FORESIGHTED JOINT RESOURCE RECIPROCATION AND SCHEDULING STRATEGIES FOR REAL-TIME VIDEO STREAMING OVER PEER-TO-PEER NETWORKS

Sunghoon Ivan Lee, Hyunggon Park, and Mihaela van der Schaar

Electrical Engineering Department, UCLA
Email: silee@ucla.edu, {hgpark, mihaela}@ee.ucla.edu

ABSTRACT

We consider peer-to-peer (P2P) networks, where multiple heterogeneous and self-interested peers are sharing multimedia data. In this paper, we propose a novel scheduling algorithm for real-time video streaming over dynamic P2P networks. The proposed scheduling algorithm is *foresighted*, since it enables each peer to maximize its *long-term* video quality by efficiently utilizing its limited resources (e.g., uploading bandwidth) over time, while explicitly considering the time-varying resource reciprocation behaviors of its associated peers. To successfully design the scheduling algorithm, we consider a distinct buffer structure that allows the peers to model the resource reciprocation behavior as a reciprocation game. Then, each peer can determine its foresighted decisions based on a Markov Decision Process (MDP). The simulation results show that the proposed algorithm significantly improves the average video quality, compared to other existing scheduling strategies. Moreover, simulation results also show that the proposed algorithm can flexibly and effectively operate in heterogeneous P2P networks.

Index Terms— Peer-to-peer (P2P) networks, real-time video streaming, foresighted scheduling strategy, resource reciprocation game.

1. INTRODUCTION

Multimedia streaming over Internet, such as Video-on-Demand (VoD), broadcast of special events, remote education, etc. [1], has recently gained increased popularity. However, multimedia streaming over Internet does not scale easily with the number of users, as it requires the deployment of numerous servers, having considerable storage capacity [1] in order to support the Quality of Service. As an alternative solution to this challenge, P2P-based approaches have been recently proposed. A P2P system consists of multiple participants, referred to as peers, which act as both clients and servers [2]. The peers in P2P networks are interconnected with each other and share their resources such as multimedia content, available bandwidth, CPU cycles, storage, etc. [2]. Several approaches for general file sharing over P2P networks have been proposed [3–7], as well as for multimedia streaming and broadcasting over P2P networks (see e.g., [8–10]).

In this paper, we focus on developing a scheduling algorithm for real-time P2P video streaming, which considers how pre-encoded *segments* (or *chunks* [11]) of multimedia contents can be optimally downloaded. For example, a scheduling algorithm should be able to determine when or from which peer the segments should be downloaded. Unlike general file sharing algorithms deployed in several P2P applications such as BitTorrent [6], a scheduling algorithm for real-time video streaming over P2P networks must take into account the timing constraints of each segment (e.g., the tolerable maximum playback delay) [8], [11]. Moreover, the negotiation of downloading and uploading rates among the associated peers becomes a key issue, because the downloading rates of each peer must be greater or equal to the minimum required decoding rate. Finally, the order of downloading contents must be addressed, as the segment that is to be immediately displayed should have higher priority than the other segments.

To address these challenges, several scheduling algorithms for multimedia streaming applications over P2P networks have been proposed [8], [10], [12]. In [8] the algorithm decides a *supplying peer*[1] that contributes the highest bandwidth and that does not violate the playback delay. Alternatively, the scheduling algorithm in [10] deploys the rarest first search. While these scheduling algorithms provide simple and heuristic approaches, they do not consider the time-varying behaviors of supplying peers towards the resource reciprocation. Thus, these approaches provide only sub-optimal performances for continuously interacting peers. The interactions among the interested peers that have time-varying resource reciprocation behaviors are modeled as a resource reciprocation game, and each peer can make foresighted decisions on its resource distribution based on an MDP in the game [11]. However, the resource reciprocation strategy based on an MDP in [11] does not consider the time constraints such as the playback deadlines, which are critical for real-time video streaming applications.

[1] A supplying peer is defined as a peer delivering the content of interest. A set of supplying peers can be swarm in [6], [7], partnership in [8], or group in [11].

The main contribution of this paper is to introduce a scheduling algorithm for P2P real-time video streaming applications that explicitly considers the dynamic resource reciprocation behaviors of peers. The proposed algorithm also explicitly considers the time constraints (i.e., playback deadlines) of the video segments. To successfully implement the algorithm, we design a new buffer structure, which enables each peer to play the resource reciprocation game based on an MDP [11]. Hence, peers can make foresighted decisions that maximize their long-term video quality, while ensuring that segments in the buffer are continuously displayed in a timely manner.

This paper is organized as follows. In Section 2, we describe the proposed scheduling algorithm for real-time video streaming over P2P networks. In Section 3, we describe the resource reciprocation game played by peers based on an MDP. In Section 4, we show the experimental results about the proposed algorithm compared against other existing scheduling schemes. Finally, the conclusions are drawn in Section 5.

2. A FORESIGHTED SCHEDULING ALGORITHM FOR REAL-TIME STREAMING OVER P2P NETWORKS

2.1. Preliminaries: Scheduling algorithms

The goal of real-time multimedia streaming applications over P2P networks is to display multimedia contents in a timely manner, while maximizing the multimedia quality, by efficiently aggregating content from peers. To achieve this goal, peers in P2P streaming applications should deploy scheduling algorithms that determine the order of segment downloads, while efficiently allocating a fixed amount of resources (e.g., upload bandwidth). Segments in the buffer are already encoded and are provided by the original supplying peer[2]. An illustrative buffer snapshot for a multimedia streaming P2P system is shown in Fig. 1, based on [1].

In Fig. 1, the buffer is represented as a collection of segments, where multimedia data can be stored. For example, in [1], buffer consists of a total 120 segments, where each segment contains multimedia data of 1 second video. The shaded and empty segments in Fig. 1 represent chunks that are completely downloaded and that are scheduled to be downloaded in the future, respectively. The dotted line shows a snapshot of the buffer, which acts as a sliding window on the multimedia bit-stream. In the rest of Section 2, we propose a scheduling algorithm with a distinct buffer structure that enables the peers to make foresighted decisions, by considering the impact of their immediate actions on their long-term performance.

[2]The original supplying peer is referred to as the origin node in [8]. It is the original source of the video, which could be a server, or a peer distributing the video data.

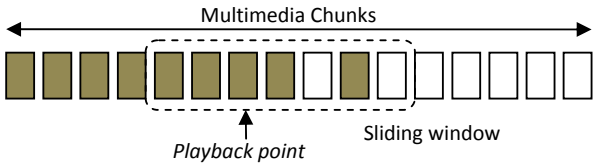

Fig. 1. Example of buffer snapshot. The dotted line represents the sliding window and the shaded segments represent completely downloaded segments.

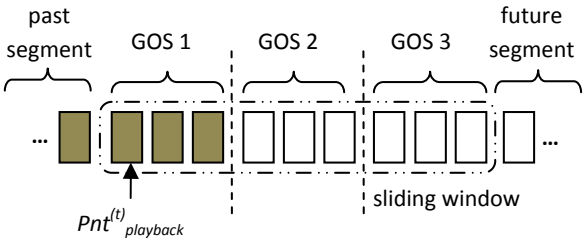

Fig. 2. An illustrative example of the proposed buffer structure.

2.2. Buffer structure for real-time P2P streaming

To successfully design the scheduling algorithm, we propose an alternative buffer structure. The proposed buffer structure enables peers to (i) play the resource reciprocation game (Section 2.4) and (ii) to deploy the system at the desired computational complexity and utility pair (Section 3.7). First, we divide the buffer into a finite number of Group of Segments (GOS). We denote the total number of segments in the buffer as N_{total} and the number of GOSs in a buffer as N_{GOS}. Thus, the number of segments in one GOS can be expressed as N_{seg} = N_{total} / N_{GOS}. For example, if there are 9 segments in the buffer and there are 3 GOSs, then one GOS has 3 segments, i.e., N_{seg} = N_{total} / N_{GOS} = 3. This is illustrated in Fig. 2.

In Fig. 2, playback point at time t is denoted by $Pnt^{(t)}_{playback}$. In the proposed buffer structure, the segments in a GOS should be completely downloaded when the GOS is located in the first GOS in the buffer (i.e., GOS 1). However, the other segments in the rest of GOSs are scheduled to be downloaded completely before their GOSs are positioned in the first GOS. For example, when a GOS newly enters the buffer, it is placed at the end of the buffer, e.g., position of GOS 3 in Fig. 2. As time t increases, the sliding window will move along the bit-stream, which includes the future segments in the buffer, and thus, the temporal distance between the new GOS and the playback point decreases. Finally, the new GOS will reach the playback point and the contents of that GOS will be displayed. The illustrative example is depicted in Fig. 3. In the next section, we discuss how we allocate the total available upload bandwidth to groups of supplying peers in different GOSs.

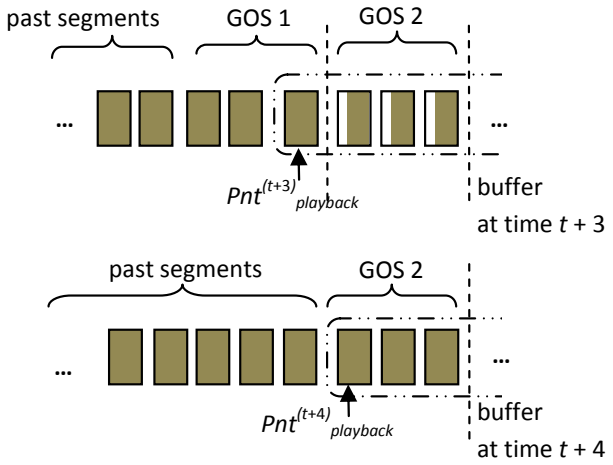

Fig. 3. Illustrative example showing that GOS 2 completes its downloading process as it reaches the playback point.

2.3. Bandwidth allocation to peers in GOS

Suppose BW_i denotes the total amount of available uploading bandwidth of a peer i. We equally divide the total uploading bandwidth and allocate them to the group of supplying peers in each of N_{GOS} GOSs in the buffer. Thus, we can express the amount of bandwidth distributed to the supplying peers in each GOS as,

$$BW_i^1 = \cdots = BW_i^{N_{GOS}} = \frac{BW_i}{N_{GOS}}, \tag{1}$$

where BW_i^k represents the uploading bandwidth of peer i allocated to supplying peers in GOS k. The amount of cumulated bandwidth that supplying peers can receive, until their GOS reaches the playback point, can be expressed as,

$$\sum_{k=1}^{N_{GOS}} BW_i^k = \sum_{k=1}^{N_{GOS}} \frac{BW_i}{N_{GOS}} = BW_i. \tag{2}$$

The above equation tells us that each GOS in the buffer enjoys the full amount of bandwidth that a peer i may provide. We may consider allocating different amount of bandwidth to supplying peers in different GOSs because a GOS temporally close to the playback point should have higher priority compared to the one that is far away. Suppose that different amount of bandwidth is distributed to supplying peers in different GOSs according to its temporal distance from the playback point. In this case, the bandwidth allocated to the first GOS will be the largest and the bandwidth allocated to ${N_{GOS}}^{th}$ GOS will be the smallest. Then, the bandwidth allocated to supplying peers in each GOS can be expressed as,

$$BW_i^k = \frac{\rho_k}{\sum_{m=1}^{N_{GOS}} \rho_m} \times BW_i,$$

where ρ_k represents the priority that GOS k holds, e.g. $\rho_k = 1/k$. As time goes by, the temporal distance between the playback point and GOS k will decrease and thus, the priority for GOS k will increase accordingly. The total amount of bandwidth that supplying peers in GOS k receive from peer i can be expressed as,

$$\begin{aligned}\sum_{k=1}^{N_{GOS}} BW_i^k &= \sum_{k=1}^{N_{GOS}} \left\{ \frac{\rho_k}{\sum_{m=1}^{N_{GOS}} \rho_m} \times BW_i \right\} \\ &= \frac{\sum_{k=1}^{N_{GOS}} \rho_k}{\sum_{m=1}^{N_{GOS}} \rho_m} \times BW_i \\ &= BW_i.\end{aligned}$$

Hence, the amount of the upload bandwidth from one GOS is constant regardless of prioritization. In this paper, we equally divide the total bandwidth of peer i and allocate it to supplying peers in each GOS. This bandwidth allocation scheme enables peers to deploy MDP, which will be discussed in detail in Section 3. In the next section, we discuss how we reciprocate the equally allocated bandwidth among supplying peers in each GOS.

2.4. Resource reciprocation in GOS

As discussed in Section 2.3, the total bandwidth of peer i is equally distributed to group of supplying peers in each GOS in the buffer. Then, the allocated bandwidth is also distributed among supplying peers of segments in each GOS. Note that the supplying peers dynamically respond to the bandwidth divisions of peer i, by allocating different amount of bandwidth to peer i. We term this interaction as "resource reciprocation" in this paper. The resource reciprocations between a peer and its associated supplying peers within each GOS are modeled as a resource reciprocation game, and each peer determines its optimal bandwidth allocation based on MDP, similar to [11]. The bandwidth allocation determined based on MDP allows each peer to optimally utilize the available upload bandwidth within each GOS, such that all segments in each GOS need to be downloaded within their playback deadlines. An illustrative example showing how MDP is deployed in each GOS is depicted in Fig. 4. The MDP-based bandwidth allocation of each peer in the resource reciprocation game is discussed in detail in Section 3.

Note that MPD needs to be solved (Section 3) only once when a GOS newly enters the buffer unless the supplying peers in that GOS leave the network or change their reciprocation behaviors before the GOS reaches the playback point. We study the impact of supplying peers' leaving before their GOSs reach the playback point, and quantify how the proposed scheduling algorithm can be optimized in such dynamic network conditions in Section 4.2.

2.5. Other issues for the proposed scheduling algorithm

In this section, we discuss several remaining consideration for implementing the proposed scheduling algorithm.

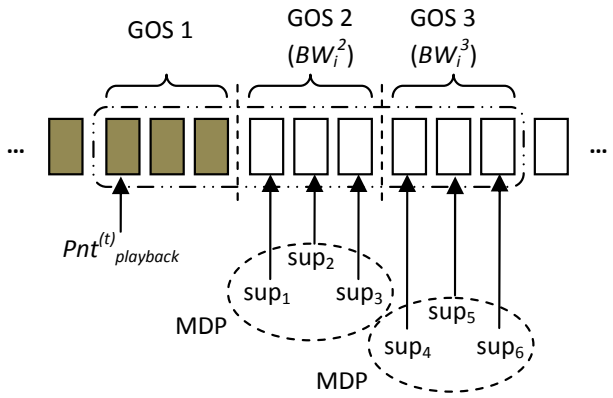

Fig. 4. Illustration of the buffer snapshot, where BW_i^k represents the allocated bandwidth to GOS k and sup_m represent the supplying peer m associated with its corresponding segment. In the proposed buffer structure, $BW_i^1 = BW_i^2 = BW_i^3$.

2.5.1. Residual bandwidth

In the proposed scheduling algorithm, the residual bandwidths are created in the buffer as the playback point moves toward the last segment in the first GOS. An illustrative example is depicted in Fig. 5. In the proposed approach, the residual bandwidth can be used to estimate the resource reciprocation behavior of supplying peers to be included in the new GOS. We deploy the resource reciprocation models proposed in [11] in order to efficiently analyze the resource reciprocation behaviors of supplying peers.

2.5.2. Past segments

The segments already passed the playback point are not useful to display a video content. However, higher chance that other peers may be interested in downloading the segments. Therefore, a peer would like to keep the past segments. We know that time lags between playback points of associated peers are usually less than 1 minute [8].

2.5.3. Incompletely filled buffer

The performance of the proposed algorithm can be maximized if all segments in the buffer are filled. Thus, when a peer newly enters the network and starts to reciprocate its resources, it can initially employ a simple heuristic scheduling algorithm such as that in [8], until the buffer is filled up. After the peer finds supplying peers for all segments, it will run the proposed scheduling algorithm. Note that the duration of this initial phase is very short compared to the overall running time of the system and thus, the impact of this heuristic algorithm on the overall performance is minimal.

The proposed scheduling algorithm is summarized in Fig. 6.

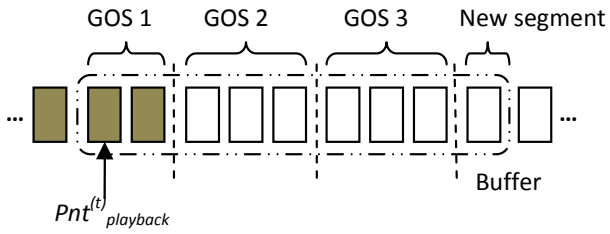

Fig. 5. Illustration of residual bandwidth created in the buffer.

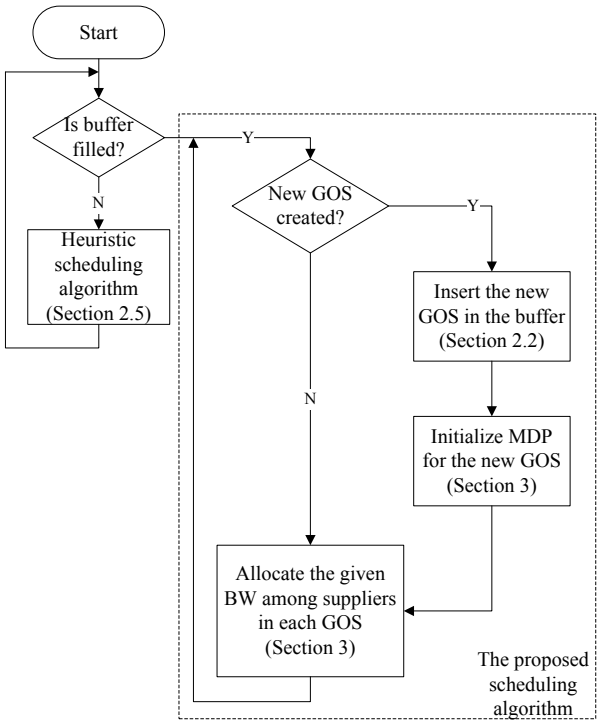

Fig. 6. The proposed scheduling algorithm for real-time video streaming over P2P networks.

3. RESOURCE RECIPROCATION GAME IN GOS

In this section, we model the resource reciprocations between a peer and its supplying peers within each GOS as a resource reciprocation game. The group of supplying peers of peer i in its GOS k is denoted by C_i^k. It can be easily computed that the total number of supplying peers in C_i^k is equal to the total number of segments in one GOS, e.g., $N_{C_i^k} = N_{seg}$. The group of all supplying peers in the buffer is denoted by C_i and it can be expressed as,

$$C_i = \bigcup_{k=1}^{N_{GOS}} C_i^k.$$

Note that every supplying peer j in C_i has also its own group of peers (i.e., supplying peers) C_j. In this paper, we assume that one segment is downloaded only by a distinct peer in the

buffer. However, several segments can also be downloaded from one peer simultaneously, which leads to the resource reciprocation behavior changes of the peer. In this case, the scheduling algorithm needs to update the state transition probability and the corresponding optimal policy for this peer.

We now model the resource reciprocation game within one GOS played by a peer and its supplying peers [11]. Since the same amount of the bandwidth is allocated to group of supplying peers in each GOS, all GOSs in the buffer will employ the identical method for the resource reciprocation strategy. Every peer participating in resource reciprocation game employs MDP, which is a 4-tuple $\langle \mathbf{S}_i, \mathbf{A}_i, P_i, R_i \rangle$. $\mathbf{S}_i$ denotes the state space of peer i. $\mathbf{A}_i$ represents the action space which contains sets of actions that a peer may take in the game. P_i is state transition probability of peer i. Suppose that a peer is at a state $s_i^{(t)} \in \mathbf{S}_i$ and it takes an action $a_i^{(t)} \in \mathbf{A}_i$ at time t. Then, the probability that the peer will be at the next state $s_i^{(t+1)} \in \mathbf{S}_i$ at time $t+1$ can be found based on the state transition probability, $P_i = \Pr\left\{s_i^{(t+1)} | s_i^{(t)}, a_i^{(t)}\right\}$. $R_i : \mathbf{S}_i \rightarrow \Re$ represents the reward function that maps the benefit of being in a certain state into a real number.

3.1. State space

The state of a peer i is defined as a set of bandwidths (e.g. the download rate) that each supplying peer in a GOS provides to peer i. The set of bandwidths received can be expressed as,

$$\Big\{(x_{1,i}^k, \cdots, x_{N_{seg},i}^k) | 1 \leq x_{j,i}^k \leq BW_j \ , \\ j \in C_i^k, 1 \leq k \leq N_{GOS}\Big\}, \quad (3)$$

where $x_{j,i}^k$ represents the provided bandwidth by peer j in C_i^k to peer i, and BW_j is the maximum upload bandwidth of peer j. In this paper, we assume that $x_{j,i}^k$ is quantized by peer i using a function $f_{i,j}(\cdot)$, which results in a quantized discrete value. For simplicity, we assume that $f_{i,j}(\cdot)$ is a uniform quantization function, i.e.,

$$f_{i,j}\left(x_{j,i}\right) = s_{i,j}^l \ \text{ if } (l-1) \cdot \frac{BW_j}{n_{i,j}} \leq x_{j,i} < l \cdot \frac{BW_j}{n_{i,j}}, \quad (4)$$

where $s_{i,j}^l$[3] is a representative output value of the l^{th} bin, and $n_{i,j}$ is the total number of bins. The state of peer i in GOS k can be correspondingly defined as,

$$\mathbf{S}_i^k = \Big\{s_i^k = (s_{i,1}^k, \cdots, s_{i,N_{seg}}^k) | s_{i,j}^k = f_{i,j}(x_{j,i}^k) \ , \\ j \in C_i^k, 1 \leq k \leq N_{GOS}\Big\}. \quad (5)$$

Therefore, the state of a peer i of GOS k can be described as a set of quantized upload bandwidth that each supplying peer in GOS k provides. Note that, throughout this paper, we sometimes omit the subscription for the GOS number in the notation of the state for simplicity, since identical reciprocation strategies are employed in all GOSs.

[3]Note that l in this notation does not represent the GOS number as in (5)

3.2. Action space

An action of a peer i of GOS k can be defined as a set of bandwidth divisions that peer i contributes to each supplying peer in GOS k. Thus, the action space of peer i in GOS k can be expressed as,

$$\mathbf{A}_i^k = \Big\{a_i^k = (a_{i,1}^k, \cdots, a_{i,N_{seg}}^k) | 0 \leq a_{i,j}^k \leq BW_i^k \ , \\ j \in C_i^k, \sum\nolimits_{j \in C_i^k} a_{i,j}^k = BW_i^k\Big\}, \quad (6)$$

where $a_{i,j}^k \in \mathbf{A}_i^k$ represents the allocated bandwidth from peer i to its associated supplying peer j in GOS k and BW_i^k denotes the maximum total available bandwidth of peer i in GOS k. The total bandwidth of i within GOS k, BW_i^k, is distributed to its supplying peers in the GOS. Therefore,

$$\sum_{j \in C_i^k} BW_{i,j}^k = BW_i^k. \quad (7)$$

We can now describe the resource reciprocation within GOS k using a pair, $(a_i^k, s_i^k) = \Big((a_{i1}^k, \cdots a_{iN_{seg}}^k), (s_{i1}^k, \cdots, s_{iN_{seg}}^k)\Big)$ where $a_{i,j}^k$ is the action of peer i in GOS k and $s_{i,j}^k$ is the quantized resource obtained from supplying peer j in GOS k, which is determined by $f_{i,j}(x_{j,i}^k)$. For example, let us consider a situation where peer i is in a state $s_i^{(t)}$ at time t and the peer takes an action $a_{i,j}^k \in \mathbf{A}_i^k$. By taking an action a_i^k, peer i allocates a certain amount of bandwidth to each supplying peer j in C_i^k. This action affects the states of all supplying peers in C_i^k and they will response with a new set of bandwidths, $x_{j,i}^k$, back to peer i This chain reaction will eventually lead peer i into a new state, $s_i^{(t+1)}$, at time $t+1$. An illustrative explanation about the resource reciprocation is provided in Fig. 7.

3.3. State transition probability

State transition probability (STP) is defined as a set of probabilities which specifies how a peer would evolve from one state to another state (within a GOS) given that the peer takes a certain action. Suppose that a peer i is in state $s_i^{(t)}$ within GOS k at time t, and it takes an action a_i^k. Then, peer i can transit to another state $s_i^{(t+1)}$ at time $t+1$ with probability[4], $P_{a_i^k}(s_i^{(t)}, s_i^{(t+1)}) = \Pr(s_i^{(t+1)} | s_i^{(t)}, a_i^k)$. We also have an action, which is a set of bandwidth divisions to each supplying peer in GOS k, $a_i^{(t)} = (a_{i,1}^{(t)}, \cdots, a_{i,N_{seg}}^{(t)})$. In order to calculate

[4]Note that, for simplicity, we omit the subscription for GOS number in the notation of the state. The subscription indicating time, (t), is used instead.

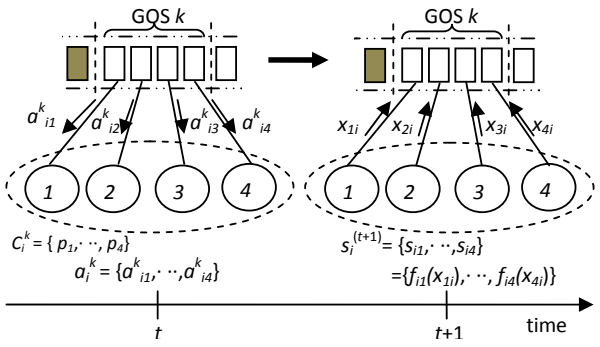

Fig. 7. An example of reciprocation game between peer i and its associated supplying peers, 1, 2, 3, and 4. This illustration shows how the state evolve from t to $t+1$ depending on peer i's action at time t and the reaction from each supplying peer.

the STP, $P_{a_i^k}(s_i^{(t)}, s_i^{(t+1)})$, we first compute the STP for each supplying peer in GOS k, denoted as $P_{a_{i,j}^k}(s_{i,j}^{(t)}, s_{i,j}^{(t+1)}) = \Pr(s_{i,j}^{(t+1)}|s_{i,j}^{(t)}, a_{i,j}^k)$, for each component state, $s_{i,j}^{(t)} \in s_i^{(t)}$. Then, the STP of peer i in GOS k can be expressed as,

$$P_{a_i^k}(s_i^{(t)}, s_i^{(t+1)}) = \prod\nolimits_{j \in C_i^k} P_{a_{i,j}^k}(s_{i,j}^{(t)}, s_{i,j}^{(t+1)}). \quad (8)$$

The above equation is valid since we assume that STP for each supplying peer in a GOS is independent from each other as it is suggested in [11]. The STP for supplying peer j in GOS k, $P_{a_i^k}(s_i^{(t)}, s_i^{(t+1)})$, can be estimated based on supplying peers' behaviors in the past using an algorithm proposed in [11].

3.4. Reward

We define the utility of peer i achieved from GOS k as,

$$U_i^k = \sum_{j \in C_i^k} U_{i,j}^k(x_{j,i}), \quad (9)$$

where $U_{i,j}^k$ is the utility of peer i achieves from the segment that is downloaded from supplying peer j in GOS k at the rate $x_{j,i}$. $U_{i,j}^k$ can be defined as,

$$U_{i,j}^k = \begin{cases} 0, \text{ if } x_{j,i} < BW_{i,j}^*, \\ \rho_{i,j} \cdot Q_i(x_{j,i}), \text{ otherwise} \end{cases}, \quad (10)$$

where $\rho_{i,j}$ is the constant representing the preference that peer i has on segment associated with peer j. Note that having $\rho_{i,j}$ in (10) allows the proposed scheduling algorithm to work with scalable video coders where each segment or packet has different impact on the output video quality. $Q_i(x_{j,i})$ is the derived quality of the downloading rate $x_{j,i}$, which can be represented in PSNR [13]. $BW_{i,j}^*$ represents the theoretical constant bandwidth (i.e. rate) that j must provide in order to complete the download within the playback deadline. It can also be defined as the minimum download rate required in order to meet the minimum required utility of segment obtained from supplying peer j. We can compute the value of $BW_{i,j}^*$ as follows. Suppose that a new GOS enters the buffer. If we assume that each segment contains data of t_{seg} second video (i.e. $t_{seg} = 1$ second in [8]), we can calculate the playback deadline from the time that a GOS newly enters the buffer as,

$$t_{deadline} = (N_{GOS} - 1) \times N_{seg} \times t_{seg}, \quad (11)$$

where N_{GOS} is the number of GOSs in the buffer and N_{seg} is the number of segments (or number of supplying peers) in each GOS. We also define B_j as the size (bits) of each segment that supplying peer j must transmit. Then, we calculate the theoretical constant bandwidth (i.e. rate) that j must maintain as, $BW_{i,j}^* = B_j / t_{deadline}$.

The utility of peer i achieved from GOS k, U_i^k, is a non-decreasing function of the x_i which is the sum of download rate from supplying peers in GOS k, e.g., $x_i = \sum_{j \in C_i^k} x_{j,i}$. Thus, we defined the reward of peer i within GOS k, $R_i^k(s_i^k)$, as the total resource received from supplying peers. Since a state of peer i is a collection of quantized value of the received resource from supplying peers in GOS k, we define the reward of a state to be a random variable. The reward can be mathematically represented as,

$$R_i^k(s_i^k) = R_i(s_{i,1}^k, \cdots, s_{i,N_{seg}}^k) = \sum\nolimits_{j \in C_i^k} \rho_{i,j} \cdot r_{i,j}(s_{i,j}^k), \quad (12)$$

where $r_{i,j}(s_{i,j}^k)$ is a random variable representing the received resources from peer j, e.g., $x_{j,i}$. Thus, the resulting utility of peer i within GOS k can be expressed as, $U_i^k = \sum_{j \in C_i^k} U_{i,j}^k \left(r_{i,j}(s_{i,j}^k)\right)$.

3.5. Optimal resource reciprocation policy

A resource reciprocation policy determines an action that a peer can take when it's in a certain state. The optimal resource reciprocation strategy determined by MDP enables each peer to maximize its long-term utility. Specifically, optimal policy π_i^* of peer i for GOS k determines the optimal action $\pi_i^*(s_i) = a_i^*$ for every state $s_i \in \mathbf{S}_i$, such that the optimal actions maximize not only the immediate expected reward, but also the future rewards. The cumulated expected reward (i.e., immediate expected reward and the future rewards) for a peer i that is in a state $s_i^{(t)} = (s_{i1}, \cdots, s_{iN_{seg}})$ at time $t = t_c$ can be express as [11],

$$R_i^{fore}(s_i^{(t_c)}) \equiv \sum_{s_i^{(t_c+1)} \in S_i} P_{a_i}(s_i^{(t_c)}, s_i^{(t_c+1)}) R_i(s_i^{(t_c+1)}) + \sum_{t'=t_c+1}^{\infty} \gamma_i^{(t'-t_c)} \sum_{s_i^{(t'+1)} \in S_i} P_{a_i^{(t'+1)}}(s_i^{(t')}, s_i^{(t'+1)}) R_i(s_i^{(t'+1)}), \quad (13)$$

where γ_i is a discount factor, which is introduced in order to provide different weights between the immediate and the

future rewards, because the immediate reward is worth more than the future rewards. The optimal actions that maximize $R_i^{fore}(s_i^{(t_c)})$ in (13) are referred to as *foresighted* actions, i.e.,

$$a_i^* = \max_{a_i \in A_i} \arg \left\{ \sum\nolimits_{s_i^{(t+1)} \in S_i} P_{a_i}(s_i^{(t)}, s_i^{(t+1)}) R_i(s_i^{(t+1)}) \right\}.$$

Moreover, the optimal policy can be always found based on well-known algorithms such as value iteration or policy iteration [14].

Using our buffer structure, the optimal policy computed by MDP is performed multiple times, while supplying peers perform the download in the buffer (i.e., from the time a GOS enters the buffer until it meets the playback point). Thereby, it is important to maximize both the immediate and the future rewards because a GOS stays in the buffer for a long-term. As a result, output video quality achieved by segments in a GOS will be improved as we perform foresighted actions. Simulation results for the impact of foresighted actions are provided in Section 4.1.

The performance of the overall scheduling algorithm may also vary depending on the value of the discount factor, γ_i. We can estimate the appropriate value for the discount factor by predicting the number of times that MDP will update the state of i in a GOS. We previously calculated the playback deadline for all segments in a GOS, i.e., $t_{deadline}$ in (11). We can realize that $t_{deadline}$ is actually equal to the amount of time that GOS stays in the buffer before it reaches the playback point. For example, in [8], the buffer is assumed to consist 120 segments, and each segment contains data of 1 second video. Hence, $t_{deadline}$ = 120 second in this case. Suppose that the MDP updates the state of i in GOS (i.e., perform the optimal action) every t_{update} seconds. Then the total number of updates of the state of i is given by,

$$N_{update} = \left\lfloor \frac{t_{deadline}}{t_{update}} \right\rfloor.$$

We can now choose a value of γ_i such that the future reward after N_{update} is negligibly small, i.e.,

$$\gamma_i^{(N_{update}+1)} \leq \gamma_{thresh}, \tag{14}$$

where γ_{thresh} is a threshold value to represent the level of accuracy. Simulation results using different discount values will be provided in Section 4.2.

3.6. Complexity analysis for optimal resource reciprocation policy

The optimal resource reciprocation policy can be computed based on the well-known value iteration algorithm, and it is shown that this algorithm requires $O(|\mathbf{A}||\mathbf{S}|^2)$complexity for each iteration [15], where $|\mathbf{A}|$ and $|\mathbf{S}|$ denote the number of actions and the number of states. Recall that, in the proposed scheduling algorithm, N_{seg} segments are included in a GOS k , and the segments can be downloaded from different supplying peers. Therefore, the complexity required to solve an MDP in GOS k can be expressed as,

$$O\left(\left|\mathbf{A}_i^k\right|^{N_{seg}} \cdot \left(\prod_{j \in C_i^k} n_{i,j} \right)^2 \right), \tag{15}$$

where $n_{i,j}$ is the number of the state description as in (4). As shown in [15], the maximum number of iterations required to find the optimal policy is determined by several pre-determined parameters such as the number of bits used to describe each state, the discount factor, etc. Hence, the total complexity required to find the optimal policy is primarily determined by the number of actions, the number of states, and the number of segments in a GOS as shown in (15). Therefore, the complexity analysis shown in (15) provides a guideline how peer i can flexibly adjust the number of its actions, states, and the segments in a GOS, while explicitly considering its available computational capacity, available time to find the policy, etc. For example, if the computation complexity is a concern, peer i can adaptively determine the number of states by adjusting $f_{i,j}(\cdot)$ in the proposed algorithm. Alternatively, the number of segments N_{seg} can also be determined. Hence, each peer can explicitly consider the tradeoffs between the complexity required to implement the proposed approach and the corresponding performance improvement.

3.7. Determining the number of segments in a GOS

In this section, we consider the tradeoffs between computational complexity and utility. Thus, a peer can determine the number of GOSs in its buffer and the number of segments in each GOS, thereby deploying its scheduling algorithm at the desired complexity and utility pair. For instance, we consider a peer i associated with a finite number of associated peers. We label the action of peer i as $a_i = (a_{i,1}, \cdots, a_{i,N_{total}})$ as in (6). As an illustrative example, we assume that associated peers of peer i reciprocate resources linearly with respect to the action of peer i (i.e., $x_{j,i} = \alpha_{i,j} \cdot a_{i,j}$) for all $j \in C_i$. We also assume that peer i has its optimal action, $a_{i,j}^*$, which maximizes the utility (e.g. output video quality in PSNR) of peer i. The range of the optimal bandwidth allocated from peer i to peer j is $0 \leq a_{i,j}^* \leq BW_i$. When the buffer is divided into GOSs, the range of the bandwidth that peer i can allocate to peer j changes according to (6). We denote the actual allowable bandwidth from peer i to peer j with considering the buffer structure as $a_{i,j}$, and the range of $a_{i,j}$ can be defined as $0 \leq a_{i,j} \leq BW_i/N_{GOS}$. Since we assume that peer j provides its bandwidth linearly proportional to $a_{i,j}$, the reduction in the bandwidth provided from peer j due to

the difference between $a^*_{i,j}$ and $a_{i,j}$ can be expressed as

$$\Delta x_{j,i} = \begin{cases} \alpha_{i,j}\left(a^*_{i,j} - \frac{BW_i}{N_{GOS}}\right), \text{ if } \frac{BW_i}{N_{GOS}} \le a^*_{i,j} \le BW_i \\ 0, \text{ if } 0 \le a^*_{i,j} \le \frac{BW_i}{N_{GOS}} \end{cases}.$$

We can compute the expected value of $\Delta x_{j,i}$ under an assumption that $a^*_{i,j}$ has a uniform probability distribution over its range $[0, BW_i]$ (i.e., $f_{a^*_{i,j}} = 1/BW_i$) as

$$\begin{aligned} E\{\Delta x_{j,i}\} &= \int_0^{BW_i} f_{a^*_{i,j}}(a) \cdot \Delta x_{j,i} da \\ &= \int_0^{BW_i} \frac{1}{BW_i} \cdot \alpha_{i,j}(a - a_{i,j})\, da \\ &= \int_{BW_i/N_{GOS}}^{BW_i} \frac{1}{BW_i} \cdot \alpha_{i,j}\left(a - \frac{BW_i}{N_{GOS}}\right) da \\ &= \frac{1}{2}\alpha_{i,j} BW_i \left(\frac{N_{GOS}-1}{N_{GOS}}\right). \end{aligned}$$

Using the distortion-rate based utility function [16] for the state-of-the-art video coders [17], we can compute the expected distortion for a fixed N_{GOS} based on $E\{\triangle x_{j,i}\}$. Since $E\{\triangle x_{j,i}\}$ is an increasing function of N_{GOS}, the distortion function will also be an increasing function of N_{GOS} and thus, the utility function of peer i will be a decreasing function of N_{GOS}.

Fig. 8 illustrates an example of the utility function of peer i with a buffer of 12 segment. We express the utility function of peer i in PSNR and the testing video sequence is a *Foreman* sequence with CIF resolution, temporal level of 4, and the frame rate of 30 Hz under the assumption that the rate $x_{j,i} = \alpha_{i,j} \cdot a^*_{i,j}$ gives the best performance (i.e., $\alpha_{i,j} \cdot BW_i =$ 250Kbps).

The computational complexity can also be computed as a function of N_{GOS} according to (15), since $N_{C_i^k} = N_{total}/N_{GOS}$. Thus, a user can choose the number of GOSs in the buffer according to the desired utility and complexity pair.

4. SIMULATION RESULTS

4.1. Comparison of various scheduling algorithms on PSNR performance

As discussed in Section 2 and 3, the buffer can be divided into several GOSs and the MDP-based foresighted decisions can efficiently allocate the available bandwidth. In this section, we compare the multimedia quality achieved based on the proposed scheduling algorithm as well as the other existing scheduling strategies - Tit-for-Tat (TFT) strategy in BitTorrent system [6], [7] and BiToS systems [12].

A peer with the TFT strategy selects a fixed number of leechers (supplying peers) that contribute the highest bandwidths to the peer, and reciprocates the equally divided bandwidth to the selected leechers. The TFT is deployed on the entire buffer that is not divided into GOSs. We assume that the buffer only serves the downloading processes and also assume a fixed set of supplying peers in the buffer.

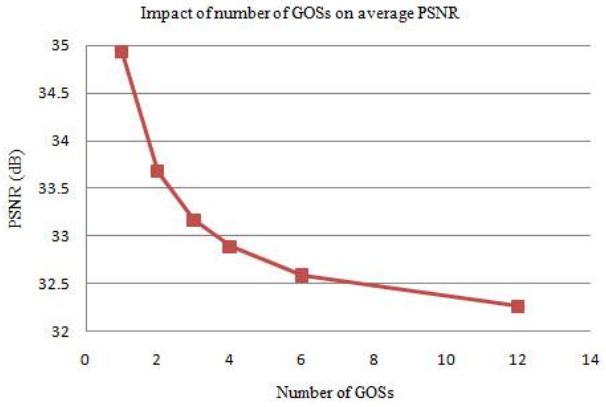

Fig. 8. An illustrative example of achieved PSNR value at different N_{GOS}.

The BiToS system is a variant of the BitTorrent system, which can support streaming applications [12]. In this system, the scheduling algorithm categorizes the segments in the buffer into two groups: *high priority group* and *remaining group*. High priority group contains segments that have not been downloaded and are close to the playback deadline. The remaining group contains segments that has not been downloaded and that are not a member of the high priority group. This algorithm has a parameter p which represents the probability that a segment in the high priority group is selected to be downloaded. Thus, this scheduling algorithm explicitly considers the real-time constraints by focusing on downloading segments that are closer to the playback deadline.

In this simulation, the buffer of peer i may contain at most 18 segments and the multimedia bit-streams of 50 segments are provided to the system in order to emulate the real-time environment. For simplicity, we assume that each segment is separated by I-frames, thereby having its own video quality. We also assume that each supplying peer can provide maximum allowable bandwidth of 125 Kbps. For the simulation, we deploy the distortion-rate based utility function for the state-of-the-art video coders [17], which was introduced in [16]. Peer i downloads *Foreman* video sequence with CIF resolution and the frame rate of 30 frames/second. The video size is 1.6 Mbytes. The simulation results are shown in Fig. 9.

In the simulations, the number of the associated peers is 18, which is same as the number of segments in the buffer. The number of slots for downloading is 4, and thus, each peer equally divides its available upload bandwidth and allocates it to 4 different supplying peers that contribute the highest bandwidth. Each peer has 10 available actions and 4 state descriptions, i.e., $a_{i,j} \in \{a^1_{i,j}, a^2_{i,j}, \cdots, a^{10}_{i,j}\}$ and $s_{i,j} \in \{s^1_{i,j}, s^2_{i,j}, s^3_{i,j}, s^4_{i,j}\}$, respectively. For BiTos system, the high priority group contains 4 segments that are closest to the play-

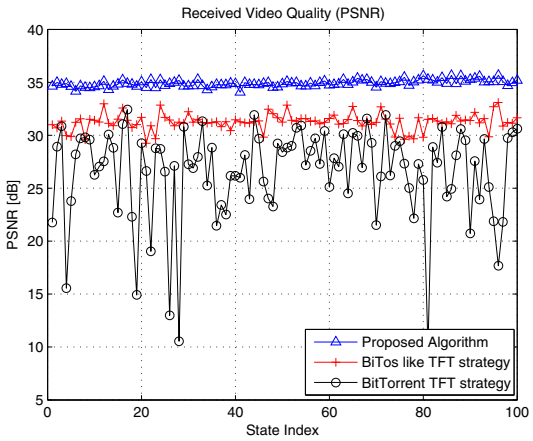

Fig. 9. An illustrative example for video quality in PSNR, achieved by different scheduling algorithms.

back deadline and the value of p is set to 1. Thus, the second test-bench system equally divides the total bandwidth and provide them to peers associated with the 4 segments that are about to be displayed. In the proposed algorithm, the buffer is divided into GOSs, where each GOS contains 3 segments. Hence, the number of GOSs in the buffer is $18/3 = 6$. However, because we assume 10 action descriptions, having 6 GOSs is not reasonable to show the result of our foresighted scheduling algorithm. Thus, in this simulation, the proposed scheduling algorithm performs the foresighted scheduling for the 6 segments closest to the playback deadline. A state of peer i is represented as $s_i = (s_{i,1}, s_{i,2}, \cdots, s_{i,6})$ but it is divided into 2 GOSs. Therefore, we have two sub-states for each GOS represented as $s_i^k = (s_{i,1}^k, s_{i,2}^k, s_{i,3}^k)$ for $1 \leq k \leq N_{GOS} = 2$. The total number of possible state of peer i can be calculated as $4^6 = 4096$, and the simulation is performed for all possible 4096 initial state of peer i. The bandwidth allocated to each GOS, denoted as BW_i^k, is equal to BW_i/N_{GOS} as in (2). The discount factor is set to 0.8, i.e., $\gamma_i = 0.8$.

The simulation results in Fig. 9 clearly show that the proposed foresighted scheduling algorithm outperforms the heuristic TFT strategies. State index in Fig. 9 represents an initial state of a peer. In Fig. 9, in order to help visualize the results, we only display a randomly selected portion of the initial state indices, i.e., from state index 1 to 100. The overall simulation results over the entire state indices are similar to what we have in Fig. 9. On average, the proposed algorithm improves 3.61 dB over the BiToS system and 5.33 dB over the BitTorrent-like TFT strategy across all state indices.

4.2. Impact of discount factor on heterogeneous network conditions

In this section, we quantitatively compare the impact of the discount factor on the video quality in heterogeneous P2P networks. In our considered P2P real-time streaming scenario, the size of the buffer is fixed, and thus, the optimal policy should not maximize the reward cumulated up to infinitely far away in the future. We can calculate the value of the discount factor based on (14) using information about the buffer size. This allows the scheduling algorithm to consider the future reward limited to the size of the buffer. However, when the network is heterogeneous and the supplying peers in GOS leave the network before the GOS reaches the playback point, the discount factor computed based on the buffer size is not optimal anymore. When one or more of the supplying peers leave the network, the scheduling algorithm must find new supplying peers and compute the optimal policy based on the reciprocation attitude of the new supplying peers as discusses in Section 3. In this case, if we assume that we know the average number of supplying peers' leaving as a GOS performs the download process in the buffer, we can compute the value of the optimal discount factor based on the average time duration that a GOS remains in the buffer without changes in supplying peers' attitudes. For example, let us assume that the scheduler performs the optimal policy 20 times per GOS until the GOS reaches the playback point. We also assume that, on average, reciprocation behaviors of supplying peers change once as a GOS progresses in the buffer. In other words, the optimal action is performed 10 times on average before the reciprocation behaviors change. In this scenario, the scheduler can compute the discount factor based on the average number of optimal actions to be performed without changes in supplying peers' attitudes, e.g. 10 times. In the example above, the discount factor computed based on the buffer size is $\gamma = 0.7943$ and the discount factor based on the average number of optimal actions performed is $\gamma = 0.6310$. In Fig. 10, the parameter settings, such as the number of associate peers and the buffer sizes are identical to the simulation for the proposed algorithm performed in Section 4.1, except that different values of the discount factors are considered. We display the multimedia video quality over a randomly selected portion of the state indices, i.e., from state index 400 to 500.

The results in Fig. 10 show that the system deploying the optimal discount factor based on the average number of action perform without changes in supplying peers' reciprocation behaviors gives higher and more stabilized PSNR performance compared to the system with discount factor computed based on the buffer size. On average, the system with the optimal discount factor shows advantages of 0.44 dB over the system with discount factor computed based on the buffer size.

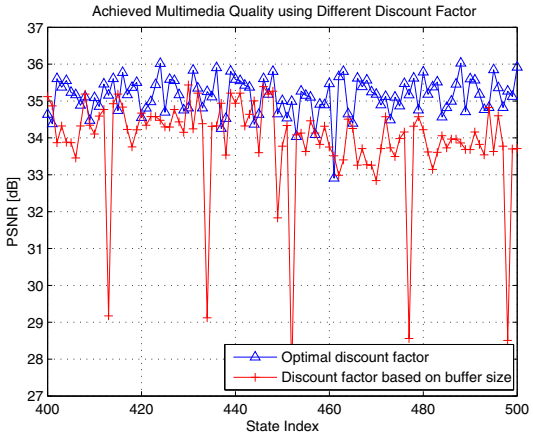

Fig. 10. Illustrative example of impact of discount factor on heterogeneous network conditions.

5. CONCLUSIONS

In this paper, we propose a scheduling strategy for video streaming over P2P networks formed by heterogeneous and self-interested peers. We propose a buffer structure which divides the buffer into a finite number of GOSs. This buffer design enables the proposed scheduling algorithm to successfully deliver video segments within the playback deadline. Moreover, this buffer structure allows us to model the resource reciprocation among peers in each GOS as a reciprocation game, and thus, peers can make foresighted decisions based on MDP, which lead them to maximize their long-term video qualities. Simulation results show that the proposed foresighted scheduling algorithm results in improved video qualities (i.e., achieves a higher PSNR) compared to other existing scheduling strategies such as TFT strategy. Moreover, the proposed scheduling algorithm enables the peers to flexibly operate in heterogeneous and dynamic P2P networks by computing the optimal value of the discount factor.

6. REFERENCES

[1] J. Liu, S. G. Rao, B. Li, and H. Zhang, "Opportunities and challenges of peer-to-peer internet video broadcast," *Proceedings of the IEEE, Special Issue on Recent Advances in Distributed Multimedia Communications*, 2007.

[2] S. Androutsellis-Theotokis and D. Spinellis, "A survey of peer-to-peer content distribution technologies," *ACM Comp. Surveys*, vol. 36, no. 4, pp. 335–371, Dec. 2004.

[3] "Gnutella." [Online]. Available: http://www.gnutella.com

[4] "KaZaA." [Online]. Available: http://www.kazaa.com

[5] "Napster." [Online]. Available: http://www.napster.com

[6] B. Cohen, "Incentives build robustnessin BitTorrent," in *Proc. P2P Economics Workshop*, Berkerly, CA, 2003.

[7] A. Legout, N. Liogkas, E. Kohler, and L. Zhang, "Clustering and sharing incentives in BitTorrent systems," *SIGMETRICS Perform. Eval. Rev.*, vol. 35, no. 1, pp. 301–312, 2007.

[8] X. Zhang, J. Liu, B. Li, and T. S. P. Yum, "CoolStreaming/DONet: A data-driven overlay network for efficient live media streaming," in *Proc. IEEE Conf. of Comput. and Commun. Soc. (INFOCOM '05)*, vol. 3, Mar. 2005, pp. 2102–2111.

[9] Z. Xiang, Q. Zhang, W. Zhu, Z. Zhang, and Y.-Q. Zhang, "Peer-to-peer based multimedia distribution service," *IEEE Trans. Multimedia*, vol. 6, no. 2, pp. 343–355, Apr. 2004.

[10] V. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy, and A. E. Mohr, "Chainsaw: Eliminating trees from overlay multicast," in *Proc. 4th Int. Workshop on Peer-to-Peer Systems (IPTPS)*, Feb. 2005.

[11] H. Park and M. van der Schaar, "A framework for foresighted resource reciprocation in P2P networks," *IEEE Trans. Multimedia*, vol. 11, no. 1, pp. 101–116, Jan. 2009.

[12] M. I. A. Vlavianos and M. Faloutsos, "Bitos: Enhancing bittorrent for supporting streaming applications," in *Global Internet Workshop in conjunction with IEEE INFOCOM 2006*, Apr 2006.

[13] M. van der Schaar and P. A. Chou, Eds., *Multimedia over IP and Wireless Networks*. Academic Press, 2007.

[14] D. P. Bertsekas, *Dynamic Programming and Stochastic Control*. Academic Press, 1976.

[15] M. L. Littman, T. L. Dean, and L. P. Kaelbling, "On the complexity of solving Markov decision problems," in *Proc. 11th Conf. on Uncertainty in Artificial Intelligence*, May 1995.

[16] H. Park and M. van der Schaar, "Bargaining strategies for networked multimedia resource management," *IEEE Trans. Signal Process.*, vol. 55, no. 7, pp. 3496–3511, Jul. 2007.

[17] Y. Andreopoulos, A. Munteanu, J. Barbarien, M. van der Schaar, J. Cornelis, and P. Schelkens, "In-band motion compensated temporal filtering," *Signal Processing: Image Communication (special issue on "Subband/Wavelet Interframe Video Coding")*, vol. 19, no. 7, pp. 653–673, Aug. 2004.