Title	Data scheduling algorithm for layered P2P VoD streaming networks
Author(s)	Wen, Z; Yeung, K; Lei, ZB
Citation	The 2013 IEEE Global Communications Conference (GLOBECOM 2013), Atlanta, GA., 9-13 December 2013. In Globecom. IEEE Conference and Exhibition, 2013, p. 1730-1735
Issued Date	2013
URL	http://hdl.handle.net/10722/189903
Rights	Globecom. IEEE Conference and Exhibition. Copyright © IEEE.

Data Scheduling Algorithm for Layered P2P VoD Streaming Networks

Zheng Wen, Kwan L. Yeung
Department of Electrical and Electronic Engineering
The University of Hong Kong
Pokfulam, Hong Kong
{wenzheng, kyeung}@eee.hku.hk

Zhibin Lei
Applied Science & Technology Research Institute (ASTRI)
Shatin, Hong Kong
lei@astri.org

***Abstract* — Streaming layered video over peer-to-peer (P2P) networks has been recognized as an efficient way to address the receiver heterogeneity problem. The distinctive characteristics of layer encoded video also introduce complexities to data scheduling. In this paper, we propose a new data scheduling algorithm for layered P2P video-on-demand (VoD) streaming networks. Our algorithm consists of two parts: 1) *layer adaptation*, where peers adaptively adjust the number of subscribed layers to ensure a continuous playback of the highest possible video quality; and 2) *piece selection*, in which a peer selects a missing data piece to request based on its utility. The piece utility is calculated based on the playback constraint and the layer dependency of a piece. Through extensive packet-level simulations we show that our proposed data scheduling algorithm can effectively enhance the video playback quality.**

I. INTRODUCTION

Although the effectiveness of P2P networks in providing video streaming service to a large number of Internet users has been witnessed during the past decade, it still faces the problem of receiver heterogeneity [1][2]. Specifically, end users/receivers connect to the Internet through devices of different computational capabilities (e.g. laptops, TV set-up boxes and smart phones) and via networks of different access bandwidths (e.g. cable/ADSL networks, WiFi and cellular networks). The heterogeneous receivers require different video qualities that can best match their processing/display capabilities, and yet can make efficient use of their uplink bandwidth to help each other out in maximizing the overall system video quality.

The traditional way of solving the receiver heterogeneity problem is to group peers with similar characteristics (e.g. access bandwidth, playback quality requirement) together and form a separate P2P overlay [3][4]. The video is encoded into multiple versions of different qualities and delivered through different overlays in parallel. Despite the simplicity of the approach, the bandwidth sharing efficiency is undermined as only peers within the same overlay can exchange data pieces with each other. Inter-overlay bandwidth sharing is not possible though all peers are watching the same video. Besides, the coarse granularity and the static nature of such an overlay grouping mechanism cannot match the dynamic traffic nature of the Internet.

With recent coding efficiency improvement of layered coding scheme such as H.264/SVC [5][6], streaming layered video over P2P networks [7-9] becomes a promising approach in addressing the receiver heterogeneity problem. The basic idea is to encode the video into multiple layers with nested dependency: The base layer carries the essential information of the video and can be decoded independently to provide basic video quality. Higher layers/enhancement layers contain the data to further refine the video quality. Without loss of generality, we assume each layer of video is encoded into data pieces of the same playback duration of one second. Each data piece is a basic unit for data exchange among peers. With layered encoding, a higher layer piece can be decoded only if all of its lower layer pieces (at the same time instant) are correctly received. By subscribing to different number of layers, a peer can playback video in different qualities. Since video is encoded in a single bit stream and distributed in a single overlay consisting of all peers of the same video session, layered video streaming greatly facilitates the mutual sharing of peers' uplink bandwidth.

To fully exploit the flexibility of layered video streaming, we focus on designing an efficient data scheduling algorithm in this paper. Unlike data scheduling in conventional single layer P2P networks [2], a layered data scheduling algorithm needs to properly consider the layer dynamics. A good layered data scheduling algorithm should provide continuous playback of the highest possible video quality for each peer, avoid frequent quality switch [8], and enhance the content diversity [9] by spreading out rare data pieces.

In this paper, a new layered data scheduling algorithm is proposed. It is implemented at each peer and consists of two parts: layer adaptation and piece selection. Layer adaptation is responsible for dynamically adjusting the number of subscribed layers of a peer to match the current network traffic load while avoiding frequent quality switch. In our algorithm, this is achieved by monitoring the amount of data pieces received and ready for playback at each peer. Piece selection decides which missing piece (a piece to be played but is not yet retrieved) should be requested from a neighboring peer next. Our piece selection algorithm is designed based on a piece utility function that takes both playback constraint of a missing piece and its layering dependency into consideration.

The rest of the paper is organized as follows. In Section II, we present a brief summary of related work on data scheduling issue in layered P2P streaming network. In Section III, our proposed data scheduling algorithm is introduced. In Section IV, we present our packet-level simulation results to illustrate

978-1-4799-1353-4/13/$31.00 ©2013 IEEE

the superiority of our algorithm. Finally, we conclude the paper in Section V.

II. Related Work

The data scheduling issue in layered P2P streaming network has attracted a lot of research attentions since the pioneering work of PALS in [11]. In PALS, peers progressively evaluate the aggregate bandwidth from a set of neighbors and determine the number of layers to subscribe accordingly. The pieces in the sliding buffer window, which is within a predetermined range after the playback point (see Fig. 1 and ignore the region ΔT), are pre-fetched in a zigzag manner, where all pieces of the base/lower layer are requested sequentially before any piece of the next upper layer is requested. Piece requests are sent to neighboring peers in a weighted round-robin fashion. It can be seen that the zigzag nature of PALS gives lower priority in retrieving enhancement layers. This tends to undermine the content diversity of higher layers. In [11], the layer adaptation decision is made based on the aggregate bandwidth and the current streaming rate of the peer. The layer content availability in neighboring peers is not considered.

LION [12] and Chameleon [13] both utilize network coding to enhance layered P2P streaming performance. As a result, piece scheduling issue is of less importance and their design objective is to maximize the throughput. But how effective network coding can be in real P2P systems is still an open question [14].

In [8], a subscribed video layer is called a regular layer and a probing layer is the next upper layer a peer would like to subscribe. A peer makes simple random requests for pieces belonging to *regular* layers and applies zigzag like priorities for pieces of *probing* layers. And regular requests have higher priority over probing requests. A peer's uplink capacity and per layer bit rate are used to distinguish regular layers and probing layers. The randomness introduced by the data scheduling algorithm can increase the content diversity thereby boosting the overall system throughput. But the same randomness undermines the performance by ignoring the layer dependency requirement, where a missed lower layer piece will render the received upper pieces useless.

In layeredP2P [9], the pre-fetch buffer window is segmented into three sections. Section one is next to the playback point. The piece scheduling algorithm aims at ensuring continuous playback by making sure all missed pieces are requested with high priority. Section three is at the other end of the sliding window. In this region, subject to bandwidth availability pieces are randomly retrieved to enhance content diversity. Section two is in the middle. The scheduling algorithm tries to strike a balance between continuous playback and content diversity. This section based scheduling design accomplishes different scheduling goals in a relatively simple way. But it is difficult to justify the section size.

In [10], the utility of each piece within the pre-fetch buffer window is calculated according to the playback time constraint, the buffering status and the layer dependency. The piece selection is formulated as a knapsack problem. The greedy solution is recommended for its simplicity and run time

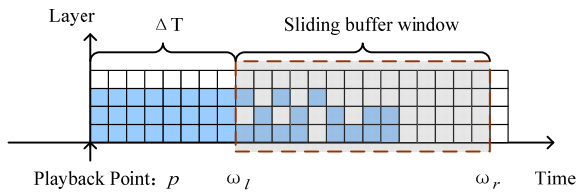

Figure 1. Example of sliding buffer window

efficiency. However, the greedy solution which *strictly* orders the pieces to be requested according to their utility values eliminates the scheduling randomness. This undermines the content diversity. This observation also inspires us to develop a utility based piece selection algorithm that increases content diversity.

III. Our Data Scheduling Algorithm

Our data scheduling algorithm consists of two sub-algorithms, which are responsible for layer adaptation and piece selection respectively.

A. Layer adaptation:

The main design goal of the layer adaptation phase is to maximize the playback video quality while preventing frequent short-term quality fluctuations/jitters [9]. The key issue is to accurately gauge the available capacity in the network and to judiciously adjust the number of subscribed layers. Given the unpredictable network bandwidth fluctuation and the heterogeneous access bandwidth of different peers, our design is based on a relatively simple mechanism as detailed below.

Let each peer be equipped with a sliding buffer window for prefetching data pieces as shown in Fig. 1. Unlike the sliding buffer window adopted in [1,2,8,9,12], our buffer window slides forward according to the piece buffering status rather than the playback process. That is, whenever the pieces of subscribed layers are successfully retrieved, the buffer window slides open and drifts away from the current playback point. Fig. 1 gives an example where the buffer window, denoted as the shadow area surrounded by dotted line, slides to the right hand side as *all* the pieces of the subscribed layers (3 in this case) are successfully received. So the buffer window is no longer bounded to start from the playback point (p). Instead, there is a gap of ΔT (playback) seconds, between the current playback point p and the starting point (left boundary) of the buffer window, ω_l.

Allowing the buffer window to drift away from the playback point has some advantages. First, in a P2P VoD streaming system, complete video is stored at the streaming server which is always available for sending. Sliding the buffer window allows a peer to make full use of the otherwise wasted downlink capacity to retrieve pieces for future playback, as well as sharing its pieces with others. Second, the value of ΔT is a good indication of the available/excess bandwidth in the network.

In our proposed layer adaptation algorithm, a peer monitors the value of ΔT and enters a layer-increase probing period when ΔT is greater than a predetermined threshold γ. And at the same time, it marks the current left boundary of the buffer

window $\chi = \omega_l$. From this moment onwards, a peer is in the probing phase and is allowed to request pieces of the next (not-yet-subscribed) upper layer, or the probing layer. By the time when the playback point coincides with the marked point (that is when $p == \chi$), this signals the end of the probing period. Based on the occupancy of an *assessing window*, the peer has to decide if the tentative layer-increase is successful or not. The assessing window is designed to be of the same length as the buffer window but starts from χ. Its *height* is the current subscribed layer plus one, i.e. the probing layer. If the occupancy of the assessing window is more than η percent, the layer-increase is deemed successful. The peer increases its subscribed layer by one. Otherwise, the layer-increase fails and the peer stops requesting pieces in the probing layer for a given time period. Fig. 2 depicts an example, where Fig. 2(a) shows the start of a probing period, Fig. 2(b) and Fig. 2(c) illustrate the cases of successful and failed layer-increase, respectively. Note that the layer probing process could only affect the pieces that are ΔT seconds away from current playback point. This ensures both the video playback continuity and quality during the layer probing process.

Note that the setting of γ and η is very important in tuning the performance of the layer adaptation algorithm. Specifically, γ is the pre-determined threshold of ΔT. It is closely related to how easily a peer can activate the layer probing process. In addition, it also implies the amount of time within which the continuity and quality of video playback is guaranteed. When γ is smaller, the layer probing process is triggered more easily. Although this helps to absorb the extra bandwidth promptly, it is at the risk of introducing more quality jitter when the bandwidth supply is not sufficient. While with a large value of γ, a peer will be more conservative in probing additional layer. This helps to reduce the quality jitter and enhance playback continuity but it tends to compromise the playback video quality in terms of average layers received on playback. On the other hand, η implies the difficulties for a peer to succeed in probing additional layer. A smaller η makes it easier for the layer probing process to be successful. But it may also lead to severer quality jitters if bandwidth supply is not sufficient for streaming one more layer. Whereas, when η increases to a large value, peers are less likely to accomplish the occupancy requirement at the end of the probing period. They are thus restricted in streaming the current subscribed layers which helps to avoid quality fluctuation. But, the average layer on playback will be significantly affected. The value of γ and η are determined by simulations in Section IV. Due to the space limit, we present only the guideline on properly setting these parameters in this paper.

There is also a layer-decrease function running in parallel. Specifically, on the playback of the video, a peer checks the fraction of received pieces of all subscribed layers within the buffer window (of size $\omega_r - \omega_l$) next to the current playback point, i.e. $[p,\ p + |\omega_r - \omega_l|]$. If a peer cannot receive more than η percent of those pieces, the number of subscribed layers is decreased by one.

One additional issue is to determine the initial number of subscribed layers when a peer joins the system. In our algorithm, an bootstrap period of τ seconds is given to a new peer. During this period, the peer should try its best in requesting pieces to fill an initial buffer window of τ seconds. The pieces in the initial buffer window are requested in the zigzag order starting from the base layer. Within each layer, pieces closer to the starting point of the video are requested first. When the bootstrap period ends, a peer will subscribe to the highest layer for which *all* the pieces within the initial buffer window have been received. (Note that we have set the length of initial buffer window to be the same as the bootstrap period for simplicity.) . If pieces of all available layers within the initial buffer window are received before the bootstrap period ends, a peer subscribes to all the available layers and starts the video playback earlier.

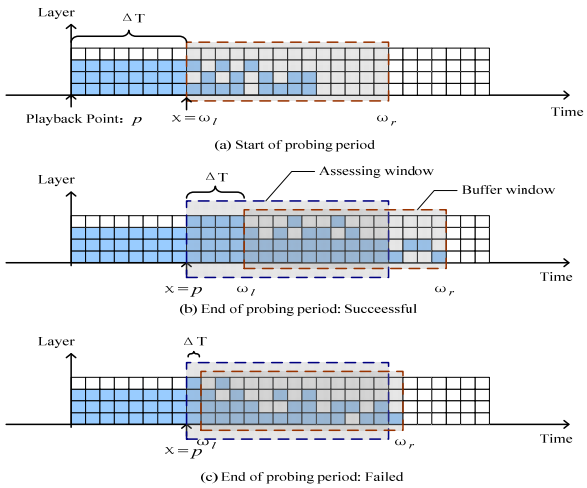

Figure 2. Example of subscribed layer increase

B. Piece selection:

For a given number of layers that a peer has subscribed to, the piece selection algorithm determines which missing pieces in the sliding buffer window are to be requested next. With layered video, the importance of a missing piece is jointly determined by its time importance, which depends on when the piece is needed for continuous playback, and layer importance, which is due to the fact that a upper layer piece cannot be decoded if any of its lower layer pieces is missing (at playback).

With the above considerations in mind, we propose a piece utility function similar to that in [10]. Specifically, for each missing piece in the buffer window, say a piece at time t_i and layer j, $piece_{ij}$, its utility U_{ij} is given by:

$$U_{ij} = \frac{\lambda^{L-j}}{S_j(t_i - t_p)^{\alpha}}, \quad j \in [1, L] \tag{1}$$

where L is the maximum number of video layers, $\lambda > 1$ is the layer importance weighting factor, S_j is the number of packets of a layer j piece, t_i is the playback time of the missing piece, t_p is time instance of current playback point, and α is the time importance weighting factor.

Given the indispensable role played by the base layer, the highest value of layer importance (λ^{L-j}) is assigned to the base layer to ensure its priority. For enhancement layers, the layer

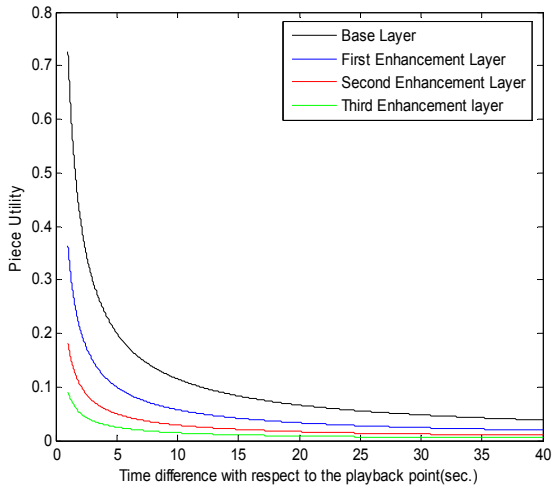

Figure 3. Piece utility value as a function of time difference $t_i - t_p$

importance value decreases with the rise of the layer index. The rationale is to ensure the in-order delivery of enhancement layer so as to avoid the un-decodable event, where enhancement layer pieces have been received but a lower layer/base layer piece is missing at playback time.

In (1), $t_i - t_p$ represents the amount of time for $piece_{ij}$ to be received before playback. As an example, Fig. 3 depicts the piece utility value as a function of time difference $t_i - t_p$ for pieces in different layers (where $S_j = 22$, $\alpha = 0.8$ and $\lambda = 2$). We can see that for pieces with the same playback time, a lower layer piece has a higher utility. This difference is more significant among pieces closer to the playback point. While for pieces of the same layer, higher utilities are given to those closer to the playback point. This emphasizes the playback importance of the missing piece. By properly combining both time and layer importance of a piece, we can show in Section IV that the utility function ensures the continuous video playback at a high video quality.

If all peers use the same utility function (1), peers with similar playback time tend to request the same piece simultaneously, causing unnecessary content bottleneck at the (few) peers with the right piece. To address this, we propose to request pieces according to their importance in a probabilistic fashion. Specifically, the probability for $piece_{ij}$ to be selected is proportional to its utility value:

$$Prob_{ij} = \frac{U_{ij}}{\sum_{i \in I, j \in J} U_{ij}} \qquad (2)$$

where I and J refer to the sets of time index and layer index within the buffer window, respectively.

Although randomness introduced by the probabilistic piece selection improves content diversity, it may affect the playback continuity. To address this, we enforce a rule of always requesting the 5 pieces next to the playback point at the base layer with the highest priority and set the timeout value for requests of these pieces to be half of the normal piece request.

Note that we adopt similar piece utility calculation for piece selection in [10], our algorithm differs from [10] in that: a) the weighted download probability on whether a piece is received in time and decodable is not considered in our proposed algorithm. Because, following calculation in [10], if any piece is missing or regarded as not possible to be received in time, its weighted download probability will be set to 0. Consequently, the utilities of all pieces of later time at higher layers may all be set to 0. If that happens, those pieces will not be retrieved. b) In our algorithm, a peer selects a piece probabilistically based on their utility, which avoids the deterministic piece request made in greedy solution in [10].

IV. Performance Evaluations

A. Performance Metrics

We evaluate the performance of our data scheduling algorithm by packet level simulations. We focus on two performance aspects: playback continuity and video quality. The playback continuity is measured by *the number of stops* and *total caching time* [15]. The number of stops records the total number of stalling events due to lack of piece to play; and the total caching time sums up all time periods when a peer suspends its playback to wait for the missing video piece to arrive. To assess the perceived video quality, we measure the *average layers on playback* and *quality jitter rate*. A quality jitter is due to the change of number of layers on playback, which can be a "burst jitter" or a "drop jitter" [9]. The quality jitter rate is the ratio of total number of quality jitter to the total pieces throughout the streaming session.

B. Simulation Setup

Our simulation is carried out using a packet level P2P simulator built on top of NS2. The correctness of the simulator has been verified using real P2P traffic data [16]. Without loss of generality, we assume all peers are connected to a central router. Peers in the network are equally divided into two groups, with uplink capacity of 1.5 Mbps and 0.75 Mbps. When a session starts, peers join the streaming session randomly at a rate of 5 peers per second until there are M peers in the system. Peers will stay in the network throughout the simulation. There is only a single VoD server in the system that stores the original video file. The video file is encoded into 4 layers. Each layer is segmented into pieces of 1 second playback. A piece is further segmented into 22 packets for transmission. Each peer in the network keeps a list of 20 neighbors. Peers exchange their buffer maps (information of pieces they have retrieved) every 1 seconds. When a peer selects a piece to request, it identifies the neighbors that hold the piece using the buffer maps stored. If there are multiple such neighbors, a peer randomly selects one to make a piece request. At any time, each peer can have up to N pending/on-going piece requests and among them, no more than D requests should be sent to the same peer. If a requested piece can not be retrieved within T sec. (piece timeout value), the requesting peer will allocate the piece request to the other neighbor that holds the piece. Each peer starts video playback after finishing the initial layer subscription process and resumes the playback as long as the base layer piece for the next second is available,

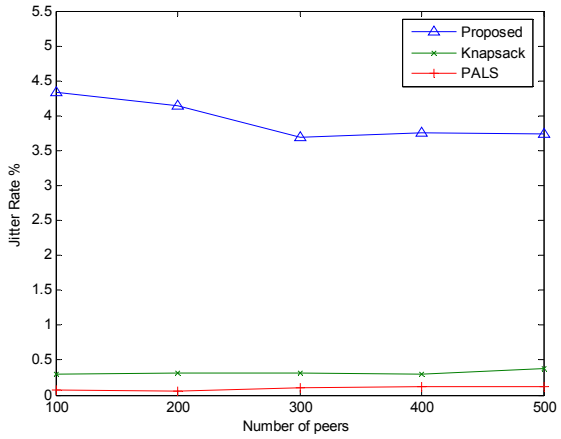

Figure 4. Jitter rate performance

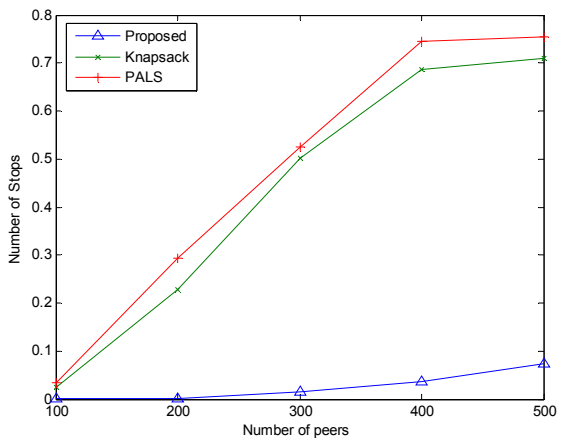

Figure 5. Number of stops performance

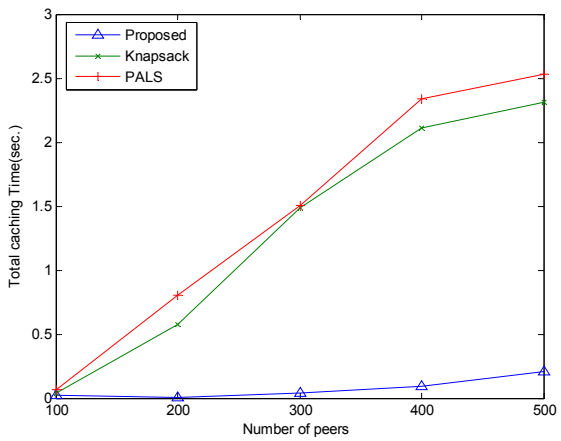

Figure 6. Total caching time performance

otherwise the playback will be suspended until the next five pieces are retrieved. Other parameters used in the simulation are summarized in Table I.

TABLE I. SIMULATION PARAMETERS

Simulation Time	300 sec.
Avg. Inter-arrival Time	4 sec.
Piece Timeout value (T)	3 sec.
Max # Requests to a Specific Peer (D)	2
Max. # Simultaneous Requests (N)	15
Buffer Window Size	30 sec.
Group I Peer Bandwidth (Down/Up)	3/0.75 Mbps
Group II Peer Bandwidth (Down/Up)	3/1.5 Mbps
Server Bandwidth (Down/Up)	5/5 Mbps
Streaming Rate of Each Layer	256Kbps
Piece Timeout Value	3 sec.
τ	10 sec.
γ	15 sec.
η	0.8
α	0.8
λ	3
S_j	22

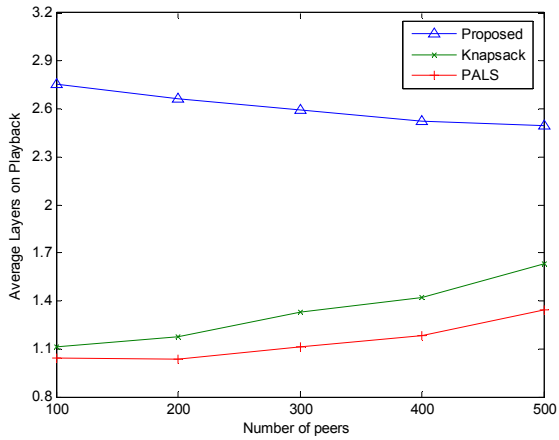

Figure 7. Average layer on playback performance

Our proposed data scheduling algorithm is compared with the PALS scheduling algorithm [11] and the knapsack scheduling algorithm [10].

C. Simulation Results

Fig. 4 shows the average layers on playback performance versus the total number of peers in the system (M). We can see that our proposed data scheduling algorithm outperforms the PALS and knapsack. The average layers on playback using our algorithm is 2 times more than that of PALS and knapsack when the network size is small. When the network size grows to 500, our algorithm still provides a 50 percent and 85 percent more layers than the knapsack and PALS, respectively. The poor performance of the knapsack and PALS is largely due to the deterministic piece request order, which tends to synchronize peer's demands and create content bottleneck. As a result, higher layer pieces can hardly be disseminated in the network. The results in Fig. 4 also confirm that the knapsack can provide more layers on playback than the PALS. This implies the utility based scheduling in knapsack is better than the sequential piece request in PALS.

The average jitter rate performance is shown in Fig. 5. Our algorithm yields a jitter rate around 4% and that of knapsack and PALS are less than 0.5%. The very low jitter rate achieved by PALS and knapsack benefit from their very low average number of layers on playback (< 1.5 in Fig. 4). Under such situations, the jitter event hardly occurs. The higher jitter rate under our algorithm is a side effect of the probabilistic piece request mechanism. As piece requests are made spreading out the buffer window, it increases the risk of leaving some vacancies in the buffer window. However, we would like to argue that jitter rate of our proposed algorithm is still at a low level. In considering the performance gain in the average layers on playback, this level of jitter rate is tolerable.

Fig. 6 and Fig. 7 illustrate the continuity performance. Similarly, both PALS and knapsack algorithm suffers badly in terms of both number of stops and total caching time. This is, again, largely attributed to the deterministic piece request in PALS and knapsack algorithm which leads to synchronized piece demand and creates content bottleneck. Consequently, the delivery of the base layer pieces are handicapped and the playback continuity is undermined.

V. CONCLUTIONS

In this paper, we proposed a data scheduling algorithm for layered P2P VoD streaming system. In the proposed algorithm, the data scheduling issues is tackled from two perspectives: 1) the layer adaptation phase, which employs a sliding buffer window to assess the network bandwidth and activate the layer probing process. The buffer window occupancy is used to make the final layer subscription decision so as to absorb extra network bandwidth while avoiding frequent quality switch; and 2) the piece selection phase, in which a utility value of each missing piece is calculated according to its playback constraint and layering dependency. The probability for a piece to be requested is proportional to its utility value. In comparison with existing algorithms, our packet-level simulation results illustrate the superiority of the proposed algorithm in enhancing the QoS in a layered VoD streaming netwroks

.

REFERENCES

[1] X. Hei, C. Liang, J. Liang, Y. Liu and K. W. Ross, "A measurement study of a large-scale p2p iptv system,." IEEE Transactions on Multimedia, 9(8):1672-1687, December 2007.

[2] Y. Huang, T. Z.J. Fu, D.M. Chiu, J.C.S. Lui and C. Huang, "Challenges, design and analysis of a large-scale p2p-vod system," in Proc. of the ACM SIGCOMM 2008 conference on Data communication.

[3] G.J. Conklin, G.S. Greenbaum, K.O. Lillevold, A.F. Lippman, and Y.A. Reznik, "Video Coding for Streaming Media Delivery on the Internet," IEEE Trans. on Circuits and Systems for Video Technology, Vol. 11, No. 3, pp. 269–281, 2001.

[4] T. Sun, M. Tamai, K. Yasumoto, N. Shibata, M. Ito and M. Moriy, "MTcast: Robust and Efficient P2P-based Video Delivery for Heterogeneous Users, " in Proc. Of OPODIS'05, pp.176-190,2005

[5] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the H.264/AVC standard," IEEE TCSVT, vol. 17, no. 9, September 2007.

[6] M. Wien, R. Cazoulat, A. Graffunder, A. Hutter, and P. Amon, "Realtime system for adaptive video streaming based on SVC," IEEE TCSVT, vol. 17, no. 9, September 2007.

[7] X. Xiao, Y. Shi, B. Zhang and Y. Gao, "OCals: a novel overlay construction approach for layered streaming," in Proc. of IEEE ICC'08

[8] Z.Liu; Y.Shen, K.W.Ross, S.S. Panwar and Y.Wang, "Layerp2p: using layered video chunks in p2p live streaming," IEEE Transactions on Multimedia, vol: 11, no: 7, PP: 1340-1352, 2009.

[9] X. Xiao ; Y. Shi ; Y. Gao and Q. Zhang "LayerP2P: a new data scheduling approach for layered streaming in heterogeneous networks," in Proc. of IEEE INFOCOM'09.

[10] M. Eberhard, T. Szkaliczki, H. Hellwagner, L. Szobonya and C. Timmerer, "Knapsack problem-based piece-picking algorithms for layered content in peer-to-peer networks," in Proc. of ACM AVSTP2P'10.

[11] R. Rejaie, A. Ortega, "PALS: peer-to-peer adaptive layered streaming", In Proc.of ACM NOSSDAV'03.

[12] J. Zhao, F. Yang, et. al, "On Improving the Throughput of Media Delivery Applications in Heterogenous Overlay Network", in Proc. of IEEE Globecom'06.

[13] A. T. Nguyen, B. Li, and F. Eliassen. Chameleon: adaptive peer-to-peer streaming with network coding," in Proc. of IEEE INFOCOM, 2010.

[14] D. M. Chiu, R. W.H. Yeung, J. Huang, and B. Fan, "Can network coding help in P2P networks?" in Proc. of Second Workshop of Network Coding, 2006.

[15] Z Wen, N. Liu, K.L.Yeung and Z Lei, "Closest playback-point first: a new peer selection algorithm for p2p vod systems", in Proc. of IEEE GLOBECOM'11.

[16] J. Huang, G.Cheng, J. Liu, and D.M.Chiu et.al, "A simulation tool for the design and provisioning of p2p assisted content distribution plateforms," under review, A copy is available at: http://personal.ie.cuhk.edu.hk/~dmchiu/references/P2P_Simulation.pdf.