Capacity aware Scalable Video Coding in P2P on Demand Streaming Systems

Changyou Xing, Ming Chen, Chao Hu
College of Command Information System
PLA University of Science and Technology
Nanjing, China
[e-mail: changyouxing@126.com]
*Corresponding author: Changyou Xing

Received October 17, 2012; revised December 9, 2012; accepted January 8, 2013; published September 30, 2013

Abstract

Scalable video coding can handle peer heterogeneity of P2P streaming applications, but there is still a lack of comprehensive studies on how to use it to improve video playback quality. In this paper we propose a capacity aware scalable video coding mechanism for P2P on demand streaming system. The proposed mechanism includes capacity based neighbor selection, adaptive data scheduling and streaming layer adjustment, and can enable each peer to select appropriate streaming layers and acquire streaming chunks with proper sequence, along with choosing specific peers to provide them. Simulation results show that the presented mechanism can decrease the system's startup and playback delay, and increase the video playback quality as well as playback continuity, and thus it provides a better quality of experience for users.

Keywords: P2P streaming; scalable video coding; data scheduling; neighbor selection

The work is supported by Natural Science Foundation of China under grant No 61103225 and No 61070173, and Key Project of Chinese National Programs for Fundamental Research and Development (973 program) under grant No 2012CB315806.

http://dx.doi.org/10.3837/tiis.2013.09.011

1. Introduction

In recent years, Peer-to-Peer based streaming applications have attracted large user communities, of which on demand streaming (video on demand, VoD) is a typical application. Like P2P live streaming systems, P2P VoD systems also deliver the video content by streaming. Videos are firstly divided into small data chunks, and each data chunk is delivered to peers by streaming server or other peers based on the data scheduling mechanism. But unlike P2P live streaming applications, peers in a P2P VoD system can watch different parts of a video at the same time, which results in a greater data diversity. To help peers sharing their playback contents, most P2P VoD systems requires each user to contribute a small amount of storage, and such a method increase the opportunities for different peers sharing contents.
Nowadays, most video coding rates are between 300kbps and 2Mbps, and as the increase of access network bandwidth, high definition videos are more and more popular. While at the same time, there are still low capacity devices such as smart phones, etc. When providing HD streaming service, the system should also take these devices into consideration. By coding a video into a basic layer and different enhanced layers, Scalable Video Coding (SVC) mechanism can dynamically adjust the video quality. A poor capacity peer can only acquire the basic video layer data for playback, while a high capacity peer can get the base layer as well as enhanced layer data in order to achieve a better video quality. Thus, SVC is very suitable for the heterogeneous P2P streaming system in terms of download bandwidth, terminal capabilities and user preferences. However, when adopting such a coding algorithm, the typical neighbor selection and data scheduling mechanism need also to be modified accordingly. For example, if a large number of poor capacity peers are selected as the immediate neighbors of streaming server, since none of them is interested in enhanced layer data, other peers will not be able to get such data effectively. Besides, how to adjust the streaming layers a peer can acquire and determine data downloading sequence so as to improve user's quality of experience also needs to be studied.

To solve these problems, we propose a capacity aware scalable video coding mechanism in P2P on demand streaming system named SVC-VoD, and the main goal of which is focusing on how to provide better video playback quality in heterogeneous P2P on demand streaming systems. To achieve such a goal, we cluster peers according to their capabilities, and streaming server distributes data chunks to high capability peer in priority, so that the data distribution efficiency can be improved. Besides, to leverage the video playback continuity, video playback quality as well as data chunk distribution efficiency, a priority data scheduling and layer adaption mechanism is also designed, so that peers can get appropriate data chunks quickly according to their capabilities. Generally, the contribution of this paper includes the followings.

Firstly, we propose a capacity aware peer cluster model, which makes peers with similar capacity as neighbors with each other, and streaming server provides service for high capacity peers in priority. Since these high capacity peers can play the role of capacity amplifiers, the video distribution process will be accelerated effectively.

Secondly, we present a priority data scheduling model, which designates the downloading probability of each data chunk according to its distance to the current playback position, the buffered data size as well as the layer sequence of such a data chunk. This model can make a compromise between playback continuity, video quality and data distribution efficiency.

Thirdly, we also propose a dynamic scalable video coding layer adaptation model. Peers can adjust the streaming layers they acquire according to their current buffer status and network performance, so as to guarantee the playback quality and video quality.

The rest of this paper is organized as follows. Section 2 presents a short survey of related work. Section 3 proposes the capacity aware scalable video coding mechanism in P2P streaming system, and presents the model of neighbor selection, data scheduling and streaming layer adaptation. Section 4 evaluates the performance of such a mechanism by simulation. And finally, section 5 concludes the paper, and gives a short discussion on future works.

2. Related Work

Using P2P method to distribute video can greatly reduce server bandwidth cost to provide video streaming service [1]. Recently, various P2P on-demand streaming systems have been deployed [2]. Peers have strong heterogeneity features in P2P streaming system, and thus there have been numerous efforts on the design and evaluation of layered video streaming systems in the last decade. Nahrstedt et al. used layered coding mechanism to solve the data request problem of peers with heterogeneous access bandwidths [3]. Hu et al. designed a taxation-based P2P layered streaming design including layer subscription strategy and mesh topology adaptation [4]. But they mainly considered how to strike the right balance between social welfare and that of individual peers, and thus the layer selection strategy mainly focused on fairness in P2P systems. Lee et al. gave a discussion on the challenges of data distribution and scheduling when constructing P2P live streaming system using SVC technique [5]. Borghol et al. presented a new adaptive window-based piece selection policy that balances piece diversity and in-order piece retrieval for P2P on demand streaming applications [6], but it mainly focused on peer incentive, and gave no analysis on the neighbor selection as well as layer coding problems. Abboud discussed the adaptive video layer adjustment mechanism in SVC based P2P video on demand applications, and they gave an analysis on the 3-dimension of H.264/SVC standard [7].

In [8], Nguyen et al. proposed a biased neighbor selection technique that can offer good performance in scalable video coding P2P streaming systems. Our work also uses the high capacity peers to help distributing rare data chunks, but we design a neighbor selection model to cluster peers as well as a streaming layer adaption model. Mokhtarian modeled the SVC based P2P video on demand system, and gave an analysis on the number of peers the system can accept during flash crowds [9], which presented a theoretical model for the SVC based P2P video on demand systems. Ding et al. proposed a scalable video coding based P2P video on demand model, and designed a Zig-Zag data scheduling model [10], but they did not take the video quality adaptation and neighbor selection problems into consideration. Nguyen et al. presented an adaptive coding quality adjustment mechanism [11], but they used network coding as their basis, and mainly focused on combining the benefits of network coding with layered streaming to mitigate the inherent challenges in unstructured P2P systems. Bradai et al. gave a discussion on playback smoothing mechanism for layered P2P streaming [12]. However, they mainly focused on layer adaption problem, and gave no analysis on the data chunk pre-fetching mechanism for the on demand streaming media system performance. The scheduling mechanism proposed in LayerP2P [13] is able to save base layer losses to the detriment of the enhancement layers. But its layer adaption was totally determined by network performance fluctuation, and thus had a relative poor quality of experience for users. Szkaliczki et al. [14] presented a number of theoretical solutions to maximize the utility

function of chunks. However, their solutions relied on the definition of chunk utility functions whose objective definition may be difficult in real-life scenarios. In [15] the authors presented a novel metric, named the Content Propagation Metric (CPM), to quantitatively evaluate the marginal benefit of available bandwidth, and CPM could guide a global allocation of bandwidth to maxmize the aggregate download bandwidth of peers. In [16], the authors employed random network coding into practical P2P streaming system, and such a mechanism improved the chunlk distribution efficiency. In [17], the authors analyzed the resource scheduling mechanism in CDN-P2P hybrid streaming system, and introduced differentiated service for peer's chunk requests according to their playback deadline.

Shiang et al. proposed a distributed resource management mechanism for delay sensitive transmission [18], which improved the multiple video streams performance effectively. Although the information exchange between network nodes of such a scheme is similar to our method, they mainly focus on resource management in multi-hop cognitive radio networks, while we pay attention to provide better quality of experience for users in P2P on demand streaming systems. Zhou et al. gave a thorough analysis on video streaming scheduling scheme over multi-channel multi-radio multi-hop wireless networks, and developed a fully distributed scheduling schemes so as to get balance between video distortion and fairness [19]. Our idea is similar to theirs with respect to self-adaptive media scheduling, but the two works try to optimize video streaming system performance at different network layers. Zhou's work mainly focuses on the channel resources assignment problem, and they give a cross-layer scheduling scheme, in which media server marks the importance of each packet at application layer, and the network links determine which packet to drop according to the under layer link status. On the other hand, we mainly focus on the application layer cooperation problems of peers in a P2P on demand streaming system. In our scheme, scheduling is the action of sending data requests to which peers or responding the data request of which peers, and the decisions of requesting which data chunks or dropping which data chunks are all made by end peers according to their buffered data size. Besides, we mainly focus on streaming data distribution in wired networks using a P2P mode.

3. Capacity Aware Scalable Video Coding Mechanism

In H.264/AVC (Advanced Video Coding) standard, a video streaming can be coded into a base layer and multiple enhanced layers. A peer can play the video after acquiring base layer data, and the enhanced layer data can help increasing video playback quality. The more number of enhanced layers a peer acquires, the better playback quality it will get. Besides, different enhanced layers also have dependency relationships, and upper layer data can only be decoded after acquiring low layer data.

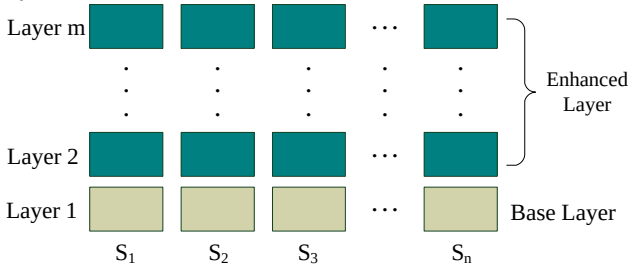


Fig. 1. scalable video coding

Fig. 1 gives a demonstration of SVC mechanism, in which the video is coded into *m* layers.

Without loss of generality, we suppose the video is divided into data chunks. In traditional single layer coding mechanism, each chunk represents a different playback time. However, in SVC mechanism with *m* layers, video data of one playback position is encoded into *m* different chunks, which includes one base layer and *m*-1 enhanced layers. A peer can acquire different number of chunks according to the capacity of itself and the network.

3.1 Neighbor Selection Model

In P2P streaming systems, the resources of peers should be used effectively so as to reduce streaming server's load. Generally, peers in P2P streaming system are heterogeneous. Some broadband access peers may have high upload bandwidth, while some low capacity peers such as wireless access ones only have limited upload bandwidth, which even cannot guarantee the basic playback rate of a video.

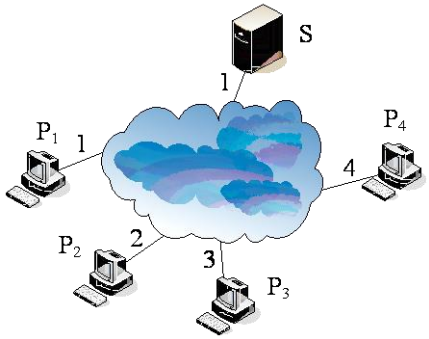


Fig. 2. Performance impact of data distribution sequence

Since peers in a P2P system have heterogeneous capacities, the data distribution sequence will affect the system performance dramatically. For example, in a simple overlay network topology shown in **Fig. 2**, which includes a server *S* and 4 peers P_1~P_4. Suppose the upload capacity of *S* is 1, and these of P_1~P_4 is 1~4 accordingly. When distributing a data chunk from *S* to all 4 peers, if *S* firstly sends the chunk to P_1, then a reasonable distributing sequence is $\{S\} \rightarrow \{S, P_1\} \rightarrow \{S, P_1, P_2, P_3\} \rightarrow \{S, P_1, P_2, P_3, P_4\}$, from which we can see that it needs 3 time slices to distribute such a chunk. Otherwise if *S* firstly sends the chunk to P_4, since P_4 can serve 4 peers in a time slice, then an optimized distributing sequence is $\{S\} \rightarrow \{S, P_4\} \rightarrow \{S, P_4, P_1, P_2, P_3\}$, which means it only needs 2 time slices to complete the data distribution. In a short word, sending data chunks to high capacity peers first will help increasing data distribution efficiency. By a further analysis we can see that, for any peers, if its upload capacity is *m* times of the video streaming rate *r*, then its helping effect for the data distribution will be (*m*-1)*r*. For P2P streaming system with strict playback time requirement, such improvement of data distribution efficiency will be more important for increasing the system performance.

Additionally, for applications using the layer streaming mechanism, the video stream is encoded into different streaming layers, and a peer can only provide data layers that it acquired from other peers. Thus high capacity peers should be neighbored with streaming server directly, so that they can acquire the complete data layers and distribute them into the overlay network. Otherwise if streaming server wastes its resources on distributing data to low capacity peers, since these peers only acquire partial layers, the upper layers of streaming data will not be distributed effectively by peers, and thus hinder the performance improvement of P2P streaming systems.

Based on the upper discussions, we propose a peer capacity based neighbor selection model. Peers are firstly clustered according to their capacity, and each peer selects peers with similar capacity with itself as neighbors. The streaming server provides service for high capacity peers in priority, so that a service chain is formed with a capacity descending order, and layered streaming data chunks can be distributed effectively in such a service chain.

The capacity comparison method is shown in equation (1). $\boldsymbol{C_i}$ is the candidate neighbor set of p_i acquired from tracker server or other peers. For each peer p_j in the candidate neighbor set $\boldsymbol{C_i}$, peer p_i will use the following method to determine whether selecting it as neighbor or not.

$$\frac{|Q_i - Q_j| - \min_{k \in C_i} |Q_i - Q_k|}{|Q_i - Q_j|} \le \tau \tag{1}$$

In which Q_i and Q_j is the upload capacity of p_i and p_j accordingly, and $0 \le \tau \le 1$ is an adaptive parameter. Besides, if there is more than predefined value k of peers satisfying the upper constraint, we will select k neighbors randomly from such candidate peers. The upper method achieves the goal of capacity based peer clustering, and its cluster effect is determined by the parameter τ. If $\tau = 0$, p_i will only select peer p_j that has the most similar capacity with it as neighbor. If $\tau = 1$, p_i will select k neighbors from candidate set completely in random.

Algorithm 1: Neighbor Selection

Input: Candidate peer list $\boldsymbol{C_i}$
Output: Neighbor list
(1) Get peer list $\boldsymbol{C_i}$ from neighbor peers;
(2) For each $peer_i$ in $\boldsymbol{C_i}$
(3) If $\frac{|Q_i - Q_j| - \min_{k \in C_i} |Q_i - Q_k|}{|Q_i - Q_j|} \le \tau$
(4) Add $peer_i$ into **Candidate**;
(5) *num* := sizeof (**Candidate**);
(6) If (*num* < *k*)
(7) $\tau = \tau + 0.1$;
(8) go to (2);
(9) Selecting *k* peers randomly from **Candidate**;

Algorithm 1 shows the capacity based neighbor selection mechanism for each peer, the basic function of which is selecting k peers as neighbors from the candidate peer set. If the number of candidate peers that satisfy upper selecting constraint is smaller than k, we will increase parameter τ so as to increase the number of feasible candidate peers. Extremely, if we define τ as 1, then all peers in the candidate set will satisfy such a constraint, and the algorithm becomes a random neighbor selection algorithm. Besides, for these peers selected as candidates, we will select k of them randomly as neighbors so as not to isolate the overlay topology. When receiving data requests, the streaming server only selects those peers with better capacities as neighbors and sends data chunks to them.

3.2 Data Scheduling Model

Rarest data first scheduling principle is quite successful in traditional P2P file sharing applications, but due to the playback continuity constraint of P2P streaming applications, such a data scheduling mechanism is not suitable here. Besides, the Interactive feature of P2P on demand media streaming system also makes different peers have large playback varieties. To

make a better use of peer resources, most of current methods use playback position to organize data scheduling, and each peer only request data from peers with playback positions a little ahead of it. However such a mechanism does not take peer capacity into consideration, and results in data chunks of later playback position very rare, while data chunks of earlier playback position can be provided by many peers, which wastes the valuable peer resources. Thus, in order to make the use of high capacity peers in helping data distribution, with the help of upper neighbor selection model, we propose a peer buffer status based data scheduling model, which uses the redundant resources of high capacity peers to pre-fetch the rare data chunks and distribute them into the overlay network, so that the overall data chunk acquirement efficiency will be increased.

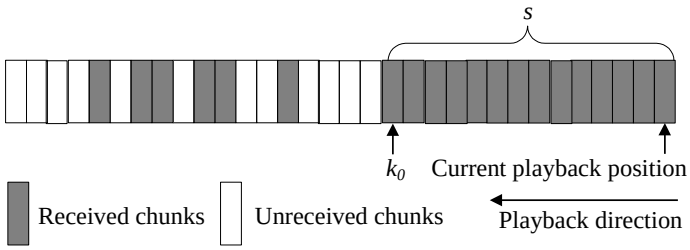


Fig. 3. buffer status aware data scheduling model

For a peer p_i, **Fig. 3** shows its typical buffer status. Suppose its current continuously buffered data size is s, and the last continuously acquired data chunk is with sequence number k_0, then we define the probability of p_i downloads some data chunk k ($k>k_0$) as equation (2).

$$P(k) = \frac{1}{(k-k_0)^{\frac{n}{s+1}}} \tag{2}$$

In equation (2) the parameter n is used for performance adaptation. $n=0$ means random data chunk scheduling, while $n \to \infty$ means strictly ordered data chunk scheduling. After a specific n is determined, a closer distance between data chunk k and the last continuous acquired data chunk k_0 means a larger probability to download it. Besides, no matter what other conditions are, the probability of acquiring the next continuous data chunk is the highest, so that the playback continuity can be guaranteed. On the other hand, larger value of s means peer p_i has more continuous data chunks buffered, and thus it can be more aggressive to download data chunks out of order. Thus, the resources of peers, and especially those high capacity peers, can be used efficiently to help improving the data chunk variety in the overlay network.

Fig. 4 shows the downloading probability of different data chunks with variable buffered data size and chunk number when $n=5$, in which s means the continuous buffered data size, x-axis means the difference between the current chunk sequence and the last continuously downloaded data chunk sequence, and y-axis means the probability of acquiring such a data chunk. From **Fig. 4** we can see that no matter what other conditions are, the probability of acquiring the next continuous data chunk is the highest, so that the playback continuity can be guaranteed. On the other hand, as the increase of distance between data chunk sequence and current last continuously acquired data chunk sequence, the probability of acquiring such a data chunk decreases accordingly. Besides, the larger a peer's buffer size of continuously acquired data chunk, the larger of the probability that it acquires the out of order data chunks. Thus, the resources of peers, and especially those high capability peers, can be used efficiently to help improving the data chunk variety in the overlay network.

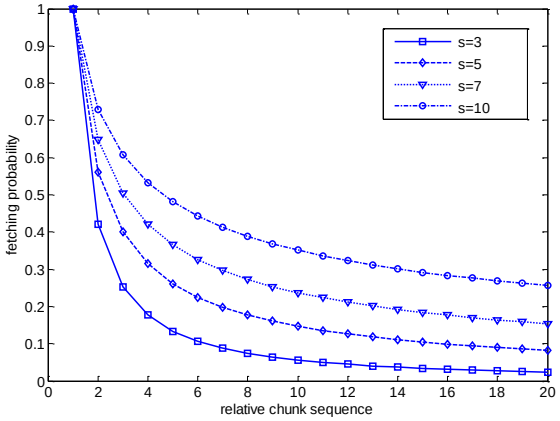
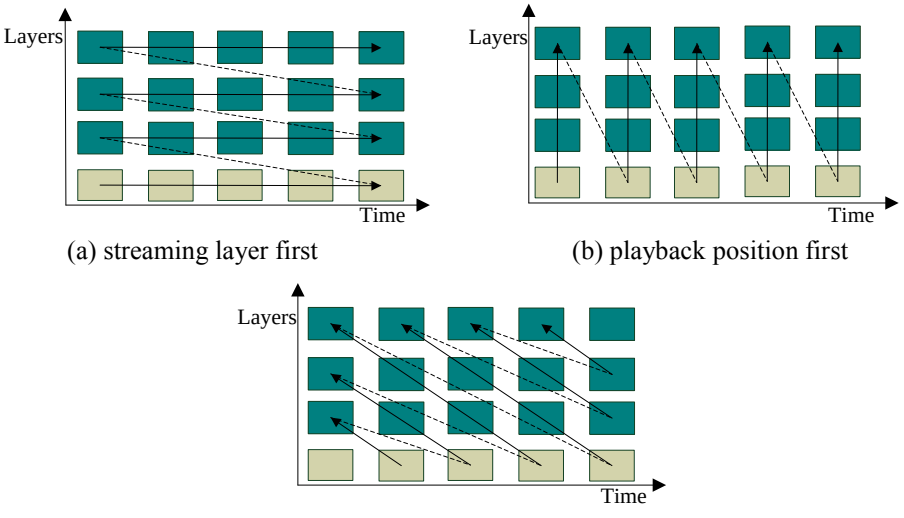


Fig. 4. data chunk download probability under different condition

However, after introducing the layered coding mechanism, besides the distance between the required and the last continuously acquired data chunks, the streaming layers of a chunk also impact its acquiring probability. According to the concept of scalable video coding, a higher layer data can be decoded only all of its lower layer data are acquired before. Thus, losses of the lower layers data will make the acquired upper layer data useless, and thus lead to the waste of system resources. As a result, the priority of different streaming layer data chunks should be different, and lower layer data should be given higher priority. Otherwise high layer data is of no use if we cannot acquire its corresponding low layer data on time. According to the discussion, we extend the upper equation (2) as follows.

$$P(k) = \frac{1}{(\alpha k + (1-\alpha)h - k_0)^{\frac{n}{s+1}}} \tag{3}$$



(a) streaming layer first

(b) playback position first

(c) hybrid mechanism

Fig. 5. data scheduling model

In equation (3) h means coding layers, which is a value between 1 and the maximum coding layer number, $\alpha \in (0,1)$ is a weighing factor, and it represents the bias between data chunk position and coding layers. $\alpha = 0$ means only the coding layers of data chunks impact

their acquiring probability, and a peer will acquire data chunks according to their coding layers, with no consideration of their distances to current playback point, which is represented as **Fig. 5**(a). While $\alpha = 1$ means only the distances between the data chunks and current playback point affect the probability, and a peer will acquire all coding layers of every data chunks sequentially, which is shown in **Fig. 5**(b). Otherwise the peer will acquire data chunks similar to **Fig. 5**(c) in a mixed style.

Fig. 6 shows the probability distribution with $\alpha = 0.2$, $n = 5$, s=10. From which we can see that a data chunk with a lower streaming layer sequence and near current playback position has higher acquiring probability. Thus such an algorithm can make a compromise between the playback continuity and the resources efficiency of high capacity peers.

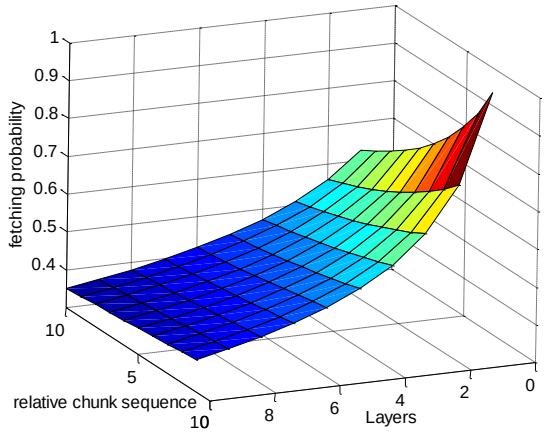


Fig. 6. data chunk acquirement probability

3.3 Dynamic Streaming Layers Adaptation Model

With scalable video coding, one important problem is how many layers each peer should acquire for playback. Obviously, every peer should acquire as many number of streaming layers as possible so as to improve the video playback quality. But due to the fluctuation in available bandwidth between peers, it will be impossible for peers to acquire all the streaming layers on time. To guarantee streaming playback continuity, the acquired streaming layers should also be adjusted dynamically in accordance with the peer capacity and network constraint. Thus we design a dynamic streaming layer adaptation model, which determines the maximum streaming layers a peer can acquire without affecting its playback continuity, and then make streaming layer adaption according to such constraint.

Fig. 7 shows the basic idea of dynamic streaming layer adaptation model. Its main operation is inspecting the status of current buffered data size, so as to determine whether the video playback continuity can be guaranteed. If the buffered data size is much larger than current playback progress, the peer can get more layers of data chunks to improve the video playback quality. Otherwise if the buffered data size too small to guarantee the playback continuity, the peer will enter an urgent state and decrease the acquired streaming layers so as to guarantee the playback continuity.

Intuitively, a peer should increase the number of coding layers it acquires whenever possible, so as to increase streaming playback quality. However, it is generally observed that it is visually more pleasing to watch a video with consistent, lower quality than one with higher but varying quality. That is to say, a better layer adaption model must have the ability of smoothing streaming quality fluctuation, so as to minimize the negative effect on the

perceived streaming quality by adding and dropping layers, and switching gracefully from one quality level to another.

Taking the upper constraints into consideration, the objective of the layer adaptation model is to optimize the perceived video quality, while at the same time ensuring the smooth delivery of the layered streaming. Thus we define two thresholds T_1 and T_2 to determine when to perform layer adaptation, and set $T_1 = T_2 + \delta$ so as to avoid the layer fluctuation of peers. Here $\delta > 0$ is a stability factor, and a larger δ means a smaller possibility of layer fluctuation.

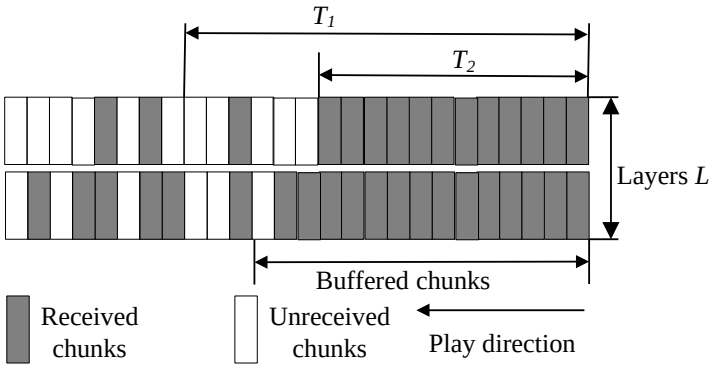


Fig. 7. streaming layer dynamic adaptation model

In such a model, each peer has the following states: Empty state, General state, and Urgent state, and different states imply different peer's operation for better playback quality. **Fig. 8** gives the finite-state-machine (FSM) definitions for the layer adaption model, in which the arrows indicate the transition of the model from one state the another, and the event causing such transition is shown above the horizontal line labeling the transition, while the actions taken when the event occurs are shown below the horizontal line. The symbol Λ means no action is taken on an event.

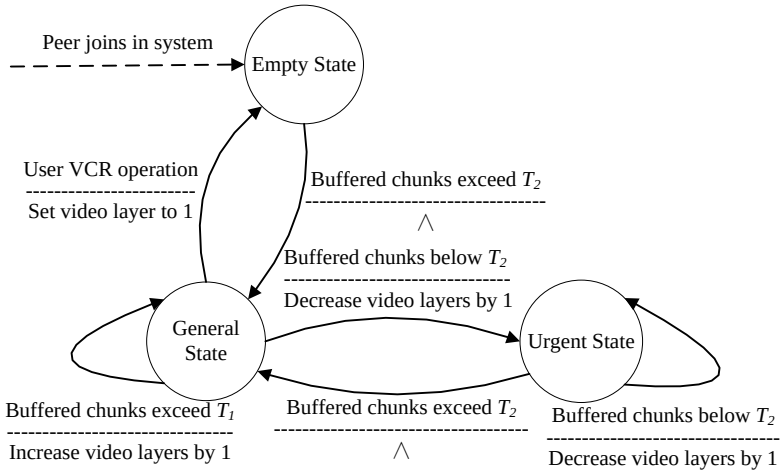


Fig. 8. state transmission graph of peers

Empty state means the state of a peer just joins in the system, or just after a VCR operation. In this state there is no data for playback in the peer's buffer, thus the peer should select neighbors and then acquire data from them and streaming server as soon as possible. In order

to decrease the user waiting time, the peer will only acquire the basic layer data. If the buffered data size exceeds threshold T_2, the peer will migrate into General state.

General state means a peer can acquire enough data chunks to guarantee playback contibuity. If the peer's buffered data size is larger than threshold T_2 but smaller than threshold T_1, which means the peer has adequate video chunks for smoothing playback, but the buffered data is also not adequate enough for adding additional layers (since buffered data size is smaller than T_1), the peer will keep the acquired streaming layers unchanged. Otherwise, if the buffered data size exceeds threshold T_1, which means the peer has quite a large number of continuous buffered data for playback, then the peer will increase its acquired streaming layer by 1, and download the video data according to the new streaming layer. If the buffered data size is smaller than threshold T_2, the peer will migrate into Urgent state.

Urgent state means a peer's buffered data chunks cannot support playback continuity, and thus it should adjust the number of streaming layers it acquires. In this state, if the number of currently acquired streaming layers is larger than 1, which means the peer also acquires one or more enhanced layer data besides the base layer data, then the peer will decrease it by 1, and download the streaming data according to the new streaming layer so as to guarantee playback continuity. When the buffered data size is larger than T_2, The peer will migrate into General state again.

Algorithm 2: Video Quality Adaptation

```
(1) if( buffered chunks size > T1)
(2)     if (LS < max_level);
(3)         LS := LS + 1;
(4) if(buffered chunks size < T2)
(5)     if (LS > 1)
(6)         LS := LS - 1;
(7) if(user firstly join in system or VCR operation)
(8)     LS := 1;
```

Algorithm 2 shows the basic operation of such a state transmission. From algorithm 2 we can see that, the streaming layer is designated to be 1 after the VCR operation, and is this adjustment too aggressive? In our opinion, since there is no buffered data after a VCR operation, thus the most important thing for a peer is acquiring enough data for playback as soon as possible, otherwise user will have to wait a long time and get poor playback experience. Besides, since the streaming is not played sequentially, a decrease of streaming layer will not result in a flagrant quality contrast. If the network performance is good, the acquired streaming layer will soon be increased quickly to enhance the playback quality.

4. Performance Evaluation

4.1 Experiment Scenario

To evaluate the performance of SVC-VoD, we design an event driven P2P streaming simulator based on [20]. The original simulator is designed for P2P live streaming simulation, and can simulate the packet-level data transmission and end-to-end delay among the peers. We modify this simulator, and enable the new one can support P2P on demand streaming with the features of scalable video coding, streaming layer adaptation and capacity based neighbor selection presented here. In our experiments, peers are divided into 3 kinds, and their network

access rates as well as distribution ratio are shown in **Table 1**. There are 1 video server and 2000 peers in total, each peer selects 15 peers as neighbors using random or our capability based neighbor selection algorithm, and neighbors exchange buffer maps to notify data chunks each has. Each peer uses pull mode to acquire data chunks, and the arrival as well as departure model follows a random distribution. For each peer, the upload bandwidth is the bottleneck. In our simulation, we define the video length to be 30 minutes long, and the video coding rate is 300kbps. The adaption factor T_1 and T_2 are defined to be 20 and 15 respectively, and the initialized value of τ is 0.2. The data chunk request probability parameters are set to be $\alpha = 0.2, n = 5$. By using the scalable video coding mechanism, the video is encoded into 10 streaming layers, and each streaming layer has a video rate of 30kbps. Thus according to the difference of streaming layer a peer acquired, the video rate it acquired has a variance between 30 and 300kbps. Besides, each peer can also buffer all of the video data it has played. In order to carry out the performance comparison, we also implement a single layer P2P on demand media streaming system, and all its parameters except neighbor selection, data scheduling and layered coding are similar to the SVC-VoD model.

Table 1. Classification of Peers

Upload Rate	Download Rate	Peers ratio
768 kbps	3 Mbps	0.15
384 kbps	1.5 Mbps	0.3
128 kbps	768 kbps	0.55

To evaluate the system performance and user experience quality, we define the following performance evaluation metrics.

Startup delay, which describes the time from a peer joins in the system to it gets enough data chunks for playback. A smaller startup delay means a better user playback experience.

Playback delay, which specifies the average waiting time due to data chunks do not arrive on time for playback. A smaller playback delay means better video play continuity, and a playback delay of 0 means there is no discontinuity during video playing.

Video playback quality, which describes the quality of video playback. Besides, with scalable video coding, the number of coding layers a peer acquired is adaptive to the network and peer capacity, and video playback quality specifies the average video quality a peer acquired. Ideally this metric should be the same as the original video coding rate.

Playback continuity. To guarantee the video playback continuity, the data chunk must arrive before its playback time point; otherwise it will be of no use. With scalable video coding mechanism, there are two phenomenon should be considered. For one thing, if the lower layer data does not arrive on time, the arrival of upper layer data is still of no use. For the other thing, if the lower layer data arrives on time, the loss off upper layer data does not affect the playback continuity. Referred to the concept of [21], we give the definition of playback continuity in layered streaming systems as follows.

$$CI = \frac{\#requested - \#missing - \#out\_of\_order}{\#requested} \tag{4}$$

In equation (4) *#requested* represents the number of data chunks that the peer requests, *#missing* represents the number of data chunks that are lost during network transmission, and *#out_of_order* represents the number of data chunks that are received without their lower layer data arriving on time, and thus making them useless.

4.2 Performance Evaluation

To evaluate the performance of SVC-VoD, we compare it with traditional single layer random neighbor selection model. In the traditional model, the video only has 1 streaming layer, and each peer uses random neighbor selection as well as in order data acquiring method. Besides, the streaming server does not care about the peers' capacity, and uses first come first service model to distribute data chunks.

During our simulation, the buffered time is defined as 10*s*, which means when firstly joining the system, a peer can only play the video after acquiring 10*s* video resources. **Fig. 9** gives cumulative distribution of the startup delay, from which we can see that SVC-VoD has a much smaller startup delay than the single layer system. That is because for one thing, by using dynamic coding layer adaptation method, a peer can start video playback after acquiring the base layer data. And for the other thing, the streaming server serves high capacity peers firstly, and thus increases the video distribution capacity of the overlay network. As a result, the startup delay can be decreased effectively.

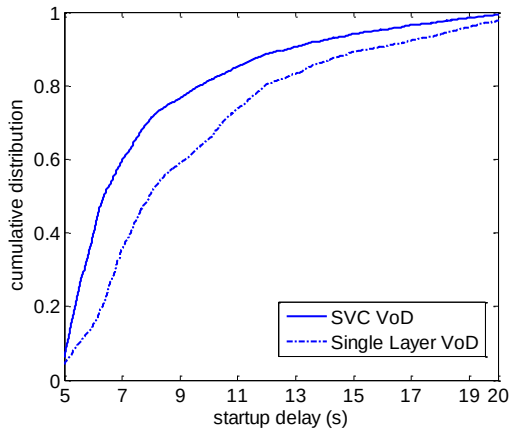


Fig. 9. cumulative distribution of startup delay

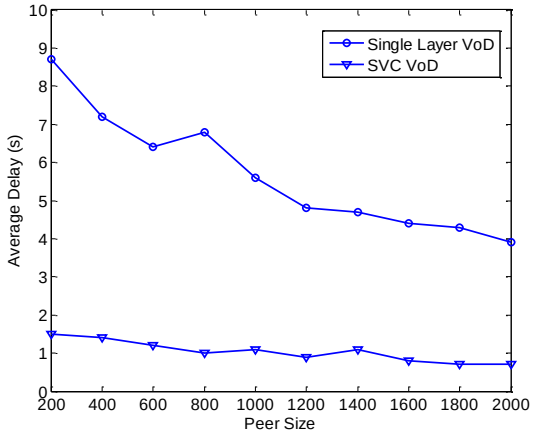


Fig. 10. average playback delay comparison

Fig. 10 shows the average video playback delay comparison of the two models, from which we also see that as the increase of peer size, the average playback delay of the two models all decrease accordingly. The main reason is that in the P2P on demand streaming system, each peer buffers the video resources it has played. When the number of peers increases, if their arrival ratio keeps on the same, the number of available video resources will keep on increasing for latter arrival peers, and thus the playback delay will be decreased accordingly. However, the performance of SVC-VoD is better than that of single layer coding model in regardless of the peer scale. That is because by capacity based peer cluster and priority service model, SVC-VoD can increase the data sharing ratio. Besides, when the network performance becomes poor, SVC-VoD can only acquire the basic layer data and keep on the playback process continually, but the single layer coding model cannot perform such an adaptation and thus result in large playback delay.

Fig. 11 gives the video playback quality comparison of the two models, which also shows the similar trend as upper metrics, and SVC-VoD demonstrates a better performance than that of single layer coding model. The key reason of such a result is similar as that of playback delay. SVC-VoD increases the data sharing ratio by capacity based peer cluster and priority service model, and it can also adjust the acquired data layers dynamically according to network status, thus its video playback quality will be better than that of single layer mechanism.

Fig. 12 describes the playback continuity of the two models, from which we can see that the SVC-VoD is much better than single layer coding model. One of the main reasons is that SVC-VoD can keep on the playback continually by only acquiring the basic layer data chunks, while the single layer coding model must acquire the entire video chunks before playback. Besides, the capacity based peer cluster and data scheduling mechanism also increases the overall data distribution efficiency, and thus increases the video playback continuity accordingly.

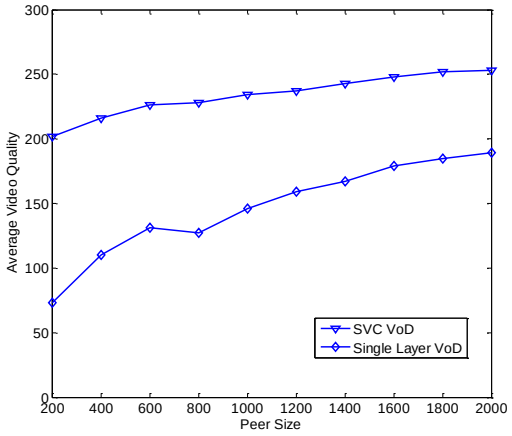


Fig. 11. average streaming quality comparison

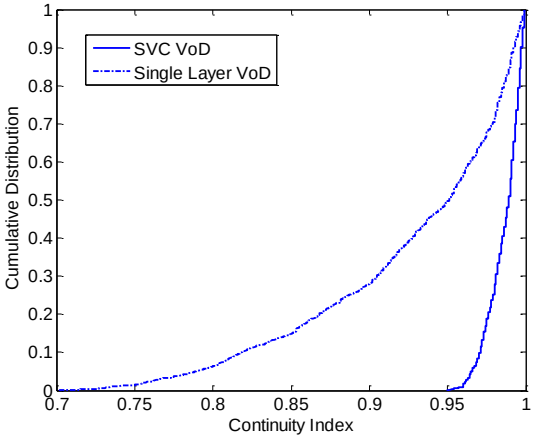


Fig. 12. cumulative distribution of playback continuity

5. Conclusion

P2P on demand streaming system is one of the most popular Internet applications, but its user quality of experience is still unsatisfied. To provide better video playback quality with heterogeneous peers, we introduce the layered coding mechanism, and propose a capacity aware scalable video coding mechanism named SVC-VoD in P2P on demand streaming system, which includes 3 main steps. Firstly each peer selects neighbors according to their capacities; secondly data chunks are scheduled based on their distance to current playback position, their coding layers as well as the sequentially buffered data size of a peer, and finally the acquired streaming layers are dynamically adapted according to network and peer capacity. To study the effectiveness of the proposed mechanism, we perform the simulation and compare the proposed mechanism with existing system. Simulation results show that SVC-VoD can decrease the system's startup and playback delay, and increase the video playback quality as well as playback continuity, and thus it provides a better quality of experience for users. When network or peer capacity changes, SVC-VoD can also adapt the streaming layers dynamically so as to guarantee the video playback continuity and video quality, which demonstrates the optimality and the effectiveness of the solution. In the future work, we will optimize such mechanism further and design a prototype on PlanetLab. Besides, studying the performance of such a mechanism on distributing short videos such as UGC (User Generated Content) is another focus of our future work.

Acknowledgment

The work is supported by Natural Science Foundation of China under grant No 61103225 and No 61070173, and Key Project of Chinese National Programs for Fundamental Research and Development (973 program) under grant No 2012CB315806.

References

[1] C. Huang, J. Li, K.W. Ross, "Can Internet Video-on-Demand Be Profitable?," in *Proc. of ACM SIGCOMM'07*, Kyoto, Japan, Aug. 2007. Article (CrossRef Link)
[2] Y. Huang, T. Fu, D. Chiu, J. Lui, and C. Huang, "Challenges, Design and Analysis of a Large-Scale P2P-VoD System," in *Proc. of ACM SIGCOMM'08*, Seattle, Washington, USA, Aug. 2008. Article (CrossRef Link)
[3] Y. Cui, K. Nahrstedt, "Layered peer-to-peer streaming," in *Proc. of NOSSDAV'03*, Monterey, California, USA, Jun. 2003. Article (CrossRef Link)
[4] H. Hu, Y. Guo, Y. Liu, "Mesh-based peer-to-peer layered video streaming with taxation," in *Proc. of NOSSDAV'10*, Amsterdam, The Netherlands, Jun. 2010. Article (CrossRef Link)
[5] T. Lee, P. Liu, W. Shyu, C. Wu, "Live Video Streaming Using P2P and SVC," in *Proc. of MMNS'08*, Samos Island, Greece, Sep. 2008. Article (CrossRef Link)
[6] Y. Borghol, S. Ardon, N. Carlsson, and A. Mahanti, "Toward Efficient On-Demand Streaming with BitTorrent," in *Proc. of IFIP Networking'10*, Chennai, India, May, 2010. Article (CrossRef Link)
[7] O. Abboud, T. Zinner, K. Pussep, S. Al-Sabea, and R. Steinmetz, "On the Impact of Quality Adaptation in SVC-based P2P Video-on-Demand Systems," in *Proc. of ACM Multimedia Systems*, San Jose, California, USA, Feb. 2011. Article (CrossRef Link)
[8] A. Nguyen, B. Li, F. Eliassen, "Quality- and context-aware neighbor selection for layered peer-to-peer streaming," in *Proc. of IEEE ICC 2010*, Cape Town, South Africa, May, 2010. Article (CrossRef Link)
[9] K. Mokhtarian, M. Hefeeda, "Analysis of Peer-assisted Video-on-Demand Systems with Scalable Video Streams," in *Proc. of ACM MMSys'10*, Phoenix, Arizona, USA, Feb. 2010. Article (CrossRef Link)
[10] Y. Ding, J. Liu, D. Wang, H. Jiang, "Peer-to-peer video-on-demand with scalable video coding," *Computer Communications*, vol. 33, no. 14, pp. 1589–1597, Sep. 2010. Article (CrossRef Link)
[11] A. T. Nguyen, B. Li, and F. Eliassen, "Chameleon: Adaptive Peer-to-Peer Streaming with Network Coding," in *Proc. of IEEE INFOCOM'10*, San Diego, CA, USA, Mar. 2010. Article (CrossRef Link)
[12] A. Bradai, U. Abbasi, R. Landa, T. Ahmed, "An efficient playout smoothing mechanism for layered streaming in P2P networks," *Peer-to-Peer Network and Applications*, Sep. 2012. Article (CrossRef Link)
[13] X. Xiao, Y. Shi, Y. Gao, "On optimal scheduling for layered video streaming in heterogeneous peer-to-peer networks," in *Proc. of ACM international conference on Multimedia*, Vancouver, British Columbia, Canada, Oct. 2008. Article (CrossRef Link)
[14] T. Szkaliczki, M. Eberhard, H. Hellwagner, L. Szobonya, "Piece selection algorithm for layered video streaming in P2P networks," *Electron Notes Discrete Math*, vol. 36, no. 1, pp. 1265–1272, Aug. 2010. Article (CrossRef Link)
[15] S. Peterson, B. Wong, E. Sirer, "A Bandwidth Propagation Metric for Efficient Content Distribution," in *Proc. of ACM SIGCOMM'11*, Toronto, Canada, Aug. 2011. Article (CrossRef Link)
[16] Z. Liu, C. Wu, B. Li, S. Zhao, "UUSee: Large-Scale Operational On-Demand Streaming with Random Network Coding," in *Proc. of IEEE INFOCOM'10*, San Diego, CA, USA, Mar. 2010. Article (CrossRef Link)
[17] C. Hu, M. Chen, C. Xing, B. Xu, "EUE principle of resource scheduling for live streaming systems underlying CDN-P2P hybrid architecture," *Peer-to-Peer Network and Applications*, vol. 5, no. 4, pp 312-322, Dec. 2012. Article (CrossRef Link)

[18] H. Shiang, M. Schaar, "Distributed Resource Management in Multi-hop Cognitive Radio Networks for Delay Sensitive Transmission," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 2, pp. 941-953, Feb. 2009. Article (CrossRef Link)
[19] L. Zhou, X. Wang, W. Tu, G. Muntean, and B. Geller, "Distributed Scheduling Scheme for Video Streaming over Multi-Channel Multi-Radio Multi-Hop Wireless Networks," *IEEE Journal on Selected Areas in Communications*, vol. 28, no. 3, pp. 409-419, Apr. 2010. Article (CrossRef Link)
[20] M. Zhang, Y. Xiong, Q. Zhang, L. Sun, S. Yang, "Optimizing the Throughput of Data-Driven Peer-to-Peer Streaming," *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, no. 1, pp. 97-110, Jan. 2009. Article (CrossRef Link)
[21] Y. Yang, A. Chow, L. Golubchik, D. Bragg, "Improving QoS in BitTorrent-like VoD Systems," in *Proc. of IEEE INFOCOM'10*, San Diego, CA, USA, Mar. 2010. Article (CrossRef Link)

Changyou Xing received the Ph.D. degree from PLA University of Science and Technology, Nanjing, China, in 2009. He is currently a lecture in the college of command information systems at PLA University of Science and Technology, Nanjing, China. His research interests include peer-to-peer multimedia communications, future network, network modeling.

Ming Chen received the Ph.D. degree from Nanjing Institute of Communication Engineering, China, in 1991. He is currently a professor in the college of command information systems at PLA University of Science and Technology, Nanjing, China. He held visiting position at Columbia University in 1999. His research interests include network architecture, future network, network measurement and performance evaluation, distributed computing.

Chao Hu received the M.S. degree from PLA University of Science and Technology, Nanjing, China, in 2008. He is currently a Ph.D. student at PLA University of Science and Technology. His research interests include peer-to-peer streaming, distributed computing.