, \ldots, N,\ \hat{t} = 1, 2, \ldots, T. \tag{1e}$$

In this formulation, the objective function in Eq. (1a) is to maximize the sum of weights of on-time segments, where the three summations iterate through all senders, segments, and time slots, respectively. Note that the last summation stops at time $d_n - s_n/b_m$, because scheduling segment $n$ *after* that time results in a late segment, which cannot improve video quality. The constraint in Eq. (1b) makes sure that we always schedule a segment to a sender who holds a copy of it, as it prevents the combination of $x_{\hat{n},\hat{m},\hat{t}} = 1$ and $a_{\hat{n},\hat{m}} = 0$ for all $\hat{t} = 1, 2, \ldots, T$. In Eq. (1c), observe that any segment $j$ scheduled for sender $\hat{m}$ between time $\hat{t} - s_j/b_{\hat{m}} + 1$ and $\hat{t}$ would *occupy* the time slot $\hat{t}$ as transmitting segment $j$ takes time $s_j/b_{\hat{m}}$. Therefore, by considering all these segments, the constraint in Eq. (1c) ensures that at most one segment is scheduled for each sender at any time $\hat{t}$. Last, the constraint in Eq. (1d) prevents segments from being scheduled to more than one sender.

**An Optimal Algorithm.** To get optimal segment schedules, we can solve the formulation in Eq. (1) using ILP solvers that support 0-1 ILP problems. In this paper, we use the general ILP solver in CPLEX [13] package for this purpose. We refer to this approach as the OPT algorithm, and we use it as a benchmark to assess the performance of the algorithm proposed in the next section.

## 4. PROPOSED ALGORITHM

Solving ILP problems is computationally expensive and may not be possible in real time. We develop an efficient approximation algorithm in this section, and we formally derive its approximation factor.

## 4.1 Overview

We propose an efficient algorithm based on the linear programming (LP) relaxation of the ILP formulation in Eq. (1). The LP relaxed formulation allows any $x_{\hat{n},\hat{m},\hat{t}}$ in Eq. (1e) to take fractional values, where $0 \leq x_{\hat{n},\hat{m},\hat{t}} \leq 1$. This LP relaxed formulation can be optimally solved using efficient LP solvers that implement Simplex or Interior Point Methods (IPMs). We use $\bar{x}_{\hat{n},\hat{m},\hat{t}}$ to denote the *fractional schedule* produced by an LP solver. We mention that fractional schedules of the LP relaxed formulation are not feasible to the original scheduling problem. This is because, in the LP relaxed formulation, the constraints in Eqs. (1c) and (1d) are interpreted in a different way: the constraint in Eq. (1c) makes sure that the fractions of all segments scheduled for each sender sum to at most one at any time, and the constraint in Eq. (1d) ensures that fractions of each segment scheduled to all senders at any time sum to at most one.

To compute a schedule for the original scheduling problem, we propose a rounding algorithm to convert fractional schedules of the ILP formulation into integral *feasible* schedules, albeit with a small approximation factor. We first explain how the proposed rounding algorithm handles a single sender $m$ for any $m = 1, 2, \ldots, M$. We then expand the description to the general case of $M$ senders. For a specific sender $m$, the rounding algorithm consists of two steps: (i) it transforms the *fractional schedule* into several feasible *integral schedules*, and (ii) it selects the best schedule out of all integral schedules. More precisely, the rounding algorithm first rounds the fractional schedule $\bar{x}_{\hat{n},m,\hat{t}}$ for all $\hat{n} = 1, 2, \ldots, N$ and $\hat{t} = 1, 2, \ldots, T$ to multiples of $1/P$, where $P = (TN)^2$. This is achieved by creating $\lfloor \bar{x}_{\hat{n},m,\hat{t}} \times P \rfloor$ copies of *time intervals* $[\hat{t}, \hat{t} + s_{\hat{n}}/b_m]$ for each positive $\bar{x}_{\hat{n},m,\hat{t}}$. These time intervals are then put in the set $\mathbf{I}_m$.

Next, we color the intervals in $\mathbf{I}_m$ using the minimum number of colors, so that: (i) two intervals overlapping in time have different colors, and (ii) two intervals of the same segment have different colors. This can be done by first sorting all intervals on their starting times, and then sequentially coloring them in that order. Once the coloring is done, intervals with the same color have two nice properties, they: (i) never overlap in time, and (ii) are not associated with the same segment. Therefore, we can construct an integral schedule using all intervals that have the same color. This gives us several feasible schedules. We then compute the objective function value of each feasible schedule, and we choose the schedule with the largest objective function value. We let this schedule be $\mathbf{Q}_m = \{<m, \hat{n}, \hat{t}>\}$, which indicates that sender $m$ should start transmitting segment $\hat{n}$ at time $\hat{t}$.

For the general case of $M$ senders, the rounding algorithm sequentially schedules segments for all senders. More specifically, for each sender $m$, the rounding algorithm sets all $\bar{x}_{n,m,\hat{t}} = 0$ for all $\hat{t} = 1, 2, \ldots, T$, *if* segment $n$ has been scheduled for any sender $\bar{m}$, where $\bar{m} < m$. This is to avoid scheduling a segment to multiple senders, which violates the constraint in Eq. (1d). Once the feasible schedule for sender $m$ ($1 \leq m \leq M - 1$) is derived, the rounding algorithm considers sender $m + 1$. The rounding algorithm stops after iterating through all senders, and returns the segment schedule $\mathbf{Q}_m$, for all $m = 1, 2, \ldots, M$. Since our proposed algorithm takes user-specified weights, we call it Weighted Segment Scheduling (WSS) algorithm.

**WSS: Weighted Segment Scheduling**

```
1.  let $\mathbf{Q}_m = \varnothing$, where $m = 1, 2 \ldots, M$
2.  let $P = (TN)^2$
3.  compute optimal $\bar{x}_{\hat{m},\hat{n},\hat{t}}$ for the relaxed Eq. (1)
4.  for $m = 1$ to $M$  // consider senders sequentially
5.    let $\mathbf{I}_m = \varnothing$  // time intervals
6.    foreach positive $\bar{x}_{m,\hat{n},\hat{t}}$
7.      insert $\lfloor \bar{x}_{m,\hat{n},\hat{t}} \times P \rfloor$ copies of interval $[\hat{t}$,
7.      $\hat{t} + s_{\hat{n}}/b_m]$ to $\mathbf{I}_m$
8.    endfor
9.    color intervals in $\mathbf{I}_m$ using fewest colors, so that
9.    two intervals overlapping in time or associated
9.    with the same segment have different colors
10.   construct a feasible schedule for each color
11.   let $\mathbf{Q}_m$ be feasible schedule that results in
11.   highest objective function value $z$
12.   for $\hat{m} = m + 1$ to $M$  // mark as scheduled
13.     let $\bar{x}_{\hat{m},\hat{n},\hat{t}} = 0$
14.   endfor
15. endfor
16. return $\mathbf{Q}_1, \mathbf{Q}_2, \ldots, \mathbf{Q}_M$.
```

**Figure 1: The proposed approximation algorithm.**

Fig. 1 shows a high-level pseudocode of the WSS algorithm. It solves the LP relaxed formulation in line 3. The rounding is sequentially done using the for-loop between lines 4 and 15. The foreach-loop between lines 6 and 8 builds the set $\mathbf{I}_m$ of time intervals for sender $m$ based on its fractional schedule. Line 9 colors $\mathbf{I}_m$ using as few colors as possible. Line 10 builds a set of feasible integral schedules, and line 11 picks the schedule $\mathbf{Q}_m$ that leads to the highest objective function value $z$. The for-loop between lines 12 and 14 prevents segments from being scheduled to multiple senders. The algorithm returns $\mathbf{Q}_1, \mathbf{Q}_2, \ldots, \mathbf{Q}_m$ in line 16.

## 4.2 Analysis and Complexity

We analyze the performance of the WSS algorithm in two steps. We first analyze the problem with a single sender, and then extend the analysis to multiple senders. In the next lemma, we derive the approximation factor of the WSS algorithm with one sender. In the proof, we first show that the floor function in line 7 incurs negligible drops on the objective function value. We then show that the number of colors required in line 9 is bounded. Therefore, we only have a *small* number of feasible schedules, and at least one of them leads to an objective value higher than half of the optimum.

LEMMA 1. *The WSS algorithm in Fig. 1 has an approximation factor of 2, when there is only one sender.*

PROOF. Let $z^*$ be the optimal objective function value. In line 7, each $\bar{x}$ is rounded to a multiple of $1/P$. This means that each $u_n^* w_n$ in the objective function (Eq. (1a)) is reduced by at most $1/P$. Let $w_{n'}$ be the maximal weight among all segments, i.e., $w_{n'} \geq w_n$ for all $n = 1, 2, \ldots, N$. Since we have $(TN)$ $\bar{x}$ variables, the $z^*$ drops for at most $(TN)w_{n'}/P = w_{n'}/(TN)$. Observe that, since scheduling only segment $n'$ results in a feasible schedule with a sum of weights $w_{n'}$, we have $q^* \geq w_{n'}$. Thus the rounding in line 7 reduces the $q^*$ value by at most a factor of $1/(TN)$, which is negligible as the number of time slots $T$ is large.

From the formulation in Eq. (1), the fractional schedule produced by LP solvers has the following property: the number of overlapping intervals is at most $P$, and the number of intervals of the same segment is at most $P$. Following the coloring strategy in line 9, the intervals can be colored with at most $C = 1 + (P-1) + (P-1) = 2P-1$ different colors. Furthermore, line 7 indicates that $q^* = \sum_{c=1}^{C} \hat{w}_c / P$, where $\hat{w}_c$ $(c = 1, 2, \ldots, C)$ is the objective function value of the feasible schedule derived from color $c$. Define $\alpha_c = 1/P$ for each $c = 1, 2, \ldots, C$, we write

$$q^* = \sum_{c=1}^{C} \hat{w}_c \alpha_c. \tag{2}$$

Moreover, we know

$$\sum_{c=1}^{C} \alpha_c \leq \sum_{c=1}^{2P-1} (1/P) = 2 - (1/P) < 2. \tag{3}$$

With Eqs. (2) and (3), we claim that there exists a color $c^*$ so that $\hat{w}_{c^*} \geq \frac{1}{2} q^*$. This claim can be easily proved by contradiction. Thus the feasible schedule identified by line 11 achieves approximation factor 2, when $M = 1$. $\square$

Next, we extend the above lemma to the general case with multiple senders.

Theorem 2 (Performance). *The WSS algorithm in Fig. 1 achieves approximation factor of 3, when there are multiple senders.*

Proof. For $m = 1, 2, \ldots, M$, we let $\mathbf{R}_m$ be the fractional schedule of sender $m$ produced in line 3, and $\mathbf{Q}_m$ be the integral schedule of sender $m$ returned in line 16. We also define $\mathbf{R}'_m$ as the fractional schedule of sender $m$ after discarding all segment scheduled in $\mathbf{Q}_1, \mathbf{Q}_2, \ldots, \mathbf{Q}_{m-1}$. We use $w(\cdot)$ to denote the objective function value of $\mathbf{R}_m$, $\mathbf{Q}_m$, and $\mathbf{R}'_m$.

Following Theorem 1, we have $w(\mathbf{Q}_m) \geq (1/2) w(\mathbf{R}'_m)$, for all $m = 1, 2, \ldots, M$. Since $\mathbf{Q}_m$ are mutually disjoint and $\mathbf{Q}'_m$ are mutually disjoint, we have:

$$\sum_{m=1}^{M} w(\mathbf{Q}_m) \geq \frac{1}{2} \sum_{m=1}^{M} w(\mathbf{R}'_m). \tag{4}$$

Since every scheduled segment is marked by the for-loop between lines 12 and 14, we write:

$$\sum_{m=1}^{M} w(\mathbf{R}'_m) \geq \sum_{m=1}^{M} w(\mathbf{R}_m) - \sum_{m=1}^{M} w(\mathbf{Q}_m). \tag{5}$$

Combining Eqs. (4) and (5) gives:

$$\sum_{m=1}^{M} w(\mathbf{Q}_m) \geq \frac{1}{2} \sum_{m=1}^{M} w(\mathbf{R}_m) - \frac{1}{2} \sum_{m=1}^{M} w(\mathbf{Q}_m).$$

Rearranging this inequality, we get

$$\sum_{m=1}^{M} w(\mathbf{Q}_m) \geq (1/3) \sum_{m=1}^{M} w(\mathbf{R}_m),$$

which yields an approximation factor of 3. $\square$

We show that the proposed WSS algorithm is a polynomial time algorithm in the next theorem.

Theorem 3 (Complexity). *The WSS algorithm proposed in Fig. 1 runs in polynomial time, i.e., it terminates in $O(3UP + M(2P-1) + MU + U^{1.5}V^2L)$ arithmetic operations, where $P = (TN)^2$ is the rounding factor, $U = MNT$ is the number of variables of the formulation in Eq. (1), $V = MNT + MT + T$ is the number of constraints in this formulation, and $L = 8U + UV + V$ is the number of total bits required to encode this formulation.*

Proof. Notice that there are $U = MNT$ variables in the objective function shown in Eq. (1a). Furthermore, there are $MNT$ constraints in Eq. (1b), $MT$ constraints in Eq. (1c), and $N$ constraints in Eq. (1d), and thus we have $V = MNT + MT + N$ total constraints.

The time complexity before line 4 can be bounded by $O(\sqrt{U}LUV^2)$ following the complexity results of IPMs in the literature [31, Sec. 4.6], where $L$ is the number of total bits required to write the formulation. $L$ can be derived as follows. In Eq. (1a), we consider the weights $w_n$ for all $n = 1, 2, \ldots, N$ are represented as 8-bit floating point values, which take $8U$ bits. In Eqs. (1b)–(1d), we have $UV$ coefficients on the left of the inequalities and $U$ on the right. Observe that all these coefficients can be encoded in a single bit, which means encoding the the constraints requires $UV + V$ bits. Combining the objective function with constraints, we get $L = 8U + UV + V$.

Next, we compute the number of operations required by each iteration of the for-loop between lines 4 and 15. The foreach-loop in lines 6–8 creates at most $O(NTP)$ intervals. The coloring in line 9 can be done in a single-pass scan on the intervals in $\mathbf{I}_m$, which takes $O(NTP)$ operations. Constructing feasible solutions in line 10 requires $O(NTP)$, and finding the solution with the highest objective function value consumes $O(NTP) + O(2P-1)$. The for-loop between lines 12 and 14 takes $O(MNT)$ operations. This leads to $O(3NTP + 2P + MNT - 1)$ operations in each iteration between lines 4–15, and $O(3UP + M(2P-1) + MU)$ for the whole rounding algorithm. Combining this with $O(\sqrt{U}LUV^2)$ yields the theorem. $\square$

Finally, we mention that a similar LP relaxation was used in designing other scheduling algorithms with different objectives, such as minimizing the task completion time [10, 14], and maximizing total weight of tasks completed by their due dates [4].

## 5. EVALUATION

In this section, we first detail the setup of our simulations and define several performance metrics considered in the simulations. We then present the simulation results.

### 5.1 Setup

We have implemented an event-driven simulator in Java to evaluate the performance of the proposed segment scheduling algorithm. We have implemented four scheduling algorithms in this simulator: OPT, WSS, RF, and MC. OPT and WSS are the implementations of our optimal and approximation algorithm, respectively. The RF algorithm implements the rarest first algorithm used in many P2P systems [33], and it schedules the segment with the fewest potential senders first. The MC algorithm is based on the ILP

Table 2: List of video sequences used in the paper.

| Sequence | Rate (kbps) | Description |
|---|---|---|
| Bus | 1834 | running bus in short distance |
| City | 589 | pan over a city in long distance |
| Crew | 1344 | astronauts walk by in short distance |
| Football | 2340 | complex movements and details |
| Foreman | 723 | talking person, camera movements |
| Harbour | 1592 | both close-by and far-away objects |
| Ice | 744 | a rink with several skating persons |
| Mobile | 1731 | moving toys and calendar with saturated colors |
| Soccer | 1215 | many details around players' legs |

Table 3: Peer Upload Bandwidth Distribution.

| Distribution (%) | 10.0 | 14.3 | 8.6 | 12.5 | 2.2 | 1.4 | 6.6 | 28.1 | 16.3 |
|---|---|---|---|---|---|---|---|---|---|
| Total Bandwidth (kbps) | 256 | 320 | 384 | 448 | 512 | 640 | 768 | 1024 | $> 1500$ |
| Contributed Bandwidth (kbps) | 150 | 250 | 300 | 350 | 400 | 500 | 600 | 800 | 1000 |

formulation proposed in [32], which is converted into a min-cost flow problem and solved by combinatorial algorithms. While we employ the same utility function defined in the evaluation section of [32], the original algorithm can only schedule transmission of fixed-size blocks. We, therefore, extend that algorithm to support variable-size segments by: (i) dividing each segment into blocks, (ii) solving the block transmission problem using their algorithm, and (iii) performing a majority vote that schedules each segment to the sender that is assigned the most blocks. Our implementations use the CPLEX [13] package to solve the ILP and LP problems. In particular, the CPLEX package provides a set of Java class libraries that allow us to specify and solve our problems using Java syntax through JNI (Java native interface).

We use a diverse set of nine video sequences in our evaluation, which are Bus, City, Crew, Football, Foreman, Harbour, Ice, Mobile, and Soccer. During the streaming of each video sequence, the scheduling algorithm is invoked many times, at least 10 times for the shortest sequence. We compress all video sequences using an H.264 coder with GoP size of 8 and QP (quantization parameter) 25. The videos have frame rate $F = 30$ fps and are in CIF (352x288) resolution. Table 2 summarizes the average bit rate of each video stream. We use PSNR as the perceived video quality metric in our simulations. The simulator aggregates every $G = 8$ video frames into a segment, and assigns each segment $n$ a decoding deadline of $G(n-1)/F$, where $n = 1, 2, \dots, N$.

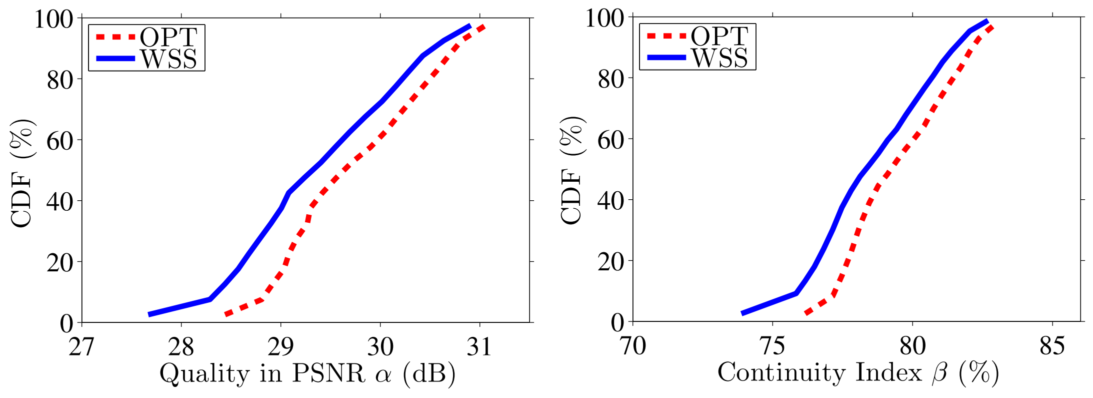
We simulate a system with a total of 2,000 peers. We initially pre-deploy the video sequences at only 1% of the peers chosen randomly, which form the initial seeding peers. We run the simulation for 24 hours of simulation time. Individual peers join and leave a swarm at different times during the streaming of a video sequence. The joining and leaving times are randomly chosen from the simulation time period following a uniform distribution within the whole simulation period, which is 24 hours. Upon joining the swarm, each peer is instructed to sequentially stream all video sequences. We also simulate dynamic replication of video segments as peers can upload segments as soon as they start downloading them. We consider a peer matching service that randomly provides each new peer up to 10 potential senders. Each new peer then connects to these senders, runs the scheduling algorithm, and requests segments following the computed schedule. Each receiving peer schedules segment transmission once every five seconds, and stop scheduling once all video sequences are finished.

The simulator determines each sender's upload bandwidth following the distribution given in Table 3. This bandwidth distribution is proposed in a recent paper [18] based on various measurement studies on both corporate and residential users. We note that peers would *not* contribute all their bandwidth to P2P streaming, because doing so would slow down other Internet applications such as email and Web. The *contributed bandwidth* of each class of peers is also given in this table as recommended in [18]. With the randomly chosen bandwidth, the simulator fairly distributes available bandwidth among all connections, and computes the transmission duration of each packet accordingly. Upon completely receiving each video sequence, the transmission statistics are written into a log file for further analysis. Finally, in the OPT and WSS algorithms, the system parameter $T$ is set to be 50, which means the time slots are 100 msec long in our formulation.

We run the simulator independently for each considered algorithm. We consider three performance metrics: the average perceived video quality $\alpha$, the continuity index $\beta$, and the load balancing factor $\gamma$. The average perceived video quality is computed by assuming that all late segments result in zero PSNR, and computing $\alpha = \sum_{n=1}^{N} w_n u_n / N$, where $w_n$ is the average perceived video quality of video frames in segment $n$. We define the continuity index as the number of frames that arrive by their decoding deadlines over the number of all considered frames. That is, $\beta = \sum_{n=1}^{N} u_n / N$. Last, we define the load of sender $m$ as its upload bandwidth utilization, which is $\left(\sum_{i \in \mathbf{Q}_m} s_i / \delta\right) / b_m$, where $\sum_{i \in \mathbf{Q}_m} s_i$ accounts for the size of all on-time segments and $\delta$ is the scheduling window length. The load balancing factor $\gamma$ is then computed as the standard deviation of all senders' loads. Similar performance metrics are used in other works in the literature, such as [11, 33].

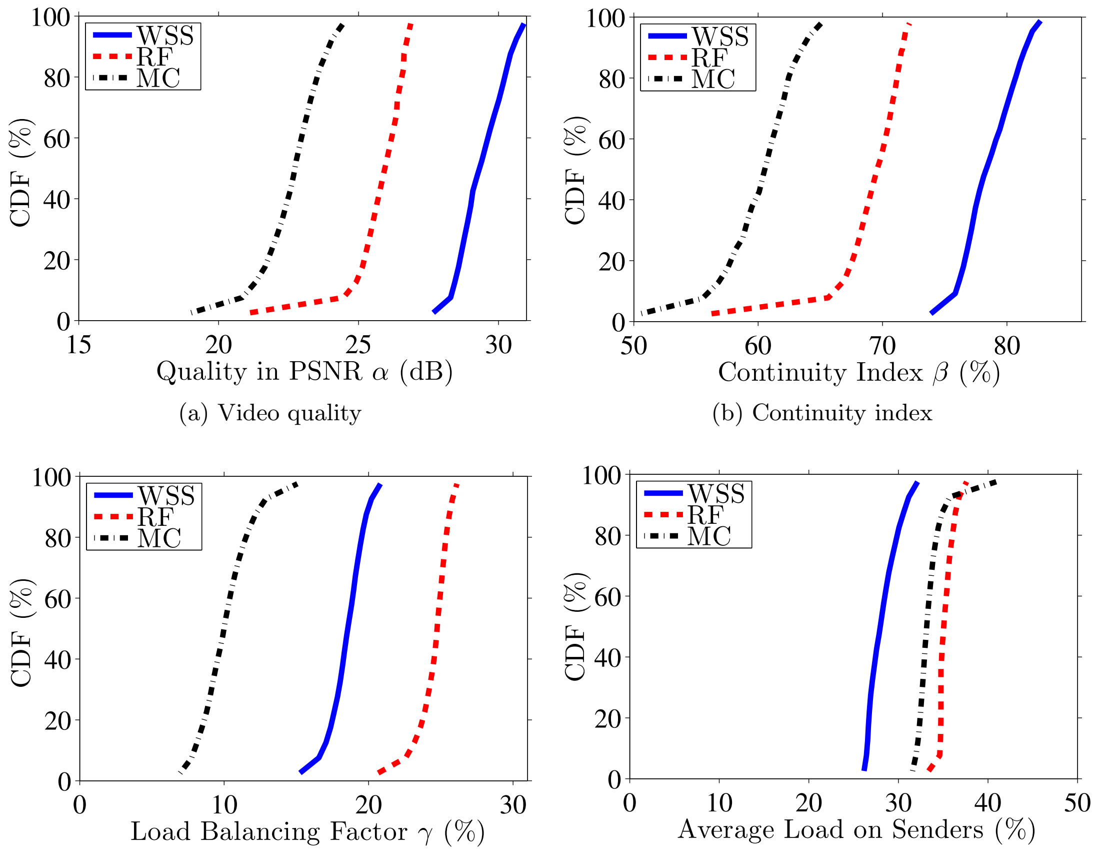
## 5.2 Comparison Against OPT

We first compare the performance of the WSS algorithm



(a) Video quality (b) Continuity index

**Figure 2: Comparison of the proposed (WSS) and optimal (OPT) algorithms.**



(a) Video quality (b) Continuity index

(c) Load balancing factor (d) Average load

**Figure 3: Overall comparison of the proposed (WSS) and heuristics algorithms used in current systems.**

against the absolute optimal (OPT) algorithm. We consider two performance metrics: video quality and continuity index. We calculate the average video quality and continuity index of each peer, and compute their CDFs (cumulative distribution functions). We plot the CDF curves of the video quality in Fig. 2(a). This figure shows that the WSS algorithm achieves almost optimal video quality: the gap is no larger than 0.5 dB for all peers. We also plot the CDF curves of the continuity index in Fig. 2(b). This figure illustrates that the WSS algorithm results in near optimal continuity index: at most 2% difference is observed. Fig. 2 reveals that, in practice, the WSS algorithm achieves near optimal performance: the gap is much smaller than the theoretical approximation factor of 3.

We measured the average running of the unoptimized Java implementation of our WSS algorithm across all runs on a commodity PC, which has a 2.66 GHz Intel CPU and 8 GB memory. The average running time was 178.47 msec, and the running time never exceeded 213.76 msec. We notice that, while we use CIF videos in the simulations the time complexity of our proposed algorithm is also efficient with high-definition videos. This is because the weights of individual segments are computed offline. Our results indicate that the proposed algorithm can easily run in real time. In contrast, computing a schedule using the OPT algorithm takes as long as several seconds.

### 5.3 Comparison Against Other Algorithms

We compare the performance achieved the WSS algorithm and by two heuristic algorithms: RF and MC in the following. We first conduct overall comparisons of all considered algorithms, and present the average performance over all nine video sequences with diverse characteristics. We then focus on a few sample video sequences and show detailed sequence-level comparisons among all the considered algorithms.

**Overall Comparison.** To derive statistically meaningful results, we calculate the average performance of each peer across all considered video sequences. We iterate through all peers and compute the CDF curves of each performance metric. We repeat the same computation for each scheduling algorithm, and we compare their CDF curves. We consider four performance metrics in our comparisons: video quality, continuity index, load balancing factor, and average load on senders.

We first plot the video quality in Fig. 3(a). This figure shows that the WSS algorithm outperforms the other two algorithms. The WSS algorithm achieves up to 5 dB and 8 dB higher perceived video quality compared to the RF and MC algorithm, respectively. These are substantial improvements in the rendered video quality. Our algorithm achieves these improvements because it carefully considers the quality contribution by each video segment in the scheduling process. It also results in more segments meeting their deadlines than the other two algorithms.

We then report the continuity index in Fig. 3(b). This figure illustrates that the WSS algorithm results in much higher continuity index than the RF and MC algorithms: up to 12% and 20% higher than the RF and MC algorithm, respectively. This means employing the WSS algorithm significantly reduces the playout glitches at receivers. These two figures clearly show that the proposed WSS algorithm results in much higher video streaming quality, compared to the heuristic algorithms used in current P2P streaming systems.

We next plot the load balancing factor in Fig. 3(c). The load balancing factor is defined as the standard deviation of the network loads on all senders. Excessive load balancing factor may slow down some senders' computers, which could discourage users from contributing to the P2P network. This figure illustrates that the WSS algorithm achieves at most 20% deviation, and is fair among loads of senders while it produces better video quality and higher continuity index as shown in Figs. 3(a) and 3(b).

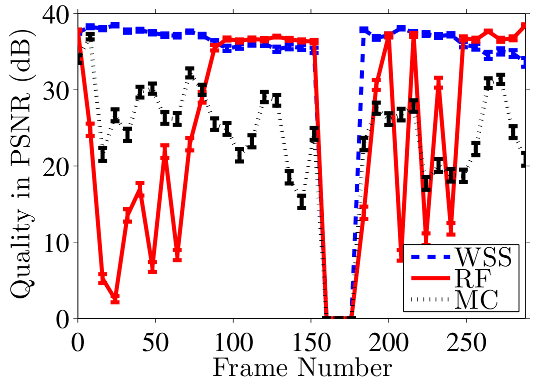
Last, we plot the average load on senders in Fig. 3(d). The average load on senders represents the typical network load that a subject algorithm incurs, and lower average load results in higher user satisfaction because more bandwidth can be used for other applications. This figure shows that the proposed WSS algorithm results in the lowest average load among all considered algorithms. Fig. 3 clearly shows that our WSS algorithm significantly outperforms other heuristic algorithms, because it produces the highest video quality (illustrated in Figs. 3(a) and 3(b)) while incurring the lowest load on peers (shown in Fig. 3(d)).

**Sequence-level Comparison.** Next, we present sample results on sequence-level comparisons. We consider two performance metrics in our comparisons: average video quality and number of late video frames in each GoP. For each video sequence, we iterate through all its video frames, and we compute the average perceived video quality in each GoP for every receiving peer. We then put all 2,000 peers together and compute the mean video quality as well as the 95% confidence interval. We report 95% confidence intervals in the figures. We use the same procedure to derive the number of late frames in each GoP. We repeat the computation for all considered video sequences, and for all scheduling algorithms.

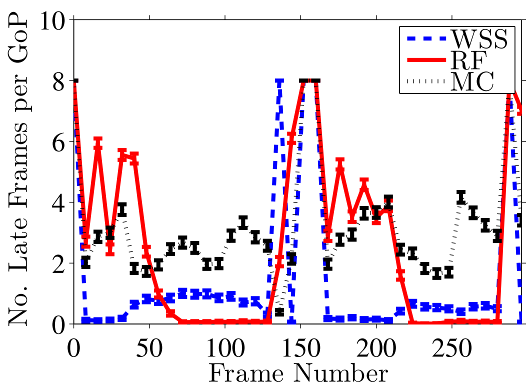
We present two sample results in Fig. 4; other results are similar. Fig. 4(a) shows the video quality of the Crew sequence produced by individual scheduling algorithms. This figure clearly shows that the WSS algorithm results in higher perceived video quality, compared to other heuristic algorithms. In fact, the WSS algorithm achieves video quality close to 38 dB, except between frames 160 and 176. In contrast, the RF and MC algorithms lead to too many quality drops, which degrade user experience. We next plot the number of late frames per GoP of the Soccer sequence in Fig. 4(b). This figure illustrates that the number of late frames produced by the WSS algorithm is much smaller than those of the RF and MC algorithms. More precisely, the WSS algorithm leads to no more than one lost frame per GoP most of the time. In summary, Fig. 4 zooms-in the performance of individual video sequences, and confirms that employing the WSS algorithm results in higher perceived video quality and fewer playout glitches.

## 6. CONCLUSIONS

We studied the segment transmission problem in P2P video streaming systems, where a receiver periodically computes a transmission schedule for all its senders to maximize the perceived video quality. We consider both live and on-demand P2P streaming systems. We proved that this problem is NP-Complete. We formulated the considered problem with an ILP formulation, and we solved it using the CPLEX [13] package. Optimally solving this ILP prob-



(a) Video quality, Crew



(b) Late frames, Soccer

**Figure 4: Sequence-level comparison of the proposed (WSS) and heuristics algorithms used in current systems.**

lem, however may take prohibitively long time, and is not suitable for P2P video streaming systems. We proposed an efficient approximation algorithm and we formally showed that it guarantees an approximation factor of 3 in the worst case. To the best of our knowledge, none of the existing polynomial time segment scheduling algorithms provide any performance guarantees.

We implemented an event-driven simulator and conducted extensive simulations to evaluate the proposed algorithm. The simulation results showed that the proposed algorithm: (i) is almost optimal with less than 0.5 dB gap in perceived video quality and less than 2% gap in continuity index from the optimum, (ii) is efficient with running time shorter than 213.76 msec, (iii) outperforms heuristic algorithms used in current systems by up to 8 dB in perceived video quality and up to 20% in continuity index, and (iv) does not impose imbalanced loads on senders.

In summary, the proposed algorithm not only provides analytical guarantees on the worst-case performance, but it also has superior average-case performance compared to other scheduling algorithms proposed in the literature and used in the deployed P2P streaming systems. Furthermore, our algorithm is computationally efficient (shown theoretically and empirically) and thus can be implemented in actual P2P streaming systems for both live and on-demand services.

## 7. REFERENCES

[1] Global IPTV market analysis (2006-2010). Technical report, RNCOS, August 2006. `http://www.rncos.com/Report/IM063.htm`.

[2] V. Agarwal and R. Rejaie. Adaptive multi-source streaming in heterogeneous peer-to-peer networks. In *Proc. of SPIE/ACM Multimedia Computing and Networking (MMCN'05)*, pages 13–25, San Jose, CA, January 2005.

[3] S. Annapureddy, S. Guha, C. Gkantsidis, D. Gunawardena, and P. Rodriguez. Is high-quality VoD feasible using P2P swarming? In *Proc. of International World Wide Web Conference (WWW'07)*, pages 903–912, Banff, Canada, May 2007.

[4] A. Bar-Noy, S. Guha, J. Naor, and B. Schieber. Approximating the throughput of multiple machines in real-time scheduling. *SIAM Journal on Computing*, 31(2):331–352, 2001.

[5] P. Brucker. *Scheduling Algorithms*. Springer, 4th edition, 2004.

[6] Y. Cai, A. Natarajan, and J. Wong. On scheduling of peer-to-peer video services. *IEEE Journal on Selected Area in Communications*, 25(1):140–145, January 2007.

[7] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. SplitStream: High-bandwidth multicast in cooperative environments. In *Proc. of ACM Symposium on Operating Systems Principles (SOSP'03)*, pages 298–313, Bolton Landing, NY, October 2003.

[8] J. Chakareski and P. Frossard. Utility-based packet scheduling in P2P mesh-based multicast. In *Proc. of SPIE International Conference on Visual Communication and Image Processing (VCIP'09)*, San Jose, CA, January 2009.

[9] Y. Chu, S. Rao, and H. Zhang. A case for end system multicast. In *Proc. ACM SIGMETRICS'00*, pages 1–12, Santa Clara, CA, June 2000.

[10] M. Goemans. Improved approximation algorithms for scheduling with release dates. In *Proc. of ACM-SIAM Symposium on Discrete Algorithms (SODA'96)*, pages 591–598, New Orleans, LA, January 1997.

[11] K. Graffi, S. Kaune, K. Pussep, A. Kovacevic, and R. Steinmetz. Load balancing for multimedia streaming in heterogeneous peer-to-peer systems. In *Proc. of ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'08)*, pages 99–104, Braunschweig, Germany, May 2008.

[12] X. Hei, C. Liang, J. Liang, Y. Liu, and K. Ross. A measurement study of a large-scale P2P IPTV system. *IEEE Transactions on Multimedia*, 9(8):405–414, December 2007.

[13] ILOG CPLEX. `http://www.ilog.com/products/cplex/`.

[14] H. Kellerer, T. Tautenhahn, and G. Woeginger. Approximability and nonapproximability results for minimizing total flow time on a single machine. In *Proc. of ACM Symposium on Theory of Computing*

*(STOC'96)*, pages 418–426, Philadelphia, PA, May 1996.

[15] J. Li and C. Yeo. Content and overlay-aware scheduling for peer-to-peer streaming in fluctuating networks. 32(4):901–912, July 2009.

[16] B. Liu, Y. Cui, B. Chang, B. Gotow, and Y. Xue. BitTube: case study of a web-based peer-assisted video-on-demand system. In *Proc. of IEEE International Symposium on Multimedia (ISM'08)*, pages 242–249, Berkeley, CA, December 2008.

[17] J. Liu, S. Rao, B. Li, and H. Zhang. Opportunities and challenges of peer-to-peer Internet video broadcast. *Proceedings of the IEEE*, 96(1):11–24, January 2008.

[18] Z. Liu, Y. Shen, K. Ross, J. Panwar, and Y. Wang. Substream trading: Towards an open P2P live streaming system. In *Proc. of IEEE International Conference on Network Protocols (ICNP'08)*, pages 94–103, Orlando, FL, October 2008.

[19] N. Magharei, R. Rejaie, and Y. Guo. Mesh or multiple-tree: A comparative study of live P2P streaming approaches. In *Proc. of IEEE INFOCOM'07*, pages 1424–1432, Anchorage, AK, May 2007.

[20] V. Padmanabhan, H. Wang, P. Chou, and K. Sripanidkulchai. Distributing streaming media content using cooperative networking. In *Proc. of ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'02)*, pages 177–186, Miami Beach, FL, May 2002.

[21] V. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy, and A. Mohr. Chainsaw: Eliminating trees from overlay multicast. In *Proc. of International Workshop on Peer-to-Peer Systems (IPTPS'05)*, pages 127–140, Ithaca, NY, February 2005.

[22] PPLive. `http://www.pplive.com/`.

[23] P. Rodriguez, S. Tan, and C. Gkantsidis. On the feasibility of commercial, legal P2P content distribution. *ACM SIGCOMM Computer Communication Review (CCR'06)*, 36(1):75–78, January 2006.

[24] H. Shojania, B. Li, and X. Wang. Nuclei: GPU-accelerated many-core network coding. In *Proc. of IEEE INFOCOM'09*, pages 459–467, Rio de Janeiro, Brazil, April 2009.

[25] SopCast. `http://www.sopcast.com/`.

[26] Y. Tu, J. Sun, M. Hefeeda, and S. Prabhakar. An analytical study of peer-to-peer media streaming systems. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 1(4):354–376, November 2005.

[27] TVAnts. `http://www.tvants.com/`.

[28] UUSee. `http://www.uusee.com/`.

[29] Y. Wang, J. Ostermann, and Y. Zhang. *Video Processing and Communications*. Prentice Hall, 2002.

[30] D. Xu, S. Kulkarni, C. Rosenberg, and H. Chai. Analysis of a CDN-P2P hybrid architecture for cost-effective streaming media distribution. *ACM/Springer Multimedia Systems Journal*, 11(4):383–399, April 2006.

[31] Y. Ye, editor. *Interior Point Algorithms: Theory and Analysis*. John Wiley and Sons, New York, NY, 1st edition, 1997.

[32] M. Zhang, Y. Xiong, Q. Zhang, L. Sun, and S. Yang. Optimizing the throughput of data-driven peer-to-peer streaming. *IEEE Transactions on Parallel and Distributed Systems*, 20(1):97–110, January 2009.

[33] X. Zhang, J. Liu, B. Li, and T. Yum. CoolStreaming/DONet: A data-driven overlay network for peer-to-peer live media streaming. In *Proc. of IEEE INFOCOM'05*, pages 2102–2111, Miami, FL, March 2005.