Priority-based Chunk Scheduling Algorithm for Large Scale P2P Live Streaming System

Hongyun Yang[1], Ruimin Hu[2]* and Xuhui Chen[3]

[1]*National Engineering Research Center for Multimedia Software, Wuhan University, Hubei, 430072, China*
E-mail:yhycxh@gmail.com

[2]*Computer School, Wuhan University, Hubei, 430072, China*
E-mail Hrm1964@163.com

[3]*College of Mathematics & Computer Science, Wuhan Textile University, Hubei, 430073, China*
E-mail: wh.chxh@gmail.com

Abstract: Chunk-based mesh-pull Peer-to-Peer (P2P) streaming is being considered as the most promising approach to deliver real-time video to large scale users over the Internet. Chunk scheduling is one of the key components. Currently the majorities of chunk scheduling approaches focus on receiver side's chunk/peer selection strategies. And most of previous research works neglect the service order and available uplink bandwidth allocation problem at supplier side, which will cause the user's video quality descending in overloaded operating environments. In this paper, we first propose the supplier side chunk priority model considering requested chunks' urgency of playback and rarity. Then based on the model, we formulate the supplier side scheduling problem as a linear programming problem and derived a greedy bandwidth resource allocation algorithm to solve it. The main contributions of this paper are: i) we incorporate the relative urgency of playback and rarity of chunks into designing the priority model; ii) we transform the supplier side scheduling to a bandwidth allocation problem and propose a greedy resolution. The extensive experiments demonstrate the proposed scheme effective in improving the quality of experience of end users in overloaded operating environments comparing to the FCFS (First Come First Service) scheme.

Key Words: Supplier side scheduler, P2P live streaming, Bandwidth resource allocation, Relative urgency of playback

1. Introduction

With the widespread of broadband residential access, Peer-to-Peer (P2P) live streaming has become a popular service in the Internet. Recently a majority of commercial P2P live streaming systems, such as PPLive[1], Sopcast[2], UUSee[3], et.al have been successfully deployed and attracted millions of users all over the world. And all of these systems adopt chunk-based mesh-pull design [4].

In chunk-based mesh-pull P2P streaming systems, users (also called peers in the paper) organize themselves into mesh overlay topology in a centralized or

decentralized matter. Each video sequence is divided into chunks (also called segments) and then chunks are streamed from multiple suppliers to a receiver along the mesh adopting certain chunk scheduling strategies. Currently, researchers mainly focus on the design of receiver side scheduler, which means that the receiver decides which chunk will be selected from which neighbor and neglect the design of the supplier side scheduler due to the fact that P2P live video streaming is still in its early stages[4] and many problems have not been solved well on chunks/peers selection of the receiver side. However, as we have known, there are two sides to make up an integrated scheduling process. One is the receiver[5] who estimates the bandwidth of suppliers and coordinates the chunk transmission among multiple suppliers. The other is the supplier side scheduler. As a supplier with finite uplink bandwidth receivers many chunks requests from multiple neighbors simultaneously. The supplier peer must determine which request should be served first and how to allocate its available uplink bandwidth to its request neighbors. Therefore neglecting the strategies of supplier side scheduling will cause the service response time increment in overloaded network environments. Figure 1 shows the affection of different scheduling schemes. We suppose when a peer does not receive a chunk before its playback deadline, the peer can freeze the playback with the most recently displayed video chunk and wait for the missing chunk to arrive. Assumption at some point in the streaming session, P is the supplier and A, B, C are three receivers who send chunk requests to P simultaneously. Each peer in the system maintains a buffer that can cache up to four chunks received from the network (assume each chunk contains 1-second video). And the number filled in every box in buffer space represents the chunk the peer has received. As we target live media streaming, the playback progresses of receivers are semi-synchronized [6]. The arrow on the box represents the playback point of that peer. The number in the box by the arrows is the requested chunk number. On the snapshot, peer A requests the 2nd chunk from peer P, peer B and peer C request the 3rd chunk respectively from peer P. Given peer P only upload one chunk one second. In figure 1(a), the service order of peer P is A->B->C, that's means peer A receives the 2nd chunk in time while peer B has to wait one second before the chunk 3rd arrives. However if the service order is C->B->A in figure1 (b), peer B has to wait one second before chunk 3rd arrived while peer A has to wait two seconds before it receives chunk 2nd. So for the second scene, the whole wait time increases. Because the time lags or semi-synchronized between peers in live streaming systems exists, one supplier peer may receive several chunk requests with a different degree of urgency. Therefore it is necessary to design rational service order to help decreasing the wait time while improve the systems throughput and users' playback quality.

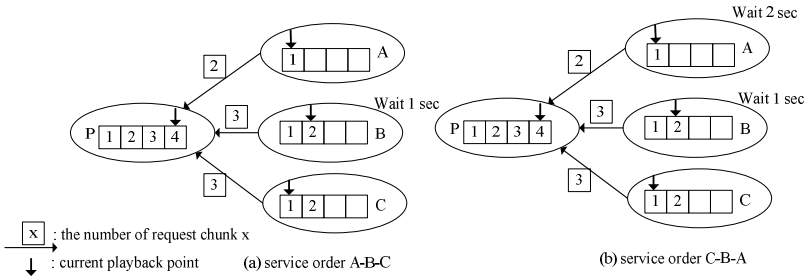

Fig. 1. The affection of different service order of supplier P

In this paper, considering different requested chunks have different urgency of playback and even for the same chunk request, the urgency of playback is different to different peers due to the time lags, we firstly introduce the chunk priority model of supplier side. Then based on the model we formulate the supplier side scheduling as linear programming problem and transform it as bandwidth allocation problem and propose a greedy algorithm to solve it. Our main contributions are as follows: first, we incorporate the relative urgency of playback and rarity of chunks into designing the priority model to achieve the tradeoff; second, we propose a greedy bandwidth allocation algorithm. The extensive experiments demonstrate the proposed scheme effective in improving the quality of experience of end users in overloaded operating environments.

The rest of this paper is organized as follows. Section 2 discusses some related works in this area. Before discussing the supplier side scheduling problem, we describe the buffer structure used in this paper in Section 3. Section 4 formulas the supplier side scheduling problem as linear programming problem and proposes greedy algorithm to solve it. Section 5 discusses our evaluation methods and metrics and then presents the evaluation results. Finally, Section 6 concludes our work and points out the further works.

2. Related Works

Recently, several supplier side scheduling algorithms have been proposed in the literature for mesh-pull based P2P live streaming. FCFS (First Come First Service) is the most widely used method adopted by the majority of current P2P streaming systems [7-9] for its simplicity and easy to realize in a direct or indirect way. However, this FCFS method neglects the urgency of playback and the rarity of requested chunks and will cause the supplier side can't response a large number of chunks requests in time, which further increases the wait time of peers and degrades the quality of experience of users.

Therefore, some studies take the rarity and urgency of playback of requests chunks into account. PULSE [10] is an adaptive, incentive-based, unstructured P2P live streaming system. In PULSE, supplier sides use "least sent first, Random" strategy. Each peer keeps a count of how many times it has sent each requested chunk. The one that has been sent the least number of times is chosen to be sent first and in case of a

tie, the chunk is selected randomly. This strategy is conducive to increase the degree of data sharing. However it doesn't take the playback urgency of missing chunks into account and may cause the number of chunks arrived after the playout deadline increment (the chunk loss ratio increment). LayerP2P[4] combines layered video, mesh P2P distribution, and a tit-for-tat-like algorithm into a P2P live streaming system. In LayerP2P, the supplier side peers maintain a different request queue for each receiver with regular and probing request types, which reflect the chunks' different importance for the reconstructed video and apply the tit-for-tat-like strategy on determining the selection probability of which receiver should be served first. And for a particular receiver, it uses a prioritized random scheduling algorithm. Regular requested chunks will be served on time with high probability and probing requested chunks will be served when the suppliers have surplus upload bandwidth allocated to this request peer. The approach achieves high efficiency, provides differentiated service and adapts to bandwidth deficient scenarios. And the more related to our work, bin [11] proposes a priority-based supplier side scheduling scheme for a VOD(video On-Demand) system. In [11], the buffer is divided into urgent region and non-urgent region by pre-fetch window and greedy strategy (or the nearest deadline first strategy) and rarest first strategy adopted respectively when define the chunk model of supplier side scheduling. In this paper, considering single-layer video is still the mainstream of the transmission, we focus on supplier side scheduling for single-layer P2P live streaming. And considering that rarity and relative urgency of playback of requested chunks for different receivers are two most important characters which affect the data scheduling methods at receiver side, we incorporate these factors into designing the chunk model of supplier side scheduling. So as to the receiver side scheduling strategy, we use latest useful chunk, random peer mechanism ,which has been proved to achieve dissemination at an optimal rate and within an optimal delay[12].

3. Buffer Structure

In mesh-based P2P streaming, the video is divided into chunks of uniform length, e.g., 0.1 second. Each chunk has a unique sequence number. A chunk with lower sequence number contains video with earlier playback time. Each chunk is disseminated to all peers through the mesh. Since chunks may take different paths to reach a peer, they may arrive at a peer out of order. For continuous playback, a peer normally buffers received chunks in memory and put them back in order before presenting them to its video media player and then buffered chunks can be uploaded to its neighbors. Generally a peer buffers up to a few minutes worth of chunks. And in the ideal case for live streaming, the sequence number of buffered chunks increases steadily as the video playback progresses. However, in fact chunks stored in peers' buffer are not continuous. Some of them may have been recently played, some may be downloaded from neighbors but not played and the others will be scheduled to play in future. We use buffer map (BM) to represent the availability of chunks in the buffer just as [6]. In this paper, assume each chunk contains 1-second video, the buffer size set M seconds. And a buffer map contains a 0-1 sequence of M bits with bit 1 indicating that a chunk is available in peer's buffer and 0 otherwise. The first chunk's

sequence is recorded by another two bytes as the buffer offset. In order to get the available information of chunks, peers exchange BM messages with their neighbors periodically. Figure 2 shows the structure of a peer' s buffer map.

In the paper, we design a sliding window to divide the buffer widow into three parts: expired region, urgent download region and non-urgent download region. Chunks in expired region represent that they have been played or missed the playback deadline and still can be used for other peers given the time lags between peers. The chunks in the urgent download region have higher priorities than those in other regions and will be scheduled in the next period if they are not received yet. And chunks which are closer to the playback point have higher priority. If peers have more bandwidth to receive chunks besides the chunks in urgent download region, they will request missing chunks in non-urgent download region from their neighbors.

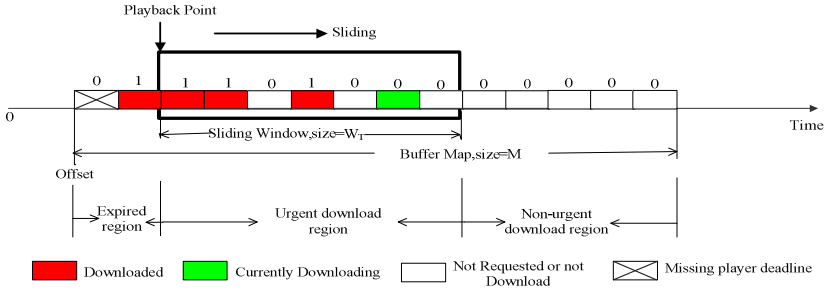

Fig 2 the structure of buffer map and sliding window

4. Problem Formulation and a Greedy Algorithm

4.1 Supplier Side Chunk Priority model

In a mesh-pull based P2P live streaming system, because of the real-time nature of live streaming, each media chunk has a playback deadline, which can be different from one peer to another by a few minutes. In addition, due to the deployment of buffering mechanisms, it is possibility of playback time lags among peers, which some peers watch frames in a video minutes behind other peers. So for the same missing chunk requested from different peers, the urgency of playback is different (we call it relative urgency of playback of chunks). For example, in figure 1, the 3nd chunk will be playback soon on peer B while to peer C, it is not emergent because the 1st chunk is now played on peer C.

In this paper, in order to improve the utilization of available uplink bandwidth of supplier peers, we model the supplier side chunk priority as following. Firstly we introduce some definitions and notations, which are summarized in Table 1.

Table 1. Notations

Notation	Description
U_p	The upload capacity of peer p, kbits/s
<p,i>	The overlay link between peer p and peer i
U_{pi}	The maximum bandwidth capacity of link <p,i>
u_{pi}	The available uplink bandwidth peer p allocates to peer i
NEI_i	peer i's set of neighbors
α	playback urgency factor
l_{ij}	The round trip time between peer i and peer j
T	The service time interval
W_T	The sliding window size, in seconds
d_{ij}	The playback deadline of chunk j on peer i
C_{ij}	The time of peer i sending the request of chunk j
m_i	The number of neighbors of peer i
S_{ij}	The chunk of Number j in the buffer of peer i
g_{ij}^k	g_{ij}^k=1 represents chunk S_{ij} in the buffer of peer k, otherwise g_{ij}^k=0
a_{ij}^k	a_{ij}^k=1 represents chunk S_{ij} in the sliding window of peer k, otherwise a_{ij}^k=0
h_{ij}	h_{ij}=1 represents peer i has received chunk j, otherwise h_{ij}=0
v_{ij}^k	binary variable, v_{ij}^k=1 represents the supplier i serve the chunk j request to peer k, otherwise v_{ij}^k=0
C	Chunk size, Kbits
m	The maximum length of request queue

According to the real-time requirement in live streaming, the priority of requested chunks is defined as formula (4.1).

$$P_{kj}^{\ q} = \begin{cases} 0, d_{qj} - C_{qj} - l_{qk} < 0 \\ \alpha \times \dfrac{W_T}{d_{qj} - C_{qj} - l_{qk} + 1} + (1-\alpha) \times \left(\dfrac{\sum_{i=1}^{m_k} (1 - g_{kj}^i) a_{kj}^i}{\sum_{i \in NBI_k} h_{ij} + 1} \right), d_{qj} - C_{qj} - l_{qk} \geq 0 \end{cases} \tag{4.1}$$

Where $P_{kj}^{\ q}$ represents the priority of requested chunk j sending from receiver q to supplier k. $d_{qj} - C_{qj}$ denotes the residual time before playback of the requested chunk j on peer q. $d_{qj} - C_{qj} - l_{qk}$ represents the maximum wait time of receiver q for chunk j, which is the serving deadline of the chunk

request and equals the surplus time subtracting the round-trip delay between peer q and peer k. $\sum_{i=1}^{m_k}(1-g_{kj}^i)a_{kj}^i$ denotes the times chunk j has been requested from neighbors during the period. $\sum_{i\in NBI_k} h_{ij}$ represents the neighbor number of peer k which have received the chunk j. As shown in (1), $P_{kj}^{\ q}$ is the sum of two terms. The first term represents the relative urgency of playback and the second term represents the value of the scheduled chunks, which uses the chunk request times and local scarcity ratio to estimate. And the urgency factor α satisfied $0 \le \alpha \le 1$.

The priority model of request chunks has the following characteristics: 1) for the receiver peer, the closer to the local current playback position, and the higher of the priority of chunk; 2) For the supplier peer, the chunks which is possessed by few neighbors and requested by more neighbors have higher priority. The model takes into account the different peers' play lag or semi-synchronized phenomenon, and the real-time and rarity characters. So in a certain extent, this scheme reduces the probability to send useless chunks to requested peers after the chunks' playback deadline and achieve high usage of available upload bandwidth of suppliers.

4.2 Supplier Side Scheduling Problem Formulation

We assume that a set of k peers $N = \{N_1, N_2, .., N_k\}$ choose an existing peer P as the supplying peer. Up is the maximum upload bandwidth capacity of peer P. we use u_{pi} to represent the available bandwidth[18] peer P allocates to peer Ni along the overlay link <P,N_i> and U_{pi} to represent the maximum bandwidth of overlay link <P, Ni>,which equals to the end-to-end path bandwidth of physical link[13]. We use $r_i = \{r_{i0}, r_{i1}, ..., r_{im}\}, \forall 1 \le i \le k$ to represent the requested chunks sequences sending from receiver peer i. v_{ij}^k is a binary random variable that depends on whether the requested chunk is served. v_{ij}^k is defined as follows :

$$v_{ij}^k = \begin{cases} 1, \text{if supplier peer i sends chunk j to request peer k} \\ 0, \quad \text{otherwise} \end{cases} \tag{4.2}$$

For each supplier peer, the goal is to maximize the sum of priority value of all served chunks, while not violating the constraints on peer and link bandwidth. Given the above definitions, the supplier side scheduling problem can be formulated as follows[11]:

$$z_p = \max \sum_{i=1}^{k} \sum_{j=1}^{m} p_{pj}^{i} v_{pj}^{i} \quad (4.3)$$

$$\text{s.t.} \sum_{i=1}^{k} u_{pi} \le U_p \quad (4.3a)$$

$$u_{pi} \le U_{pi}, \forall 1 \le i \le k \quad (4.3b)$$

$$\sum_{j=1}^{m} v_{pj}^{i} \le \frac{u_{pi} \times T}{C}, \forall 1 \le i \le k \quad (4.3c)$$

Equation (4.3) is the objective function of supplier peer P, which is to maximize the total priority value of the supplier peer P during the next service period. Equation (4.3a), (4.3b) and (4.3c) are the outbound bandwidth constraints. Equation (4.3a) denotes that the sum of allocated uplink bandwidth to receivers can't excess the outbound bandwidth of supplier peer P. Equation (4.3b) denotes the allocated uplink bandwidth from supplier peer p to receivers can't excess the maximum available link bandwidth between supplier peer and each receiver. Equation (4.3c) denotes the number of chunks served is limited by the link available bandwidth between the supplier and each receiver.

In fact, the problem the supplier side deals with chunks requests can be transform as a bandwidth resource allocation problem. The resource allocation process model can be described as figure 3.

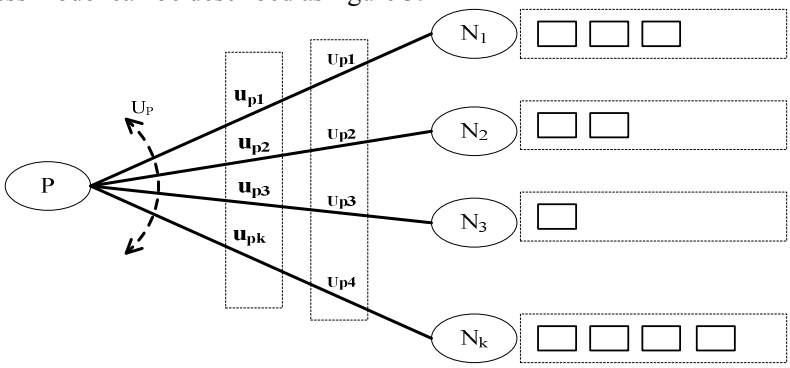

Fig.3 the sketch of resource allocation process at supplier side P

In the paper, we assume chunks have the same size. And the maximum number of chunks supplier peer P serves for receiver i is calculated as formula (4.4).

$$SC_{P \to i} = \sum_{j=1}^{m} v_{pj}^{i} \le \frac{u_{pi} \times T}{C} \le \frac{U_{Pi} \times T}{C} \quad (4.4)$$

The maximum number of service chunks during T service interval of

supplier peer P can be calculated as follows: $SC_p = \frac{U_p \times T}{C}$ .The problem of supplier side scheduling equals to the picking balls problem discussing as follows. Given there are k bags. Bag j has NM_j balls. The maximum number of balls picked out of bag j is M_j. The weight of each ball is already well-known. Let's $W_{i1}, W_{i2}, ..., W_{im} (1 \le m \le NM_i)$ represent the weight of each ball in bag i. Then the problem can be transformed as how to pick SC_p balls so as to achieve the maximum weight. We use greedy strategy to achieve the optimal solution of the problem. First, we sort all the balls in k bags in descending order according to their weight, denotes as p^{sort}. Then we select the first SC_p balls from p^{sort} and class them according to their original bag number. Let Q_j denote the number of selected balls from bag j. Suppose Q_j is larger than M_j , we will discard the latter M_j-Q_j balls from bag j and select the following M_j-Q_j balls from p^{sort}. The process continues until SC_p balls are selected without violating the maximum constraints of each bag.

4.3 A Greedy Bandwidth Resource Allocation algorithm (GBRA)

The algorithm runs as follows. Through exchanging buffer map in period, peers know which chunks have been in its neighbors' buffer or not (also called the chunks' distribution of each other). When a peer receives several chunks requests from its neighbors, the peer launches the supplier side scheduler. First the supplier computes whether it is necessary to service the chunk request according to requested chunk's deadline of playback, the sending time and transmission delay between the receiver and the supplier. Then after the coarse filtering, all the chunk requests are ranked according to their priority value computed as formula (4.1). And for every chunk request, it is pushed into respective response queue under link capacity constraint until the total service chunks number reached (Up*T)/C. The pseudocode of the algorithm is described as follows:

Algorithm 1 Pseudocode for GBRA algorithm at supplier p

```
Struct ChunkRequestList{
   Size_t  receiverPeerId;
   Time deadline;
   Time sendTime;
   Time receiveTime;
} set_RequestChunks;

S: temporary array to save the request chunks which satisfy the time
```

constraint

Input:
Up: the outbound bandwidth capacity of supplier peer p
Array Up[i]: the maximum bandwidth capacity of overlay link<p,i>;
T: service interval;
C: the chunk size;
set_RequestChunks: set of requested chunk sequence;

Output:
Matrix CS : the response queues for each receiver

GBRA-Algorithm:

```
GBRA (Up, set_ RequestChunks, Array Up,C,T)
CS = NULL;
CR=Null;
For j=0 to set_RequestChunks.length do
  CR = set_RequestChunks[j];
  If CR.deadline-CR.sendTime-2*(CR.receiveTime-CR.sendTime)>0 then
     ComputerPriority(CR);
     Push(S, CR);
  End If
End For
SS<—sort S according to requested chunk priority in descending order
Index = 0;
Count = 0;
qSize = 0 ;
While index < (Up*T)/C do
   For j=0 to SS.length do
     Count = CS[SS[j].receiverPeerId].length;
     qSize = (Up[CS[SS[j].receiverPeerId]]*T)/C;
     if(Count < qSize) then
        push(CS[SS[j].receiverPeerId],SS[j]);
        index++;
     end If
   end For
end While
return     CS
```

The computational complexity of the proposed method is decided by the uplink bandwidth of supplier and the length of the requested chunks sequence.

However, the number of requested chunks of each receiver is fairly small. E.g., the uplink bandwidth of a supplier is 1Mbps, the chunk size is 0.1Mbps, the service interval is 1 second, the number of neighbors of a peer is 4 to 20 and the maximum number of requested chunks is 10. Then the maximum computational times is 2000, which only takes a few tens of milliseconds on a reasonable PC. Therefore, although designing more efficient allocation algorithm is possible, we believe that the payoff will not be significant.

5. Performance Evaluations

In this section we carried out our simulation to evaluate the performance of the proposed scheme (Priority-based method) with FCFS method. In the FCFS scheme, the supplier side adopts first-in-first-out way to response chunks request.

5.1 Simulation Setup

P2PTV-sim is an open source chunk-level discrete event-driven simulator written in standard C++ developed within the Napa-Wine project[14]. Based on P2PTV-sim, we make some extension and conduct a series of simulations to study the impacts of our supplier side scheduling algorithm.

In the simulation we consider scenarios comprising N = 500 peers, if not otherwise indicated. The bandwidth distribution is shown in Table I. and we suppose the download bandwidth of peers is infinite. The default uplink bandwidth of source node is 5Mbps. We employ real-world end-to-end latency matrix (2500*2500) measured on the Internet provided by Meridian project[15]. We simply map each node pair in our simulation to each pair in the latency matrix randomly. We use fixed random neighbor selection method to construct the overlay. The number of neighbor of each peer is 20. During the simulation, the total number of delivery chunks is 200. And the receivers adopt LU/RP(Latest Useful Chunk, Random Peer)[12] scheduling algorithm.

Table 2. Bandwidth Ration Distribution Setup

Peer Type	Uplink Bandwidth	Percentage
DSL	400Kbps	45%
Cable	800Kbps	40%
Ethernet	1500Kbps	15%

The video rate of the encoder is a free parameter that we vary to enforce different values of the system load, which we use resource index [17] (RI) to describe. Table 3 shows the system's load under different video rate. The playback urgency factor α is 0.5.

Table 3. System resource index changing with the video rate

Video Rate(Kbps)	RI	Video Rate(Kbps)	RI	Video Rate(Kbps)	RI
400	1.853	700	1.059	1000	0.741
500	1.482	800	0.926	1100	0.674
600	1.235	900	0.823	1200	0.618

5.2 Simulation Results

To estimate the system's network performance, we mainly focus on the performance metrics[16]: chunk delivery latency, chunk miss ratio, peer bandwidth utilization study the impact of different design parameters and simulate a stable environment. While the whole session only persist less than 20s, only a marginal percentage of peers are expected to leave or join the system. So the effect of peer churning is neglected at first. And we will explicitly assess its impact in our future works. When all the nodes join in an initialization period, they persist in the lifetime of the streaming. All results are averaged over at least ten independent simulation runs.

Figure 4 shows the average chunk delivery delay as a function of the target video bitrate with a certain playout delay (5s) of two schemes we study. We observe that with the increment of the video bitrate, the average chunk delivery latency increases. That's because as the video coding rate increases, the chunk size increases as well and therefore the diffusion of a given chunk takes longer. When the video rate is less than 0.5Mbps, FCFS scheme is better than priority-based scheme due to its simplicity and needn't calculate. However as the video rate grows, priority-based scheme outperforms FCFS scheme. This can be attributed to the nature of priority-based scheme which uses the more effective bandwidth to distribute the more valuable chunks and pre-filtering also benefits to remedy the consumption of calculation time. When the video rate is larger than 1.0 Mbps, the chunk delivery delay descends because a large number of chunks are lost and we use the on-time arrived chunks to compute the delay. Figure 5 shows the max chunk delivery delay as a function of the target video bitrate. With the video rate grows, the max chunk delivery delay increases and the performance of priority-based scheme is better than FCFS scheme. However they all less than 5s for we set the playout delay is 5s and chunks which miss the deadline will be lost.

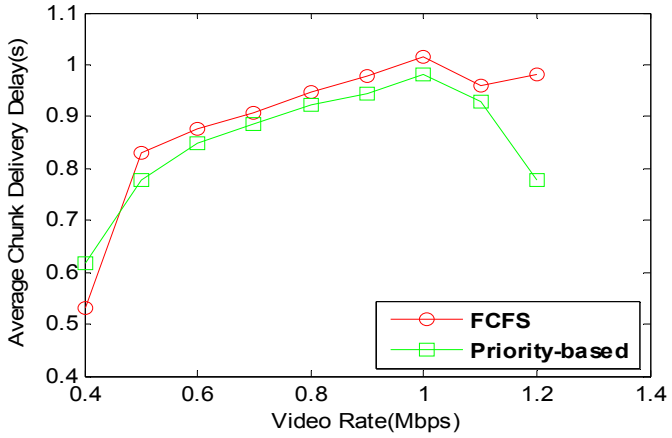

Fig.4. Average chunk delivery latency as a function of the video rate

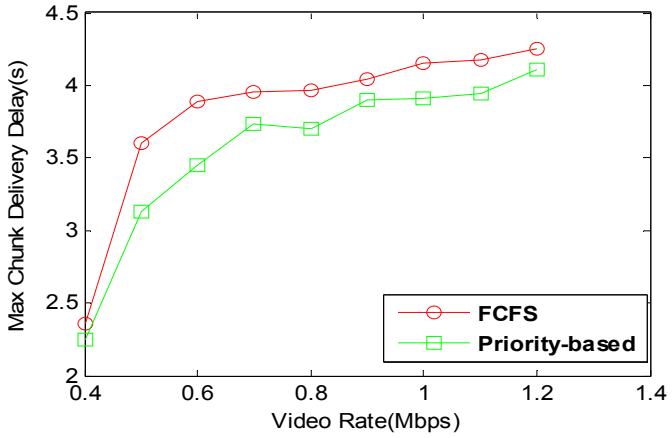

Fig.5. Max chunk delivery latency as a function of the video rate

Figure 6 shows the average chunk loss ratio as a function of the target video bitrate with a certain playout delay (5s). When a chunk is not received within its playout time, it is deemed "lost". When the video bitrate is low and the system is under-loaded, e.g. video bitrate is no more than 0.6Mbps; the chunk loss rate is less than 10%. However when the video rate grows, the system's total requirement increases while the total supply unchanged, that's the total load increases. As a result, the chunk delivery delay becomes longer which causes the number of postponing chunk increment at a given target playout delay. So the number of lost chunks depends on the media bitrate. The loss ratio in our proposed scheme is lower than FCFS scheme.

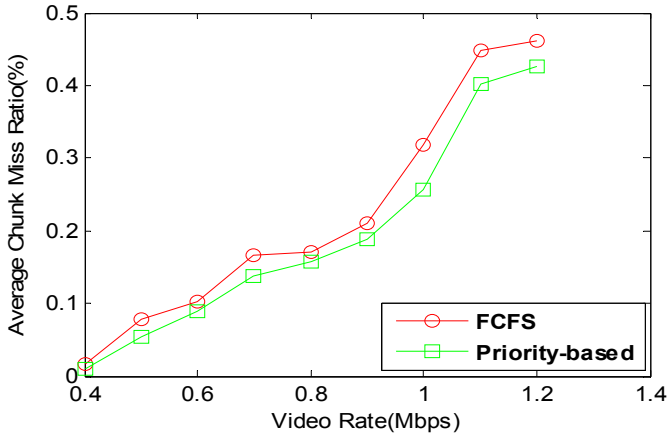

Fig.6. Average chunk miss ratio as a function of the video rate

Figure 7 depicts the average uplink bandwidth utility as a function of video rate. From figure 7 we observe when the system's bandwidth is rich, the uplink bandwidth utility increases as the video bitrate grows. But when the system's requirement is larger than peers' supply, that's the load of the system is larger than it can support, the bandwidth utility is descending fast. Especially when the video rate is 0.9Mbps, the resource index is smaller than 0.9. That's because with the increment of video rate, the chunk delivery delay increases which causes the number of useless chunks missing the playback deadline increase. So the ratio of the number of useful chunks received to total number of delivery chunks decreases. Our proposed method still outperforms the FCFS scheme.

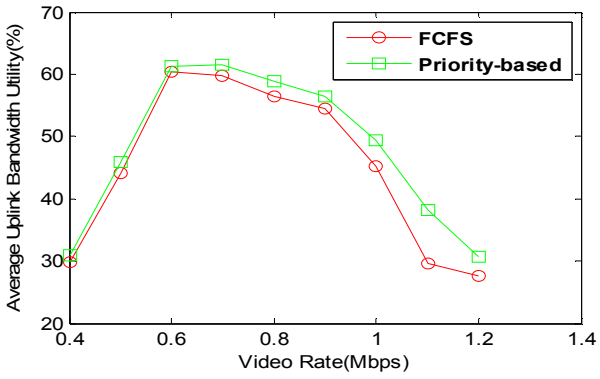

Fig.7. Average bandwidth utilization as a function of the video rate

6. Conclusion and Future Works

In this paper, we study the problem of supplier side scheduling problem in chunk-based mesh-pull P2P live streaming system. We have proposed the supplier side chunk priority model considering requested chunks' urgency of playback and rarity. Based on the model, we have formulated the supplier side scheduling problem and derived a greedy solution for it. Simulation results have shown our scheme achieving higher performance than techniques commonly used.

As a future work, we plan to pursue this work along the following directions: 1) Considering there are free-riders in the system, we will research how to incorporate incentive mechanism into bandwidth allocation; 2) flash crowds are common in live streaming, which would arise acute churn when a large amount of users joining the session when a popular program is about to begin and leaving immediately after the program finish. So it is essential to investigate how flash crowds impact our scheme and find out ways to deal with this problem.

7. Acknowledgments

This paper is supported in part by the major national science and technology special project in China (Grant No.2010ZX03004-003-03) and National Nature Science Foundation of China (No.60832002, No.61172173)

References

[1] PPLive, http://www.pplive.com.
[2] SopCast, http://www.sopcast.org.
[3] UUSee,http://www.uusee.com.
[4] Zhengye, L., Yanming, S.,Ross, K.W.,Panwar, S.S., Wang, Y., LayerP2P: Using Layered Video Chunks in P2P Live Streaming. J.IEEE Trans. on Multimedia, 11:7(2009), 1340-1352.
[5] Guangxue, Y., Nanqing, W., Jiansheng, L., et al.,Survey on Scheduling Technologies of P2p Media Streaming. J. OF Networks,6 :8(2011).
[6] Xinyan, Z., Jiangchuan,L., Bo,L., et al.,Coolstreaming/Donet: A Data- Driven Overlay Network for Peer-to-Peer Live Media Streaming. The 24th Annual Joint Conference of the IEEE Computer and Communications Societies(INFOCOM 2005),IEEE Press, New York(2005),2102-2111.
[7] Carta, A., Mellia, M., Meo, M.,Traverso, S.,Efficient Uplink Bandwidth Utilization in P2P-TV Streaming Systems. 2010 IEEE Global Telecommunications Conference (GLOBECOM 2010), IEEE Press, New York(2010), 1-6.
[8] Robert Birke, Csaba Kiraly, Emilio Leonardi et al.,Hose Rate Control for P2p-Tv Streaming Systems.2011 IEEE International Conference on Peer- to-Peer Computing (P2P) IEEE Press, New York(2011), 202-205.
[9] Hai, J., Xupin,g T., Nearcast: A Locality-Aware P2p Live Streaming Approach for Distance Education. J. ACM Trans. on Internet Technology (TOIT), 8:2 (2008), ACM New York, NY, USA.
[10] Pianese, F., Perino, D., Keller, J.,Biersack, E.W.,PULSE: An Adaptive, Incentive-Based, Unstructured P2P Live Streaming System. J.IEEE Trans. on Multimedia,9:8(2007),1645-1660.
[11] Bin Cheng,Research on Data Dissemination for Peer-to-Peer Video-on- Demand Systems, Ph.D Dissertation, Huazhong University of Science and Technology, 2009,54-58.
[12] Bonald,T., Massoulie, L., Mathieu, F., et al. ,Epidemic Live Streaming: Optimal Performance Trade-Offs. Proceedings of the 2008 International Conference on Measurement & Modeling of Computer Systems(Sigmetrics'08),Acm Sigmetrics Performance Evaluation Review, New York: Assoc

Computing Machinery ,Special Issue 1(2008), 325-336.
[13] Hefeeda, M., Habib, A.,Collectcast: A Peer-to-Peer Service for Media Streaming. J. ACM/Springer Multimedia Systems. 11:1(2005), 68-81.
[14] NAPA-WINE Project, http://www.napa-wine.eu/cgi- bin/twiki/view/Public.
[15] Meridian- Project,http://www.cs.cornell.edu/People/egs/meridian/data.php.
[16] Chao, L., Yang, G., Yong, L.,Is Random Scheduling Sufficient in P2p Video Streaming? The 28th International Conference on Distributed Computing Systems, IEEE Press, New York (2008), 53-60.
[17] Kumar, R., Yong, L.,Ross, K.,Stochastic Fluid Theory for P2P Streaming Systems. 26th IEEE International Conference on Computer Communications, IEEE Press,New York(2007),919-927.
[18] Hongyun, Y., Xuhui,C., Ruimin, H.,An End-to-End Congestion Control Approach based-on Content-Aware, J.IJMUE,4:2(2009),81-90.