An Improved Foresighted Resource Reciprocation Strategy for Multimedia Streaming Applications

Ester Gutiérrez*, Hyunggon Park† and Pascal Frossard‡

*Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona, Universitat Politècnica de Catalunya, Barcelona, Spain
†Multimedia Communications and Networking Laboratory, Ewha Womans University, Seoul, Korea
‡Signal Processing Laboratory (LTS4), Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland
* estergut@gmail.com, † hyunggon.park@ewha.ac.kr, ‡ pascal.frossard@epfl.ch

***Abstract*—In this paper, we present a solution to efficient multimedia streaming applications over P2P networks based on the foresighted resource reciprocation strategy. We study several priority functions that can explicitly consider the timing constraints and the importance of each data segment in terms of multimedia quality, and successfully incorporate them into the foresighted resource reciprocation strategy. This enables peers to enhance their multimedia streaming capability. The simulation results confirm that the proposed approach outperforms existing algorithms such as tit-for-tat in BitTorrent and BiToS solutions.**

***Index Terms*—multimedia streaming, peer-to-peer networks, foresighted resource reciprocation strategy, priority functions.**

I. Introduction

In recent years, multimedia streaming services are growing rapidly and becoming pervasive applications over the Internet. Traditional client-server approaches allocate a dedicated stream from the server for each client request. However, this renders these approaches expensive and does not scale well to a large number of users. Therefore, peer-to-peer (P2P) based approaches have been proposed to overcome these limitations. In P2P networks, there is no dedicated infrastructure. Rather, the peers in the networks share their resources (e.g., content, bandwidth, etc.) by receiving them from other peers or contributing them to the other peers. One of the fundamental advantages of using P2P networks for multimedia streaming applications is to leverage peer upload capacities to minimize the bandwidth costs on dedicated streaming servers. Thus, P2P networks provide superior reliability and scalability compared to classic approaches based on the client-server model. However, existing solutions for content distribution or content sharing cannot efficiently support the quality of service (QoS) in P2P multimedia streaming applications.

The most popular P2P protocol that is currently deployed in file sharing is BitTorrent [1], [2]. However, the focus of this protocol is on efficient content distribution over P2P networks, without considering the timing constraints. Hence, this protocol can only provide a limited performance for multimedia streaming applications. A slightly modified BitTorrent protocol is designed for multimedia streaming, which is referred to as BiToS [3]. This protocol incorporates the packet (or data segment) selection process into the original BitTorrent protocol, thereby improving the multimedia streaming performance. There are also several approaches such as [4], [5] for efficient multimedia streaming in P2P networks. In order for efficient support of multimedia streaming, they deploy different peer or piece selection algorithms. For example, in [4], supplying peers having the highest bandwidth are selected and time allocation slots and data segments are selected depending on the number of potential suppliers. In [5], however, data segments in the sliding window are randomly selected.

While these solutions lead to an improved support for multimedia streaming, the resource reciprocation strategy does not consider the interactions of self-interested and heterogeneous peers. Moreover, the peers' resource reciprocations are determined such that they maximize their immediate utilities, without taking into account the impact of them on their future utilities. To overcome these limitations, a foresighted resource reciprocation strategy is proposed in [6], where peers with the foresighted resource reciprocation strategy can decide their resource reciprocations, such that they can maximize their long-term utilities, i.e., the immediate utility as well as the future utilities. This approach thus can lead to a better efficiency for resource reciprocation in P2P networks. However, this solution does not explicitly consider the time constrains that are critical for efficiently supporting streaming applications.

In this paper, we build on the foresighted resource reciprocation strategy by explicitly considering the timing constraints for continuous display of the multimedia data and the importance of each multimedia data segment for the multimedia quality. In particular, we incorporate data priority functions into the reward function in order to adapt to the specific characteristics of media streaming applications. As a result, the peers exchange video packets with a strategy that ensures that the most important packets have a higher probability to reach the decoder on time for proper decoding. Our simulation

This work was supported in part by the Swiss National Science Foundation grant 200021-118230, in part by the Ewha Womans University Research Grant of 2010 (Grant No. 2010-0238-1-1), and in part by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2010-0009717). This work was performed while E. Gutiérrez and H. Park were with EPFL.

MMSP'10, October 4-6, 2010, Saint-Malo, France.
978-1-4244-8112-5/10/$26.00 ©2010 IEEE

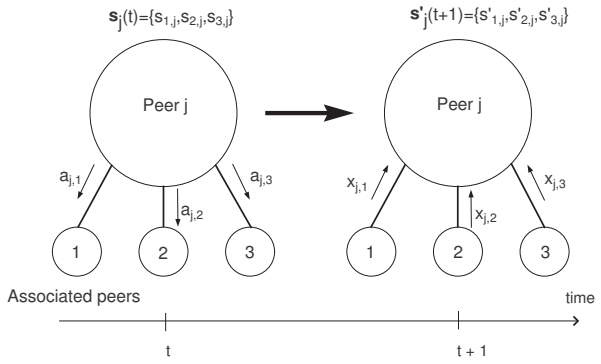

Fig. 1. An illustration of resource reciprocation of peer j associated with three peers at time t and $t+1$.

results show that the proposed algorithm improves the packet loss rates, utility value, and the video quality in media streaming scenarios, where it outperforms the BitTorrent and BiToS solutions.

This paper is organized as follows. In Section II, we briefly overview the foresighted resource reciprocation strategy and discuss its limitations for media streaming applications. In Section III, we propose a solution that can improve the foresighted resource reciprocation strategy, thereby efficiently supporting the media streaming over P2P networks. In Section IV, we describe how the implementation of the proposed solution is actually developed. Then, we can confirm that the proposed solution provides an improved performance compared to several existing approaches in Section V. Finally, the conclusions are drawn in Section VI.

II. Preliminaries

A. Overview of Foresighted Resource Reciprocation

In this section, we briefly overview the foresighted resource reciprocation [6], which was originally proposed to replace the tit-for-tat and optimistic unchoke mechanisms in BitTorrent systems [1]. In [6], the resource reciprocation among peers that are interested in each other's content is modeled as a stochastic game, and the foresighted resource reciprocation strategy can be found in the Markov Decision Process (MDP) [7] framework.

Specifically, the MDP for peer j consists of a set of states S_j, a set of actions A_j, reward function $R(\mathbf{s}_j, \mathbf{a}_j)$ for $\mathbf{s}_j \in S_j$ and $\mathbf{a}_j \in A_j$, and the resulting state transition probability P_j. The state transition probability determines a probability of mapping each state into the next state given an action, i.e.,

$$P_j : S_j \times A_j \times S_j \rightarrow [0, 1]. \tag{1}$$

An example of the resource reciprocation game is shown in Fig. 1. This example shows how the state of peer j evolves from $\mathbf{s}_j$ to $\mathbf{s}_j^{'}$ depending on action $\mathbf{a}_j$ at time t, and the reactions of its associated peers $\mathbf{x}_j$ at time $t+1$. In this section, we briefly review the key ideas how to formulate the resource reciprocate games in the MDP framework. More details can be found in [6].

1) State Space: A state space of peer j represents a set of resources received from the associated peers in the peer set at time t. In this paper, we consider that each state $\mathbf{s}$ can have two values 0 and 1, i.e.,

$$s_{i,j} = \begin{cases} 1, & \text{if } r_{i,j} > 0 \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

where $r_{i,j}$ represents peer j resources received from peer i. Then, we can denote the state space of peer j in its peer set as

$$S_j = \{(s_{1,j}(t), \ldots, s_{N,j}(t)) | s_{i,j} \in \{0, 1\}\}.$$

2) Action space: An action space is defined as a set of actions that peer j can take in its peer set. We consider that an action of peer j to peer i can be either 0 or 1, which represents whether peer j chokes or unchokes peer i, respectively. If peer j unchokes peer i, this means that peer j shares its contents with peer i. Otherwise, peer j chokes i by sharing no content with peer i. A set of actions that peer j can take to its associated peers can thus be expressed as

$$A_j = \{(a_{1,j}, \ldots, a_{N,j}) | a_{i,j} \in \{0, 1\}\}.$$

3) Reward function:

Reward $R_j(\mathbf{s}_j(t))$ for a peer j in state $\mathbf{s}_j(t) \in S_j$ represents received resources from its state $\mathbf{s}_j(t)$, expressed as

$$R_j(t)(\mathbf{s}_j(t)) = \sum_{i=1}^{N} r_{i,j}(s_{i,j}) \tag{3}$$

where $r_{i,j}(s_{i,j})$ denotes the received resources from $s_{i,j}$.

4) Reciprocation Policy π_j^* : A reciprocation policy provides optimal actions $\mathbf{a}_j(t) \in A_j$ from states $\mathbf{s}_j(t) \in S_j$, i.e.,

$$\pi_j^*(\mathbf{s}_j) = \mathbf{a}_j.$$

This policy can be obtained from a solution to the MDP and peer j can make foresighted decisions from all of its states. The foresighted actions enable peer j to achieve a maximum cumulative discounted rewards (i.e., the immediate expected reward and the discounted future rewards) [6]. The cumulative discounted expected reward at time t can be expressed as

$$R_j^{fore}(t)(\mathbf{s}_j(t)) \triangleq \sum_{t=t'+1}^{\infty} \gamma_j^{(t-(t'+1))} E[R_j(t)(\mathbf{s}_j(t))] \tag{4}$$

where γ denotes a discount factor. The resource reciprocation policy can be obtained based on well-known methods such as value iteration and policy iteration. Note that solving MDP may require high computational complexity, which exponentially increases as more peers are considered in a peer set.

Thus, it is important to only consider appropriate peers in each peer set. An illustrative implementation is discussed in [8].

B. Limitations in Real-time Streaming Applications

The foresighted resource reciprocation strategies in [6] enable peers to maximize their cumulative discounted expected rewards (e.g, total received resources). Since the focus is only on maximizing the received resources (i.e, download rates), it only provides a limited performance for multimedia streaming applications, as it does not consider multimedia characteristics especially timing constraints of multimedia data. This may result in undesirable interruptions of continuous playback of multimedia streams. In order to overcome this limitation, each peer also needs to explicitly consider the *orderings* of the data segments, while downloading them as fast as possible. Specifically, each peer needs to receive the data before their decoding deadlines. Moreover, each data packet may have different quality impact depending on the encoding structure and the type of picture in multimedia streaming applications (e.g., video streaming). Hence, the original resource reciprocation strategy in [8] needs to be modified, such that it can efficiently consider these additional constraints.

In the next section, we introduce a priority function, which will be incorporated into the original resource reciprocation strategy, while explicitly considering the decoding deadlines (e.g., positions of data segments) and the quality impact.

III. Foresighted Streaming Resource Reciprocation Strategy

The newly introduced priority function ρ is incorporated into the reward function in (3), which is described as

$$R_j(\mathbf{s}_j(t)) = \sum_{i=1}^{N} \rho_{i,j}(x) r_{i,j}(s_{i,j}) \tag{5}$$

where $\rho_{i,j}(x)$ represents the preference of peer j for segments that peer i has at time t. $\rho_{i,j}(x)$ can be further specified by considering the delay deadline and the quality impact of each segment, which can be expressed as

$$\rho_{i,j}(x) = \rho^T_{i,j}(x)\rho^D_{i,j}(x) \tag{6}$$

where $\rho^T_{i,j}(x)$ and $\rho^D_{i,j}(x)$ are the functions that depends on the timing constraints (i.e., relative position in the ideal decoding buffer) and the quality impact of segment x, respectively.

A. Position Depending Priority Functions

In order to successfully support real-time multimedia streaming applications, timing constraints from each data packet should be taken into account. Thus, the data segments can be prioritized based on their display time and peers can download more important data segments (i.e., the segments having higher priority) earlier.

An illustrative example of data segments (or chunks) in a buffer at time t are shown in Fig. 2. In this example, shadowed packets represent already downloaded segments and the rest of the packets are scheduled for downloading. The dotted square box represents a sliding window, which represents the packets that are scheduled for decoding.

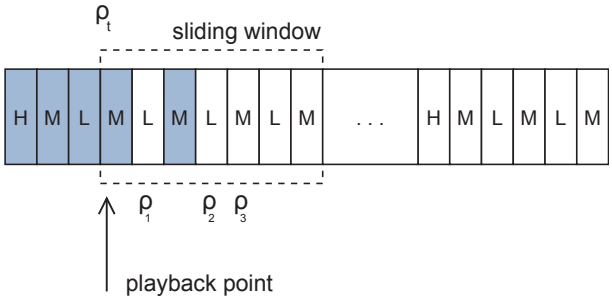

Fig. 2. An illustrative example of time-ordered data segments in an ideal decoding buffer.

In general, data segments that are not downloaded yet and are closer to the playback point can have higher priorities. Hence, we can define $\rho^T_{i,j}(x)$ as a monotonically non-increasing function depending on segments' positions in the buffer. This enables each peer to download more important segments earlier. In this section, we consider two illustrative priority function shapes that depend on segments' positions.

1) Square Shape Priority Function: The motivation of square shape priority function is to divide the segments into two groups - groups with high priority and with low priority. This can be easily extended to several levels of priority by defining stair shape priority functions.

The corresponding priority function $\rho^T_{i,j}(x)$ can be defined as

$$\rho^T_{i,j}(x) = \begin{cases} \alpha_H & \text{if} \quad x < d \\ \alpha_L & \text{if} \quad x > d \end{cases} \tag{7}$$

where $\alpha_H > \alpha_L$ and d is a threshold dividing the two groups. The data segments in the group of high priority are closer to the playback point than those in the group of low priority. The value of d can be determined as a percentage for the part of the total number of data segments in a file.

2) Exponential Shape Priority Function: Unlike the square shape priority function, which imposes the same priority on the data segments in a group, exponential shape priority function can impose different priorities to data segments. This type of function also enables the peers to easily control the variation of priorities among the segments, leading to maximum performance.

The exponential shape priority function can be expressed as

$$\rho^T_{i,j}(x) = e^{-\beta \cdot x} \tag{8}$$

where β determines how fast/slow the priorities are decaying.

B. Segment-Type Depending Priority Functions

A sequence of compressed video frames may have different types, such as I-frame, P-frame, and B-frame in MPEG or H.264 standards [9], [10]. Thus, the corresponding data segments can also be characterized by the type of the frame they represent. For simplicity, we consider that the priority function based on the segment types is determined by the quality impact of the types.

An example is depicted in Fig. 2, where data segments are marked with H, M, and L, denoting High, Medium and

Low quality impacts, respectively. The corresponding priority function $\rho^D_{i,j}(x)$ is expressed as

$$\rho^D_{i,j}(x) = \begin{cases} \gamma_H, & \text{if} \quad x \in \mathcal{H} \\ \gamma_M, & \text{if} \quad x \in \mathcal{M} \\ \gamma_L, & \text{if} \quad x \in \mathcal{L} \end{cases} \tag{9}$$

where $\mathcal{H}$, $\mathcal{M}$, and $\mathcal{L}$ denote the set of data segments marked with H, M, and L, respectively. The priority function $\rho^D_{i,j}(x)$ can provide different levels of priorities on each data segments based on its quality impact.

Finally, the streaming reward function will we calculated as

$$R_j^{fore}(t)(\mathbf{s}_j(t)) \triangleq \sum_{t=t'+1}^{\infty} \gamma_j^{(t-(t'+1))} \sum_{i=1}^{N} \rho_{i,j}(x) r_{i,j}(s_{i,j}) \tag{10}$$

C. Utility

In order to quantify the impact of the priority function on the average performance of the proposed algorithm, we define a utility measure $U_j(t)$ for peer j, as

$$U_j(t) = \sum_{x \in N_p} \rho^T_{i,j}(x) \cdot (1 - P_L(x)) \tag{11}$$

where N_p represents the total number of data segments and $P_L(x)$ represents the packet loss probability or equivalently the probability for a data segment located at position x not to be received by the playback deadline. The utility can be a measure for the system's efficiency, as higher utilities imply that more urgent packets are successfully received with higher probability.

IV. Implementation

Our implementation of the proposed algorithm is based on [8], where the tit-for-tat and the optimistic unchoke mechanisms in BitTorrent systems are replaced with the foresighted reciprocation mechanisms. The implementation of the protocol consists of three main processes running in parallel, which are Learning Process, Decision Process and Policy Finding Process [8]. We briefly review these processes next. A new piece selection mechanism that can replace the rarest first protocol in BitTorrent is also implemented in our design.

A. Learning Process

The goal of the *Learning Process* is to provide the system with the information of peer set behaviors in order to calculate the optimal resource reciprocation policy. To successfully calculate the policy, the peer j needs to update the information of the peers in its peer set, such as the probability of state transition and their current upload rates to peer j. Thus, the implementation of the Learning Process consists of two main methods: the state transition probability calculation and the upload rate estimation.

B. Policy Finding Process

The *Policy Finding Process* plays a role of obtaining the optimal resource reciprocation policy by solving MDP. Based on the information updated by the learning process, it is able to compute the future rewards in (10) and calculate the optimal actions for each state. However, the calculation of the optimal policy may require considerable computational time and complexity, depending on the number of peers that are associated with peer j. Hence, this process requires to consider a limited number of associated peers, which can be found based on the peer-set reduction algorithm [8]. Once the reduced peer set is found, the policy finding process computes the optimal policy and provides the policy to the decision process. In our implementation, the policy holds up for 5 rechoke periods, and the peer set reduction is calculated every 10 rechoke periods, or whenever a peer leaves the system.

C. Decision Process

The purpose of the *Decision Process* is to determine the optimal actions identified by the optimal policy π^*_j. Therefore, it can generate the actions that peer j will take in its current state, i.e.,

$$\pi^*_j(\mathbf{s}_j(t)) = \mathbf{a}_j(t)^*.$$

However, in the initial phase where the peer j first joins the system, it does not have enough information about its associated peers to calculate the policy. For this reason, in our implementation, the decision process begins with applying the tit-for-tat strategy until the number of reciprocations is sufficient for capturing the behaviors of other peers.

D. Piece Selection Algorithm

The objective of the *Piece Selection Algorithms* is to receive the most important data segments (i.e., urgent pieces) on time and improve the diversity of the pieces among the peers. The piece selection algorithm in our system includes two approaches. One approach enables the peers to download pieces with higher priorities first, leading to maximum system utility. The other approach adopts the rarest first search algorithm which is used in BitTorrent systems. This can maximize the entropy of the pieces. The replication of the rarest pieces can reduce the risk of losing them in the case where current peers stop uploading the pieces.

In our implemented system, we adopt the former algorithm with probability p and the latter algorithm, i.e., rarest first search algorithm, with probability $(1 - p)$. The probability p can be determined by considering the tradeoff between receiving the most important pieces on time and the fast replication of the rarest pieces. As an illustration, we use a value of $p = 0.8$.

V. Simulation Results

A. Impact of Position Depending Priority Functions on Utility

We compare the utility achieved for the different position-dependent priority functions defined in Section III. Simulation results in Fig. 3 and Fig. 4 show the utilities achieved for

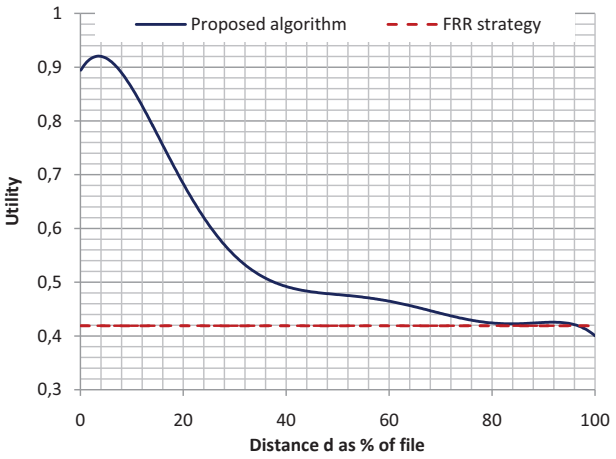

Fig. 3. Utility achieved for different values of d.

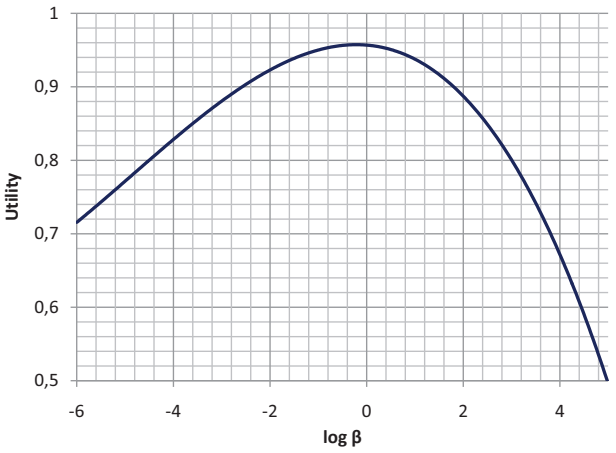

Fig. 4. Achieved utility for different values of β.

different values of d and β with the priority functions in (7) and (8), respectively. In the simulations, we consider a small set of peers, where the number of peers is 10 and each peer has a total upload rate of 128Kbps. The downloading file size is 17MB. The achieved utility is calculated after 50 rechoke periods. Simulations are performed based on the assumption of static network condition, i.e., a given network topology with a fixed number of peers.

In Fig. 3, we can observe that higher utilities can be achieved for lower values of d, while the achieved utilities decrease as the values of d increases. Finally, the achieved utilities becomes similar for a large values of d. If the values of d for each peer become high (e.g., $d > 80\%$), the role of the priority functions is minimized, leading to the similar effect that there is no priority function. However, if lower values of d (e.g., $d < 10\%$) are used for each peer, then all the peers focus on downloading only a small number of data segments with high priority, leading to a small availability of data segment replicas.

Simulation results for the exponential shape function with different values of β are shown in Fig. 4. The behavior of the resulting utilities are similar to the case of the square shape function. For large values of β, the priority function becomes similar to a narrow square function, which results in the similar effect of the case where small values of d are used in (7). On the other hand, for small values of β, which is the similar to the case where d becomes large, the achieved utilities decrease because higher priorities are imposed to more data segments. In this example, $\beta = 1$ leads to the highest utility.

Since more granular priorities than the square shape priority function can be imposed in the exponential shape priority function, better performance (i.e., higher utility) can be achieved, as shown in Fig. 3 and Fig. 4.

B. Comparison of Scheduling Algorithms

We compare here the performance of the foresighted streaming resource reciprocation algorithm with existing approaches such as BitTorrent [1], [2], BiToS [3] and Foresighted Resource Reciprocation [6] .

Recall that the tit-for-tat strategy in BitTorrent is implemented to perform the choking decisions and the rarest first search algorithm is implemented for selecting the data segments to download first. Since the tit-for-tat strategy does not consider the timing constraints in transmission of data segments, this may not be appropriate for multimedia data transmission. In order to overcome this, an improved BitTorrent protocol, BiToS has been in [3] designed.

This protocol uses the tit-for-tat strategy but improves the streaming capability by adding a *selection process*. In this selection process, data segments (or pieces) are divided in two groups, which are the *high priority set* and the *remaining pieces set*. Peers choose to download pieces contained in the high priority set with probability p and download the other pieces with probability $(1 - p)$. This enables the BiTos algorithm to outperform the BitTorrent protocol in terms of media streaming performance.

In our simulation, we consider a small set of 15 peers as an illustration, where each peer has a total upload rate of 128Kbps. Each peer has six associated peers, which are selected based on the *peer set reduction* algorithm. We assume that the number of simultaneous downloading slots is four, which leads to four maximum associated peers unchoked simultaneously. The discount factor is $\gamma = 0.8$ as in [6]. For BiToS, we set $p = 0.8$, and for the tit-for-tat strategy, the number of slots for downloading is 4, which is a default setting of BitTorrent [1], [2]. In our algorithm, we use the position depending priority function defined in (8) with $\beta = 1$.

Packet loss rate and utilities achieved based on these four algorithms after 50 rechoke periods are shown in Fig. 5. The results clearly shows that our approach provides a better performance in terms of packet loss rates. We also note that our approach not only reduces the packet loss rates of urgent data segments, but also improves the long-term performance. This is because of the foresighted resource reciprocation strategy, which focuses on maximizing the long-term rewards. The utilities achieved for a single peer based on these algorithms

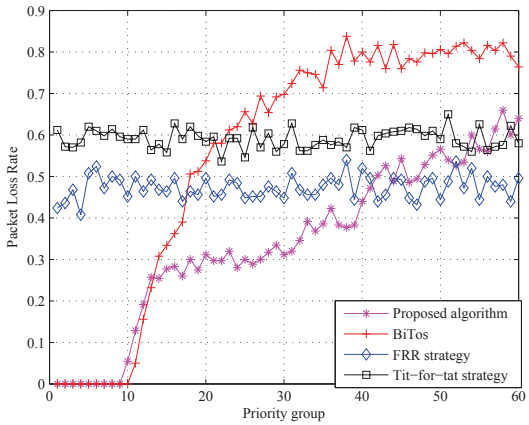

Fig. 5. Packet loss rates with different priorities. Smaller numbers in x-axis (priority group) represent higher priorities than those of larger numbers.

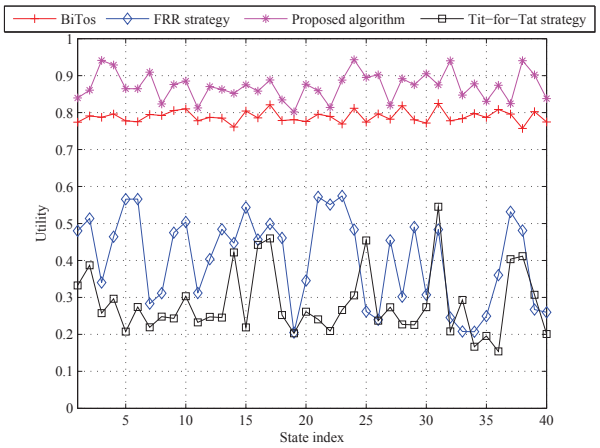

Fig. 6. A peer's utilities achieved based on several algorithms. The state index denotes the initial state of the peer.

are shown in Fig. 6. The state index represents the different initial states of the peers. The results show that the proposed algorithm always outperforms the other algorithms regardless of a peer's initial states.

We also quantitatively compare the proposed algorithm with the other existing algorithms in multimedia streaming applications. In these simulations, the peers download a video sequence, which is the *Salesman* sequence with QCIF resolution at the frame rate of 30 frames/second. The video file has a length of 120 seconds, which is generated by concatenating the original video sequence 8 times. Fig. 7 shows the PSNRs achieved for different numbers of different values of the playback delays, represented by the number of rechoke periods. The results also show that the proposed algorithm outperforms the other algorithms in terms of video quality. Alternatively, we can observe that the proposed approach requires less number of rechoke periods to achieve a certain level of PSNR, which implies that the proposed algorithm

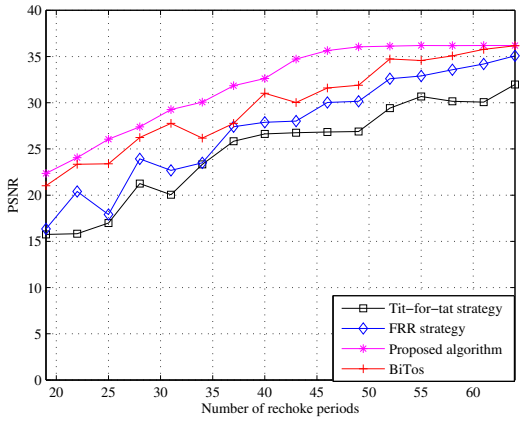

Fig. 7. Achieved PSNRs for different rechoke periods.

permits to decrease the playback delays.

VI. Conclusions

We propose a foresighted resource reciprocation algorithm, that can efficiently support the multimedia streaming applications. In order to explicitly consider the timing constraints and different impacts of each data segment on the quality, we propose and study several priority functions, and we successfully incorporate them into the foresighted resource reciprocation strategy. Simulation results confirm that our approach outperforms several existing algorithms such as tit-for-tat and BiToS in P2P networks, in terms of packet loss rates, utility, and the video quality for given playback delays. Interesting future research topics may include the study of how to decide the system variables, β and p, in the Piece Selection Algorithm in dynamic networks.

References

[1] B. Cohen, "Incentives build robustness in BitTorrent," in *Proc. P2P Economics Workshop*, Berkeley, CA, 2003.

[2] A. Legout, G. Urvoy-Kellerand, and P. Michiardi, "Understanding bittorrent: An experimental perspective," INRIA, Tech. Rep. INRIA-00000156, Nov. 2005.

[3] A. Vlavianos, M. Iliofotou, and M. Faloutsos, "BiToS: Enchancing bittorrent for supporting streaming applications," *IEEE INFOCOM 2006*, Apr. 2006.

[4] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum, "Coolstreaming/donet: A data-driven overlay network for efficient live media streaming," *Proc. IEEE INFOCOM 2005*, pp. 2102–2111, Mar 2005.

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

[6] H. Park and M. van der Schaar, "A framework for foresighted resource reciprocation in P2P networks," *IEEE Trans. Multimedia*, vol. 11, pp. 101–116, Jan. 2009.

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

[8] R. Izhak-Ratzin, H. Park, and M. van der Schaar, "Foresighted resource reciprocation strategy in BitTorrent systems," UCLA Computer Science Department, Tech. Rep. 10-0014.

[9] A. Luthra, G. J. Sullivan, and T. Wiegand, "Overview of the h.264/avc video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, Jul. 2003.

[10] P. A. Chou and Z. Miao, "Rate-distortion optimized streaming of packetized media," *IEEE Trans. Multimedia*, vol. 8, no. 2, Apr. 2006.