Peer-to-Peer Video-on-Demand Streaming Using Multi-Source Forwarding Scheme

Yu-Chih Teng, Peng-Jung Wu, Chung-Nan Lee
Department of Computer Science, National Sun Yat-sen University, Kaohsiung, Taiwan
M953040012@student.nsysu.edu.tw, wupl@cse.nsysu.edu.tw, cnlee@cse.nsysu.edu.tw

Abstract*-In a video-on-demand streaming system, each user watches different video frame according to his arrival time. In the paper, the concept of multi-source and the forward error correction scheme are integrated to decrease the workload bandwidth of server and reduce the packet loss probability of each peer in the peer-to-peer video-on-demand system. Here, in order to share data to some peers arriving later, each peer must cache part of video data that recently viewed. Each new arrival peer needs to contact multiple sources who have initial part of video data to get video streaming data, and each source transmits different part of video streaming. Once receiving all of these partial streams, the peer will get completely video data. Simulation shows that a suitable preserve time of peers in the system can be used. Thus, the workload bandwidth of server and memory used by peers can be reduced. Furthermore, the packet loss probability is decreased by the sources diversity and the FEC recovery.*

Keywords: video-on-demand, peer-to-peer, FEC, multiple source, packet loss.

1. Introduction

Peer-to-Peer (P2P) technology has been successfully used on many applications, such as file sharing [1] and video multicasting [2]. In P2P systems, peers act as senders and receivers at the same time. By employing the P2P concept, the workload of server can be reduced. Since the scalability of P2P systems, many P2P streaming systems have been proposed [3-8] that can be divided into two categories: live streaming systems [3-5] and video-on-demand (VoD) streaming systems [6-8].

For live streaming systems, all peers are synchronously watching the same video. Thus any peer can be a parent to share the current viewed video contents to any other peer in the same system. In VoD systems, however, peers join the system to watch video from the beginning. Namely, peers in VoD streaming systems may be watching different parts of the same video. Therefore, only a subset of peers that are watching roughly the same video frames can be selected as parent. Since the video is pre-recorded and each peer will watch different video frames according to its arriving time. In this paper, we are interested in providing VoD service by employing P2P technology, the following discussion focuses on VoD P2P systems.

Recently, numerous P2P systems have been proposed, which can be further classified into two categories: Tree-base and Mesh-based systems. In tree-based P2P systems, including P2Cast [6], and P2VoD [7, 8], are proposed. In mesh-based P2P systems such as PROMISE [3] and SplitStream [4], no specific topology is created. Peers, based on the design rules, connect to multiple parents to obtain video packets.

Guo et al. [6] proposed an early patching scheme for VoD service, called P2Cast. In P2Cast, peers arriving roughly in the same time will be grouped into a session, and each session form a multicast tree to delivery video stream on it. P2Cast uses a tree-based structure to delivery data, which is easy to construct but difficult to maintain. Moreover, the recovery scheme in P2Cast is to let peers receive data from server directly when parent departure occurs. However, it increases the workload of the server.

P2VoD [7, 8] proposed by Do et al. is similar to P2Cast. In P2VoD, it defines the group which consists of fixed number of peers that arrive closely. Peers in the same group share video data according to the sequence in the group between two nearly group. But P2VoD does not consider the heterogeneous bandwidth of peers.

Hefeeda et al. [3] proposed PROMISE, a P2P media streaming system is based on a novel P2P service called *CollectCast*. In PROMISE, a peer connects to multiple senders, dynamically

switches active senders and standby senders, monitors neighbor peers and connections, and recovers the peer's connection failure or degradation. It does not focus on VoD system.

Castro et al. [4] proposed the SplitStream to deal with the problem of the unbalance forwarding load of interior nodes in tree-based P2P system. It stripes the data into *k* stripes and forwards each stripes using separate trees. Thus, the interior node in one tree may not be interior node in other tree. But SplitStrem does not focus on VoD either.

However, these P2P systems do not discuss the video quality impact due to the packet loss. Wu et al. [5] used multiple sources and *FEC* recovery method [9] to overcome the burst packet loss due to peer departures. In this method, each peer gets *N* video data packets including *k* data packets and *N-k FEC* packets from different parents. When one or few of these parents leaves, the missing packets can still be recovered by *FEC*. The method also eliminates packet loss propagation.

In this paper, we propose a video-on-demand streaming system using multi-source concept in the P2P environment. The main contributions of this work are as follows. 1) By dividing the video streaming data into several different sub-streams, the upload bandwidth of peers will be used more efficiency, thus the workload of the server can be reduced. 2) Since employing multi-source concept and the *FEC* recovery method, the packet loss propagation is eliminated. 3) The minimum required bandwidth of server can be determined by given a buffer threshold.

The rest of this paper is organized as follows. Section 3 presents the proposed multi-source VoD system. Simulation and performance results are presented in Section 4. Finally, conclusion and summary are drawn in Section 5.

2. Design of a Multi-source VoD System

In this section, the proposed multi-source VoD system is explained in details. In this paper, peers have different upload bandwidth capacity. We assume that each peer must determine how many upload bandwidths it would like to provide when it joins into the system. The server maintains information about each peer in the system in order to assign the candidate peer for other peers.

2.1. Data Caching Concept

In this paper, the video streaming is considered as a constant bit-rate video, which is equal to the playback bit-rate. The video is packetized into a sequence of packets. The video server then generates *N-k* redundant packets from every *k* original packets using *FEC(N, k)* as shown in Fig. 1. These *N* packets including *k* data packets and *N-k* redundant packets altogether are called a *block* and are transmitted to peers who connect to video server directly. On receiving the packets, peers forward the received packet to their child peers and so on. Every time some packets are dropped during transmitting to a peer, the dropped packet can be recovered by using *FEC* if the total number of dropped packets is smaller than or equal to *N-k*.

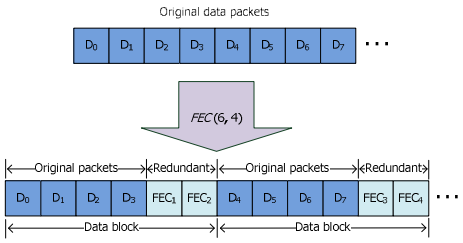

Figure 1. Original data packets transform to data block by *FEC*

Since peers join the system to watch video from the beginning of the video, video server will have to provide the initial part of the video for every new arriving peer if peers always discard the viewed packets. It increases the workload of the video server. To overcome this problem, peers in the proposed system are requested to preserve a certain number of recently viewed video frames. Let *T* be the length of preserved viewed video frames of each peer, *T* is predefined by the system and the unit of *T* is equal to the time unit of video stream. If a new peer arrives at time *t*, it can obtain the video, including the initial part, from other peers that arrived *T* time unit before *t*. That means the peers arrived during *t-T* to *t* are potential parents of the new arriving peer and the set of these peers is defined as a candidate list of the new arriving peer.

A new arriving peer can choose any peer as parent itself from candidate list. If there is no peer in the candidate list, then the peer connects to server and gets video stream from server directly.

2.2. Multi-Source Video Data Forwarding Concept

To avoid burst packet loss, the multi-source structure is employed in this system. In the multi-source structure, each peer connects to

multiple parents and each parent only forwards the *i*-th packet of each block to the child peers who subscribe to packet forwarding sub-stream *i* as shown in Fig. 2. Packet forwarding sub-stream *i* is a sequence of packets which consists the *i*-th packet of each data block. Since each parent forwards different part of the streaming, multi-source structure does not introduce additional packets transmission overhead.

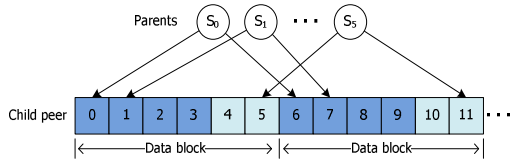

Figure 2. Data forwarding in multi-source structure.

Every time a peer joins the system, it connects to *N* parents and subscribes for all *N* sub-streams to obtain all the streaming packets. Since each parent is only responsible for forwarding one packet sub-stream to the peer, once a parent departs from the system, the missing packets can still be recovered from the remaining part of packets by using *FEC*. By considering the bandwidth asymmetry of parent, a child peer can also subscribe multiple packet forwarding sub-streams from one parent peer if the parent has sufficient bandwidth. In other words, the child peer can subscribe more than two sub-streams from the same parent peer.

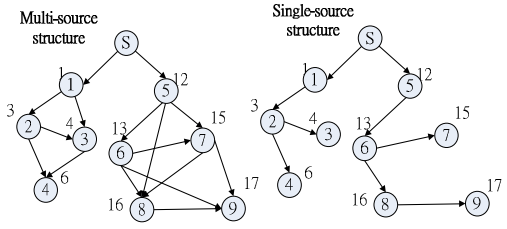

Figure 3. Comparisons of a multi-source and a single source structures.

Figure 3 is an example of multi-source structure compared to a single source structure. In this figure, S represents the video server, and each peer id is written inside the circle. The arrival time of each peer is written around the circle. In this example, the buffer threshold is five seconds. Thus, peers can forward the data to the latter arrival peers whose arrival time exceed in the threshold time. Otherwise, the server will open a new session for the latter peer, such as peer 5. In this example, we suppose each peer needs to subscribe three sub-streams from other peers. When the peer 1 arrives, it must subscribe all sub-streams from server directly. Then peer 2 subscribes all three sub-streams from peer1 since there is only peer 1 in the system. After that, peer 1 and peer 2 collaboratively forward data to peer 3 with different sub-streams, i.e., peer 1 provides one sub-stream and peer 2 provides two sub-streams. As time goes by, there are more and more peers in the system, the new peer can get more and more candidates. As a result, a new arriving peer can choose three peers as parents easily, such as peer 8 and peer 9.

There are two advantages of a multi-source structure in comparison with a single source structure. First, the workload bandwidth of server is reduced. For example, assume the transmission rate of the video streaming is 200 Kbits/sec., and there are one parent peer with upload bandwidth 50Kbits/sec. and one child peer. In the single-source structure, the upload bandwidth of parent peer is useless, since it is not capable of providing the minimum contribution requirement, i.e. 200 Kbits/sec.. So, the server must provide total 200Kbits/sec. to the new coming peer. In a multi-source structure, however, even the upload bandwidth of parent peer is 50 Kbits/sec., so it can also contribute all its upload bandwidth to child peer.

The second advantage of multi-source system is to reduce the propagation of the packets loss. In the single source P2P system, once packet losses occur at a parent peer, all its child peer will have no way to receive the lost packets from the parent. Even those child peers pull the lost packets from other parents; it introduces additional delay which might make the packets useless. However, in a multi-source structure, since each parent only forwards partial packets to child peers, child peers can still receive most packets from other parent when packet loss occurs at parents. The received packet can then be used to reconstruct the original data packets by *FEC*.

2.3. Peer Join Process

Due to the heterogeneous download bandwidth of each peer, some peers, with lower download bandwidth than video transmission capacity, might not receive video data smoothly. Such peer cannot watch the video successfully. Thus, here we discuss all peers in the system have enough download bandwidth to get the video data.

When a new peer wants to join the system, it contacts the server first and determines how many upload bandwidth it can provide. Then, the

server replies a candidate list to the new peer. Once the new peer gets the candidate list from server, the new peer chooses N different sub-stream sources from the candidate list. If there are not enough N sources and the server has no bandwidth to support the new peer, the new peer would be rejected.

After the server replies a candidate list to new peer p, peer p chooses parents from the candidate list. In our system, the parent choosing method is based on *Round Robin Selection.* That is the requesting peer selects the candidate peer who has not served other peer for the longest time. Generally, since the new coming peer has no more child peer to support, the arrival time of a peer which is closer to the arrival time of the new peer can have more enough bandwidth to support. Since the candidate list is ranked as the arrival time from younger to older, the new peer will choose the parents from the first peer in the candidate list. The parent peer choosing process stops until the new peer has N sub-stream sources or there is no peer in the candidate list, where N is the parameter of *FEC*.

2.4. Failure Recovery Process

Once a child peer detects a sub-stream fails or the quality of the sub-stream falls to a threshold, the child peer starts a failure recovery process to find a new parent. The child peer selects a new parent based on the candidate list it got when it joined in. This recovery procedure is similar to the join algorithm.

In the failure recovery procedure, suppose p is a peer who needs to find a new parent. Peer p avoids choosing the peer who has been a parent of p to maintain source diversity. But if there are not enough different peers can be the sub-stream sources of p, p can still choose some peers who are already a sub-stream source of p. Finally, if p cannot find the recovery parent, the server will try to support the peer. If the server does not have enough to support, p will be rejected.

Since the multi-source structure is employed, a peer receives packets from multiple parents and each parent is only responsible for forwarding partial packets, i.e., one forward sub-stream. The benefit of obtaining packets from multiple sources is that when one or few parents leave, other parents can still provide most remaining part of video streaming packets. If the number of leaving parents plus the number of packets dropped during transmission is smaller than the number of redundant *FEC* packets, all the missing packets can still be recovered by the *FEC* scheme. That means a peer will have more time to find a new parent in the proposed system. The *FEC* will take care of the missing packets before locating to a new parent to replace the departed one.

3. Performance Evaluation

This section presents the simulation. We use MSVoD to denote the proposed system for simplification. In order to compare our scheme with the traditional "cache-and-relay" approach, P2VoD [8] is also simulated.

3.1. Simulation Setting

In our simulation, the underlying network topology is created using the GT-ITM [10]. The whole network topology consists of one transit network (with 4 nodes), and 12 stub domains (with 96 nodes). The video server is placed on a transit node and all peers are randomly placed into one node of the stub domains. The peer arrival follows the *Poisson process.* The length of video is 3600 seconds, and the viewing time of each peer is exponentially distributed with mean 80% of the total length of video. We denote the buffer threshold by T. In the simulation, the upload bandwidth capacity of peers to share follows a *Bounded Pareto distribution.* The upload bandwidth is generated from *BP*(0.4, 15, 0.7) times 300 with lower bound 0.4, upper bound 15, and scale 0.7. The video playback bit rate is set to be 200 Kbits/sec.. However, due to the *FEC* packets, the real transmission bit rate is

$$\textbf{\textit{Video bit rate}} \times \frac{N}{k},$$

where N and k are parameters of *FEC*. We perform the simulations by using NS-2 [11]. The simulation time is 4500 seconds.

3.2. Server Stress

Figure 4 shows the server stress in MSVoD and P2VoD with different T from 100, 150, to 200 seconds and employing *FEC*(6, 4). The server stress is measured as the workload bandwidth of the server. The arrival rate is set to be six peers arriving per minute. In the result, the server stress in MSVoD is much lower than P2VoD. That is because in MSVoD, the sending peer may send video data for one child peer with fewer sub-streams than totally six sub-streams. But in P2VoD, the receiver must find the sender who can provide six sub-streams at least, otherwise it must connect to server directly to get video data. So, the bandwidth of each peer in the MSVoD system can be consumed more. In the end, the server stress is

convergent to the maximum workload bandwidth that is because the number of coming peers and leaving peers are reaching to a balance. We will compare the maximum throughput with different T latter.

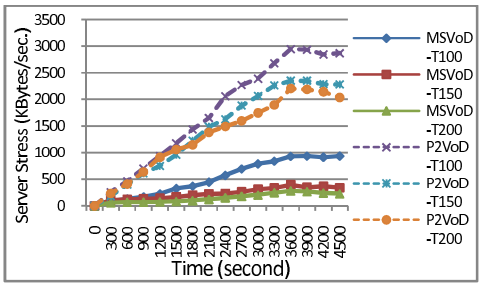

Figure 4. The server stress of P2VoD and MSVoD with different buffer threshold.

Figure 5 shows that the maximum server stress as different normalized workload and T. The value of T is fixed as 100, 150, and 200 seconds. The *normalized workload* means the total number of peers arriving into the system during a simulation run is determined as (Arrival Rate × Length of the Video). In the result, with a fixed T, the server stress decreases in MSVoD, while increasing in P2VoD when the normalized workload increases. Thus, in MSVoD, the number of peers in parent candidates list of each peer is also increased. So, each peer can always find enough parents with enough bandwidth to obtain video data easily without connecting to the server directly. However, in P2VoD, even though there are more candidates, but almost peers have no enough bandwidth to provide video streaming to other peers. Then, more peers contact to the server. Furthermore, in MSVoD, the server stress can be decreased slowly by increasing T.

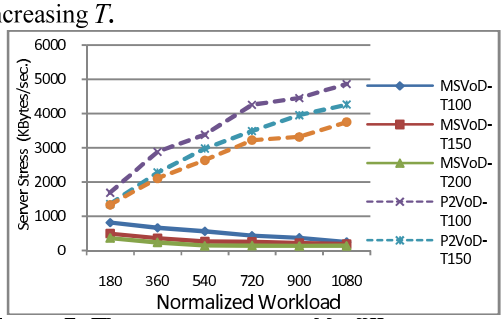

Figure 5. The server stress with different normalized workload and values of buffer threshold.

However, in Fig. 5, the server stress with T is 100 is similar to T is 150 and 200 as normalized workload is 1080. That means that most peers have enough peers in the candidate list if setting the T is 100. This result shows that T is not necessary to be large.

3.3. Peak Signal-to-Noise Ratio

The packet loss will result is the video quality decreased. In this section, we will measure the peak signal-to-noise ratio (PSNR) of video quality. The *news_cif.yuv* is used as the raw video and PSNR is measured by EvalVid[12]. And then, this YUV video is encoded into the MPEG-4 format video by ffmpeg [13]. We set the frame rate to 30 frames/sec..

Figures 6 (a) and (b) show the results of the PSNR measurement of peers in MSVoD and P2VoD respectively. This result shows the PSNR can be affected by different protection quality due to employing different $FEC(N, k)$. In MSVoD, the PSNR is keep higher quality when using $FEC(4, 3)$, and $FEC(5, 3)$ while lower quality when using $FEC(5, 4)$, and $FEC(6, 5)$. And the PSNR in MSVoD is stable as later peer arriving while is still decreased in P2VoD. The result shows that using $FEC(5, 3)$, each peer can get the best video quality both in MSVoD and P2VoD. Although the tendencies of result in P2VoD are similarly in MSVoD, but the decrease rate of PSNR in P2VoD is faster than in MSVoD. There is a smaller figure in Figs. 6 (a) and (b) show the PSNR by $FEC(5, 3)$ and $FEC(6, 4)$ specifically.

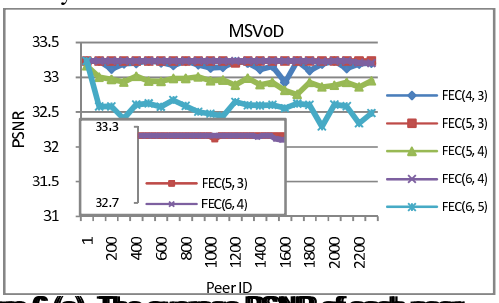

Figure 6 (a). The average PSNR of each peer in MSVoD.

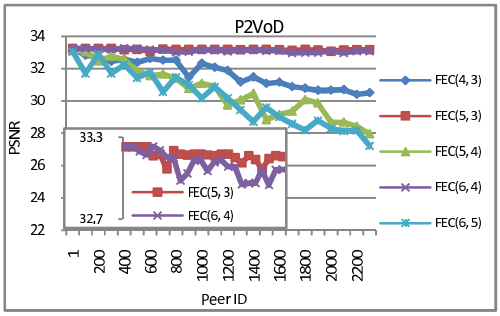

Figure 6 (b). The average PSNR of each peer in P2VoD.

3.4. Control Overhead

Figure 7 shows the control overhead of P2VoD and MSVoD with different *T*. The control overhead is measured by how many number of peers that a new peer has to contact during the viewing time in the system. That includes number of peers that is contacted during joining process and failure recovery process. Since each peer in MSVoD needs to connect *N* sources, the control overhead is more than single source VoD during joining process. But in failure recovery process, the child peer of a failure peer in MSVoD can find recovery peer with enough bandwidth easier than single source VoD. Thus, the control overhead of MSVoD is not *N* times than P2VoD. Here, the control overhead of MSVoD is totally about four to six times of P2VoD while the *N* is six.

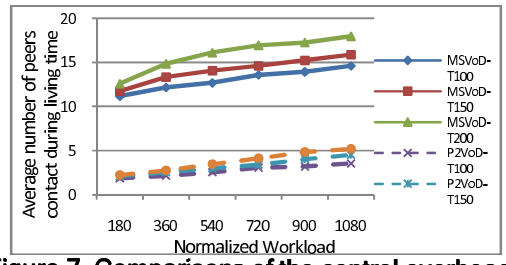

Figure 7. Comparisons of the control overhead of MSVoD and P2VoD with different thresholds.

4. Conclusions and Future Work

This work has proposed a video-on-demand P2P system using the concept of multi-source and *FEC* recovery method to minimize the workload of server and maintain video quality. In this work, each peer preserves partial video data within a buffer threshold and the new arrival peer gets these data from multiple sources. Here, we consider that peers with low upload bandwidth that is determined by each peer and the upload bandwidth may be lower than the video rate. Thus, since employing the concept of a multi-source, the workload of server can be decreased that results as each peer can divide the upload bandwidth into several units. So the upload bandwidth of peers can be used more efficiently in multi-source P2P system than the traditional single source P2P systems.

Due to connect multiple peers, the startup delay in multi-source system is higher than the traditional P2P system. This is a key point in VoD system and we will deal with this problem hereafter. Besides, another challenged problem of VoD system is the interactive operation, such as forward and rewind. This operation is not discussed here. Thus, we will focus on this design in the future.

References

[1]. B. Cohen, "Incentive Build Robustness in BitTorrent," in *Proc. P2P Economics Workshop*, May 2003.

[2]. PPStream, http://www.ppstream.com

[3]. M. Hefeeda, A. Habib, B. Botev, D. Xu, B. Bhargava, "PROMISE: Peer-to-Peer Media Streaming Using CollectCast," in *Proc. of the 11th ACM international conference on Multimedia*, Berkeley, CA, USA, Nov. 2003.

[4]. M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh "Splitstream: high-bandwidth multicast in a cooperative environment," *ACM Symposium on Operating Systems Principles (SOSP"03) Bolton Landing*, NY, Oct. 2003.

[5]. P. J. Wu, C. N. Lee, V. Gau, and J. N. Hwang, "Overcoming Burst Packet Loss in Peer-to-Peer Live Streaming Systems," *IEEE International Symposium on Circuits and Systems (ISCAS)*, Seattle, USA, May 2008.

[6]. Y. Guo, K. Suh, J. Kurose, D. Towsley "P2Cast: peer-to-peer patching scheme for VoD service," in *Proc. of the 12th World Wide Web conference (WWW)*, Budapest, Hungary, May 2002.

[7]. T.T. Do, K.A. Hua, and M.A. Tantaoui, "P2VoD: Providing fault tolerant video-on-demand streaming in peer-to-peer environment," in *Proc. of IEEE International Conference on Communications*, Paris, France, June 2004.

[8]. T.T. Do, K.A. Hua, and M.A. Tantaoui, "Robust video-on-demand streaming in peer-to-peer environments," *ELSEVIER Computer Communications*, vol. 31, Issue 3, pp. 506-519, Feb. 2008

[9]. W. T. Tan and A. Zakhor, "Video multicast using layered FEC and scalable compression," *IEEE Transaction Circuits and System for Video Technology (TCSVT)*, vol. 11, no. 3, pp. 373-386, Mar. 2001.

[10]. E. W Zegura, K. L. Calvert, S. Bhattacharjee, "How to model an interwork," in *Proc. of IEEE INFOCOM*, San Francisco, CA, 1996.

[11]. The network simulator–ns-2, http://www.isi.edu/nsnam/ns/.

[12]. EvalVid, http://www.tkn.tu-berlin.de/research/evalvid/.

[13]. ffmpeg, http://ffmpeg.sourceforge.net/index.php.