VIDEO TRANSMISSION SCHEDULING FOR PEER-TO-PEER LIVE STREAMING SYSTEMS

Ying Li[1] *Zhu Li*[2] *Mung Chiang*[1] *A. Robert Calderbank*[1]

[1] Department of Electrical Engineering, Princeton University, NJ, USA.
[2] Department of Computing, Hong Kong Polytechnic University, Hong Kong.

ABSTRACT

For Internet based video broadcasting applications such as IPTV, the Peer-to-Peer (P2P) streaming scheme has been found to be an effective solution. An important issue in live broadcasting is to avoid playback buffer underflow. How to utilize the playback buffer and upload bandwidth of peers to minimize the freeze-ups in playback, is the problem we try to solve. In this work, we propose a successive water-filling (SWaF) algorithm for the video transmission scheduling in P2P live streaming system, to minimize the playback freeze-ups among peers. SWaF algorithm only needs each peer to optimally transmit (within its uploading bandwidth) part of its available video segments in the buffer to other peers requiring the content and pass small amount message to some other peers. Moreover, SWaF has low complexity and provable optimality. Numerical results demonstrated the effectiveness of the proposed algorithm.

Index Terms— Peer-to-peer, scheduling, water-filling.

1. INTRODUCTION

Peer-to-Peer (P2P) live streaming has become a viable solution for IPTV [4] services with medium quality video for a large number of concurrent users [3]. With the popularity of video on demand applications over Internet, the traditional client-server and content server at edge solutions are not adequate in handling dynamic viewer behaviors and do not scale well with a large audience. On the other hand, the P2P based solutions utilizing application layer overlay are becoming popular, because it is easy to implement and cheaper than duplicating content servers at edges. The core benefit of P2P based solution is that it utilizes the buffering and uploading capacities of the participating peers, and provides a more scalable and robust content delivery solution.

With the great success of many P2P streaming systems, e.g., PPLive [10], PPStream [11], CoolStreaming [14], there comes some work analyzing the operation and tradeoff of such systems (see [13] [6] [9] [2] and references therein). In [13] the authors investigate the scaling law by quantitatively studying the asymptotic effects and tradeoffs. Recently in [6] the authors analyze the pull-based P2P streaming systems based on fluid model, and construct a simple 2-hop scheduling algorithm for a buffer-less system. Systems with buffer are also studied using extensive simulations. In [9] the authors consider the resilience utilizing path diversity for multicast tree based P2P streaming systems. In [2] the authors present a survey of the media distribution methods, overlay structures and error control solutions proposed for P2P live streaming. However, none of these works gives an analytical model of the P2P streaming system under heterogeneous buffer occupancies.

P2P downloading system, like Bit Torrent [12] has been extensively studied. However, P2P downloading system does not consider real time playback constraint and thus does not have the playback buffer underflow issues as is addressed in this paper.

In a P2P streaming system, it is desirable to minimize the probability of "freeze-ups" during the live video playback. In other words, the extra buffered content (i.e., content reserve level) of each peer should be kept positive. Paper [7] uses the well-known α-fair utility function to model users' satisfactory level of their current content reserve levels, and allocates network resources (i.e., transmission rates) to all users to maximize the total utility. It shows that the optimal solution can balance the buffer occupancy level among users and it proposes a greedy heuristic solution that achieves the desirable result.

The heuristic in [7] is a centralized algorithm, which needs a coordinator to do all the computing for scheduling. In a P2P network, a distributed algorithm is favorable, where the computing load is distributed over peers. In this work, we propose successive water-filling (SWaF) scheduling algorithm. In general, SWaF can be implemented in a centralized way, moreover, for the case where the propagation delay of the message passing is small compared with the time interval for uploading capacity to vary, SWaF can be implemented in a distributed way, where every peer can solve its local problem, with some message passing to other peers. Theoretically we prove the optimality of proposed sWaF algorithm, and we show that our algorithm is even simpler than the heuristic in [7].

The rest of this paper is organized as follows. In Section II, we provide the system model and problem formulation. In Section III, we propose the successive water-filling algorithm. We present numerical examples in Section IV and in Section V we summarize our results and propose future work.

2. SYSTEM MODEL

As in [7], we consider a P2P system with a video source and N peers, with uploading bandwidth $C_0, C_1, \cdots, C_N$ respectively. Let the content be streamed over all peers with a constant bit rate (CBR) R_0. The playback buffer of each peer k is characterized by the buffer state tuple, $\{x_k, t_k, y_k\}$, where x_k and y_k are the buffer content starting point and ending point, respectively, and t_k is the current playback time. The content reserve level is $y_k - t_k$, which is the video segment remained in the buffer to be played.

For a scheduling interval T seconds, within its uploading capacity C_j, each peer j provides a content of playback time $z_{j,k}$ to peer k if the content is available ($0 \le z_{j,k} \le y_j - y_k$), and within its upload capacity C_0 the source provides a content of playback time $z_{0,k}$ to peer k. Then peer j transmits in a total bit rate of $\sum_k z_{j,k} R_0/T$ which should be no greater than C_j. The content reserve level of each peer k becomes $\sum_{j=0}^{N} z_{j,k} + (y_k - t_k)$.

An important consideration for P2P live video streaming system is to prevent peer buffers from underflow which may cause unpleasant "freezes" in the playback. Since each peer gets the content via the uploading link of other peers, and each peer wants high content reserve level, the peers are competing for the uploading bandwidth.

This work was partially supported by United States NSF CCF-0448012 and DARPA grant HR0011-06-1-0008.

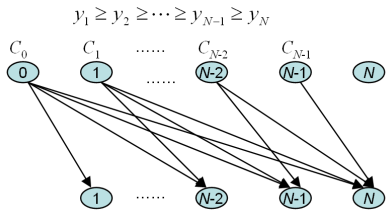

Fig. 1. Content provision pattern based on availability.

The limited uploading bandwidth needs to be allocated reasonably to make each peer have enough content reserve level. To characterize the system requirement, as in [7], we adopt a model similar to the network utility maximization framework in [5], to maximize the total happiness about playback smoothness of all the peers, where the happiness of peer k is measured by a utility function, $U_k(l_k)$, for content reserve level l_k.

In general, $U_k(l_k)$ is a concave function in l_k, which means the increase of the happiness slows down as the reserve level l_k gets larger. An example of such concave function is, as adopted in [7], the well-known α-fair utility function [8], $U_k(l_k) = w_k \frac{l_k^{1-\alpha}}{1-\alpha}$, where w_k (a positive number) is the weight indicating the relative importance of peer k in the system and $\alpha \in (0,1)$ is a parameter.

Hence, the problem of uploading bandwidth allocation is formulated as follows,

$$\begin{array}{ll} \text{maximize} & \sum_k U_k(\sum_{j=0}^{N} z_{j,k} + (y_k - t_k)) \\ \text{subject to} & \sum_k z_{j,k} \leq \beta C_j, \\ & 0 \leq \sum_{j\in J} z_{j,k} \leq \max\{0, \max_{j\in J} y_j - y_k\}, \quad (1) \\ & j \in \{0,1,\cdots,N\}, k \in \{1,\cdots,N\} \\ \text{variables} & z_{j,k}, \end{array}$$

where $\beta = T/R_0$, J is any subset of $\{0,1,\cdots,N\} \setminus \{k\}$. In this problem, the goal is to maximize the total utility over the playback buffer content reserve level, the first constraint states that the serving rate of the source or the peer should be no grater than the uploading capacity, and the second constraint states the content availability and no transmission repetition of content.

In [7], a centralized solution based on a greedy heuristic is proposed for a discrete version of problem (1) where the unit for the playback buffer is Group of Pictures (GoP), instead of the continuous time as in problem (1). In the following, we propose a distributed **successive water-filling (SWaF)** algorithm, where every peer solves its local problem, with some message passing to other peers. Theoretically we show that SWaF yields an optimal solution of (1). Computationally, SWaF is simpler than the algorithm in [7]. Note that in general, SWaF can be implemented as a centralized solution if a peer with good computing capacity is selected as a coordinator.

3. SUCCESSIVE WATER-FILLING (SWAF) ALGORITHM

For a practical solution, we assume a small group consisting of a source and N peers from all the peers in the system forms a cooperative group. The SWaF algorithm is within each cooperative group.

Denote θ_k as the content reserve level. Initially (at the beginning of current scheduling interval), $\theta_k = y_k - t_k$. Suppose $y_1 \geq y_2 \geq \cdots \geq y_N$, which means each C_j ($j = 0,1,2,\cdots,N-1$) can serve peers $j+1, j+2, \cdots, N$. Note that $z_{j,k} = 0, j \geq k$ because peer j cannot serve peer $k \leq j$ due to the content availability. We assume $z_{j,k} \leq y_j - y_k$ for $j < k$, which means peer j has enough content to serve peer $k > j$. Figure 1 shows the content provision pattern.

We propose the following distributed SWaF algorithm.

3.1. Distributed SWaF Algorithm

1. Initialization:

Every peer k, $k = 1,2,\cdots,N$, reports its y_k to peer 0 (this can be any other peer as well), then peer 0 sorts all the y_k's and then reports every peer k its neighbors $k-1$ and $k+1$ according to the sort result $y_1 \geq y_2 \geq \cdots \geq y_N$.

Peer k asks peer $k+1$ for its initial θ_{k+1}.

Every peer has one bit Cur_flag with initial value zero. Cur_flag=1 means current peer is running water-filling algorithm.

Peer 0 asks peer $N-1$ to set its Cur_flag=1.

2. Each peer j ***(***$j = N-1, N-2, \cdots, 1, 0$***) does the following:***

IF peer j's Cur_flag=1

Peer j calculates $z_{j,k}$ for $k = j+1,\cdots,N$, using water-filling approach:

$$z_{j,k} = \max\{U_k'^{-1}(\lambda_j) - \theta_k, 0\}, \quad k = j+1,\cdots,N, \qquad (2)$$

where λ_j is a positive number which is chosen such that $\sum_{k=j+1}^{N} z_{j,k} = \beta C_j$. λ_j is found by bisectional search.

Peer j sets its Cur_flag=0.

Peer j ($j \neq 0$) updates $\theta_k = \theta_k + z_{j,k}$, for $k = j+1,\cdots,N$, passes them to peer $j-1$.

Peer j ($j \neq 0$) asks peer $j-1$ to set its Cur_flag=1; Peer j ($j = 0$) reports 'The End'.

END IF

Note that $z_{j,k}$, as calculated in (2), is a water-filling solution of the following optimization problem,

$$\begin{array}{ll} \text{maximize} & \sum_{k=j+1}^{N} U_k(z_{j,k} + \theta_k) \\ \text{subject to} & \sum_{k=j+1}^{N} z_{j,k} \leq \beta C_j \\ \text{variables} & z_{j,k}, \ k \in \{j+1,\cdots,N\}. \end{array} \qquad (3)$$

By Lagrange dual approach and KKT condition [1], it is readily verified that the optimal solution of the problem (3) is the water-filling solution as calculated in (2), where λ_j is the Lagrange multiplier associated with the constraint in (3).

3.2. Optimality

Proposition 1 *SWaF algorithm yields an optimal solution for problem (1) assuming the second constraint is satisfied.*

Proof:

First, we look at C_{N-1}. Since peer $N-1$ can only serve peer N, it is easy to check SWaF algorithm gives $z_{N-1,N} = \beta C_{N-1}$ which is the optimal solution.

Then, we look at C_{N-2}. Since peer $N-2$ can only serve peer $N-1$ and peer N, at the optimum we have $z_{N-2,N-1} + z_{N-2,N} = \beta C_{N-2}$. Note that in problem (1) the first inequality constraint becomes equality at the optimal. We have the following,

$$\begin{aligned} & \max_{\substack{\sum_{k=j+1}^{N} z_{j,k} = \beta C_j \\ j=0,1,\cdots,N-1}} \ \sum_{k=1}^{N} U_k(\sum_{j=0}^{k-1} z_{j,k} + \theta_k) \\ = & \max_{\substack{\sum_{k=j+1}^{N-2} z_{j,k} \leq \beta C_j \\ j=0,1,\cdots,N-3}} \ \sum_{k=1}^{N-2} U_k(\sum_{j=0}^{k-1} z_{j,k} + \theta_k) \\ & + \max_{z_{j,N-1}+z_{j,N} = \beta C_j - \sum_{k=j+1}^{N-2} z_{j,k}, j=0,1,\cdots,N-2} \\ & [U_{N-1}(\sum_{j=0}^{N-2} z_{j,N-1} + \theta_{N-1}) \\ & + U_N(\sum_{j=0}^{N-2} z_{j,N} + \beta C_{N-1} + \theta_N)] \\ = & \max_{\substack{\sum_{k=j+1}^{N-2} z_{j,k} \leq \beta C_j \\ j=0,1,\cdots,N-3}} \ \sum_{k=1}^{N-2} U_k(\sum_{j=0}^{k-1} z_{j,k} + \theta_k) \\ & + \max_{z_{j,N-1}+z_{j,N} = \beta C_j - \sum_{k=j+1}^{N-2} z_{j,k}, j=0,1,\cdots,N-3} \\ & [U_{N-1}(\sum_{j=0}^{N-2} z_{j,N-1} + \theta_{N-1}) \\ & + U_N(\sum_{j=0}^{N-2} z_{j,N} + \beta C_{N-1} + \theta_N)]|_{z_{N-2,N-1}, z_{N-2,N}:P} \\ = & \max_{\sum_{k=j+1}^{N} z_{j,k} = \beta C_j, j=0,1,\cdots,N-1} \\ & \sum_{k=1}^{N} U_k(\sum_{j=0}^{k-1} z_{j,k} + \theta_k)|_{z_{N-2,N-1}, z_{N-2,N}:P} \end{aligned} \qquad (4)$$

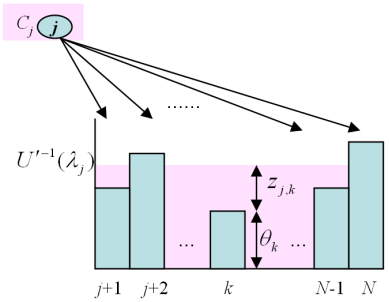

Fig. 2. Illustration of water-filling.

where P is the problem of max $[U_{N-1}(z_{N-2,N-1} + \theta_{N-1}) + U_N(z_{N-2,N} + \theta_N)]$ over variables $z_{N-2,N-1}, z_{N-2,N}$ with constraint $z_{N-2,N-1} + z_{N-2,N} = \beta C_{N-2}$. Equation (4) shows that an optimal $z_{N-2,N-1}$ and $z_{N-2,N}$ can be derived by solving P, where P can be solved by a water-filling algorithm.

Similarly, given the optimal $z_{N-1,N}$, $z_{N-2,N-1}$ and $z_{N-2,N}$ derived in previous steps, we further assign C_{N-3} by a corresponding water-filling algorithm getting an optimal $z_{N-3,k}$ for $k = N-2, N-1, N$ where $\sum_{k=N-2}^{N} z_{N-3,k} = \beta C_{N-3}$.

Successively, when SWaF algorithm finishes assigning C_0, we get an optimal solution. ■

3.3. Complexity

The complexity of SWaF algorithm is $O(N \log \Lambda)$, where N is the number of peers and Λ is the number of equal-partitions of the searching interval of Lagrangian multiplier of the water-filling algorithm where the partition is based on the accuracy tolerance.

The complexity of the algorithm in [7] is $O(NM)$ where M is the number of GoPs transmitted by one peer in a scheduling interval. As our numerical results show, SWaF is simpler than the algorithm in [7], and this advantage is more obvious when M is large. Note that if the accuracy tolerance for the scheduling by SWaF algorithm is about one GoP, SWaF has a complexity $O(N \log M)$. This explains that when M is large, the simplicity of SWaF is more obvious.

3.4. Remarks

We have the following comments on SWaF.

Remark 1. We assume the second constraint in problem (1) is satisfied above, which means the solution of z happens to satisfy the constraint. If we do not have such assumption, the water-filling in (2) becomes

$$z_{j,k} = \min\{y_j - (y_k + \theta_k), \max\{U_k'^{-1}(\lambda_j) - \theta_k, 0\}\}, \qquad (5)$$

and correspondingly, in the initialization of SWaF, peer j needs peer k, $k = j+1, \cdots, N$ to pass its y_k to peer j.

Remark 2. If all the peers use the same concave utility function, the shape of the utility function does not affect the algorithm. This is because, in such case, $U_k'^{-1}(\lambda_j)$ in the water-filling algorithm (2) will become a common $U'^{-1}(\lambda_j)$, which can be interpreted as 'water level' [1], and a direct bisectional search for the 'water level' can be done instead of the bisectional search for λ_j. The 'water level' is of our interest for the optima, not the dual variable λ_j. The illustration for this case is shown in Fig. 2. It is easy to see that the 'water level' is in the interval of $[\min \theta_k + \beta C_j/(N-j), \max \theta_k + \beta C_j/(N-j)]$ and the bisectional search is easy and fast.

Remark 3. SWaF only gives one optimal solution of problem (1). Note that the optimal solution of problem (1) may not be unique.

Remark 4. SWaF here is in a distributed fashion, but in general, it can be implemented centrally, if a peer with good computing capacity is selected as a coordinator and it does all the computing.

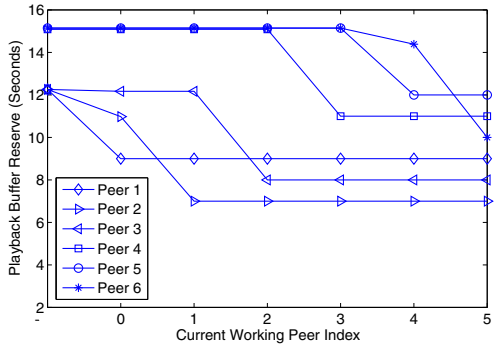

Fig. 3. Peer buffer content reserve states.

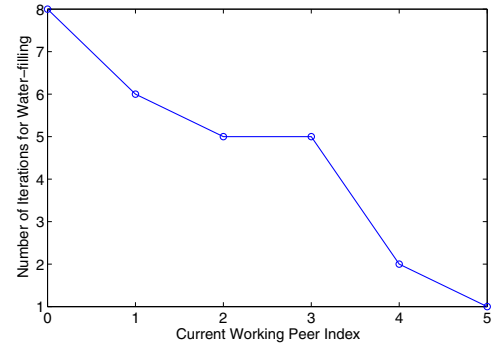

Fig. 4. Number of iterations for water-filling at each serving peer.

Remark 5. When SWaF is in a distributed fashion, the dynamics of bisectional search for λ is local, which means each peer k searches its own λ_k, and would not affect the upload-link dynamics of others.

Remark 6. Distributed SWaF requires the scheduling interval greater than the total computing time and message passing time. Note that since SWaF is a successive algorithm, the propagation delay of the message passing accumulates. Hence for the case where uploading capacity varies slowly w.r.t. time, distributed SWaF is a good choice, while for the case where uploading capacity varies fast, SWaF in a centralized fashion may fit well.

4. NUMERICAL RESULTS

To verify the performance of the proposed algorithm, we set up P2P sessions with the following parameters similar to [7], video playback rate $R_0 = 300$ kbps, uploading bandwidth standard variation 40 kbps. The average uploading bandwidth is $C = \sum_{j=0}^{N} C_j/N$. We assume each peer has the same utility function. In [7], the advantage of utility-based scheduling compared with utility-blind scheduling has been illustrated and it is not our focus in this work. Here we focus on how the distributed SWaF algorithm works.

Consider a system with a source (peer 0) and 6 peers (peer $1, \cdots, 6$). Assume the scheduling interval $T = 4$ seconds. Assume the initial playback buffer reserves for peer $1, \cdots, 6$ are (9,7,8,11,12,10) seconds, respectively. Assume the initial buffer ending times are $(y_1, \cdots, y_6) = (30, 24, 20, 16, 11, 6)$. The average uploading bandwidth is $C = 324.6$ kbps=$1.28R_0$.

Figure 3 shows the peer buffer content reserve states for a successive water-filling carried out by a sequence of working (serving) peers (peer 5,4,3,2,1,0). It can be seen that SWaF has the tendency to make all the peers have similar (fair) content reserve level. It is read-

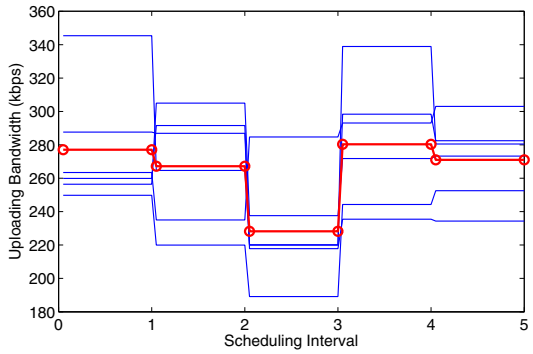

Fig. 5. Changes of uploading bandwidth.

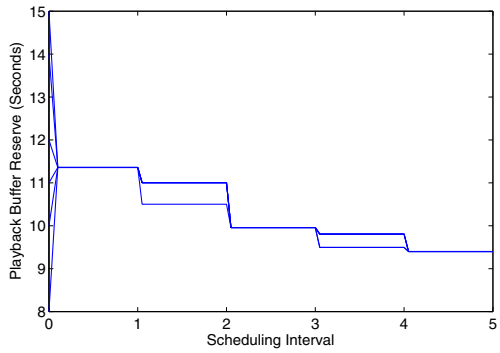

Fig. 6. Performance with time-varying uploading bandwidth.

ily verified that the greedy heuristic in [7] actually gives a solution similar to SWaF algorithm in a discrete version with segment of GoP.

Figure 4 shows the number of iterations for water-filling at each serving peer (peer 5,4,3,2,1,0), with an accuracy tolerance 2%, i.e., the bisectional search ending condition being $|\sum_{k=j+1}^{N} z_{j,k} - \beta C_j| \leq 0.02$ for all j. The average iterations over 6 peers is 4.5 in this figure which is just a random case. We run 10^5 cases for different uploading capacities, and we find an average iterations 4.58. For the same setting, the average iterations over 6 peers for the algorithm in [7] is 8, where for each iteration it has the similar computing complexity as in SWaF algorithm. Hence we can see SWaF algorithm has the advantage of low complexity.

The advantage of low complexity of SWaF can be more obvious when it is compared with the algorithm in [7] for a scheduling interval of more number of GoPs. For a scheduling interval $T = 8$ seconds, 10^5 cases of different uploading capacities are tested. An average number of iterations over 6 peers is about 5.54 with an accuracy tolerance 2%, while the heuristic in [7] needs 16 iterations.

Assume the uploading bandwidth varies from one scheduling interval to another, but fixed for each scheduling interval. The SWaF algorithm is effective for such system. For the same 6 peers set up, we let uploading bandwidth vary as plotted in Fig. 5, where the average bandwidth is marked in circle. Each scheduling interval T is 4 seconds. Figure 6 shows the content reserve level for 6 peers. The figure doesn't show the detailed reserve level for every water-filling, but it shows the reserve level after every 6 water-fillings in one scheduling interval. Here we assume the total time of computing and message passing in one scheduling interval is much less than T. As discussed in Remark 6, if the total time of computing and message passing in one scheduling interval is more than T, SWaF is better to be implemented centrally where a coordinator does all the computing.

5. CONCLUSIONS AND FUTURE WORK

In this paper we develop a successive water-filling algorithm (SWaF) to solve the video segment scheduling for P2P live video streaming system. We prove the optimality of SWaF and show it has lower complexity compared with the heuristic algorithm in [7]. In general, SWaF can be carried out in a centralized fashion, (note that the algorithm in [7] is centralized as well), if a peer with good computing capacity is selected as a coordinator and it does all the computing. Moreover, for the case where the propagation delay of the message passing is small compared with the time interval for uploading capacity to vary, SWaF can be carried out in a distributed fashion where the optima can be found by each peer solving its own problem with some message passing over peers. The numerical results show the effectiveness of SWaF algorithm.

For future work, we will investigate alternative distributed algorithms. We have investigated a price based distributed algorithm similar to the algorithm in [5], but it seems a slow convergence. We will study more and compare it with SWaF. In addition, we will include the end-to-end transmission delays in the system modelling and simulations. We will also investigate richer video adaptation schemes and quality metrics in conjunction with underlying P2P distribution scheme, and work towards a graceful degradation in playback when system resources are limited.

6. REFERENCES

[1] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[2] V. Fodor and G. Dan, "Resilience in live peer-to-peer streaming," *IEEE Communications Magazine*, 45(6), pp. 116-123, June 2007.

[3] X. Hei, C. Liang, J. Liang, Y. Liu and K. Ross, "Insight into PPLive: a measurement study of a large-scale P2P IPTV system," *in Proc. WWW 2006 Workshop of IPTV Services over World Wide Web*, 2006.

[4] R. Jain, "I want my IPTV," *IEEE Multimedia*, 12(3), 2005.

[5] F. P. Kelly, A. Maulloo, and D. Tan, "Rate control for communication networks: shadow prices, proportional fairness and stability," *Journal of Operations Research Society*, vol. 49, no. 3, pp. 237-252, March 1998.

[6] R. Kumar, Y. liu and K. Ross, "Stochastic fluid theory for P2P streaming systems," *in Proc. IEEE INFOCOM*, May 2007.

[7] Z. Li, J. Huang, and A. K. Katsaggelos, "Content reserve utility based video segment transmission scheduling for Peer-to-Peer live video streaming system," *in Proc. 2007 Allerton Conference on communication, control and computing*, Oct. 2007.

[8] J. Mo and J. Walrand, "Fair end-to-end window-based congestion control," *IEEE/ACM Transctions on Networking*, vol. 8, no. 5, pp. 556-567, 2000.

[9] V. N. Padmanabhan, H. J. Wang, and P. A. Chou, "Resilient Peer-to-Peer streaming," *in Proc. IEEE International Conference on Network Protocols*, 2003.

[10] PPLive system, URL: www.pplive.com

[11] PPStream, URL: www.ppstream.com

[12] D. Qiu and R. Srikant, "Modeling and performance analysis of BitTorrent-like Peer-to-Peer networks," *in Proc. ACM SIGCOMM*, Portland, OR, Sept. 2004.

[13] T. Small, B. Liang and B. Li, "Scaling Laws and Tradeoffs in Peer-to-Peer Live Multimedia Streaming," *in Proc. ACM MULTIMEDIA*, 2006.

[14] X. Zhang, J. Liu, B. Li and T.-S. P. Yum, "CoolStreaming/DONet: A data-droven overlay network for Peer-to-Peer live media streaming," *in Proc. IEEE INFOCOM*, 2005.