Live Video Streaming Using P2P and SVC

Tsung-Chieh Lee[1], Pin-Chuan Liu[2], Woei-Luen Shyu[1], and Chen-Yih Wu[1]

[1] Industrial Technology Research Institute
Bldg. 14, 195, Sec. 4, Chung Hsing Rd., Chutung, Hsinchu, Taiwan 31040, R.O.C.
{cjlee,wlshyu,Kevin_Wu}@itri.org.tw
http://www.itri.org.tw
[2] National Tsing Hua University
101, Sec. 2, Kuang-Fu Rd., Hsinchu, Taiwan 30013, R.O.C.
flash@rtlab.cs.nthu.edu.tw
http://www.nthu.edu.tw

Abstract. The research of Video Streaming has often focused on the methodologies of P2P protocols, such as peer selection, network structure, group organization, etc. In addition, streaming mechanism and system deployment are significant to provide a Video Streaming service. Research in this field, however, is scant. On the other hand, the layered video codec provides scalability when environment is divergent due to different transmission rate, computational power, and so on. Recently, Scalable Video Coding (SVC), which is a layered video codec, has been standardized and caught much attention. The study is to explore how to provide Video Streaming service by employing P2P and SVC. A system architecture which involves video layering, dynamic Segment seeding and scheduling, and Segment downloading and sharing is developed. To provide high quality Live or On-demand P2P Video Streaming service, a Video Streaming system, GaiaSharp, is implemented and deployed, and the experience is shown to explain the importance of layered video codec.

Keywords: P2P, SVC, Video Streaming.

1 Introduction

With the innovation of transmission technology and improvement of hardware, services used to rely on specific devices are now available in computers; for example, composing a song is not always done in studio, because computer software helps musician accomplish their work at any place. In the mean while, new services are proposed or combined across different domains, and applications which were stand alone are connected to each other. The computer and Internet have been changing our life style rapidly. From desktop to invisible equipment, information is exchanged with our friends or transmitted to someone we do not know. Recently, Digital Home has become a hot topic, and to watch TV is major entertainment. Computers were able to convert analog signal with a TV adapter plugged and configured correctly - this is not convenient for most people. Is there any way to achieve this goal without additional requirement? Traditional

G. Pavlou, T. Ahmed, and T. Dagiuklas (Eds.): MMNS 2008, LNCS 5274, pp. 104–113, 2008.

Video Streaming protocols have been designed, but nowadays, they cannot afford the growth of users and higher video quality because of hardware overload or transmission bandwidth. P2P has been caught much attention for data delivery in large scale overlay; although P2P solves the bottleneck (e.g. the maximum concurrent connections), it brings up new issues and makes many applications re-designed. In the mean while, a standard of layered video codec, Scalable Video Coding (SVC), is published. The layered structure of video codec yields flexible quality and is configurable based on computation power, transmission bandwidth, and monitor size, etc. Different from research in P2P protocol and SVC structure, we propose a mechanism to provide a Video Streaming system by taking advantages of P2P and SVC. As P2P which is an unstable network, this article describes issues of deploying live P2P Video Streaming system and provides dynamic configuration to support better service quality.

In this article, we introduce P2P Video Streaming applications, systems, and Scalable Video Coding in Section II. In Section III, we present our work to build a Video Streaming system based on P2P and SVC. The system deployment and configuration of GaiaSharp, a live Video Streaming system we are developing, is shown in Section IV. The conclusion and future work are described in the last section.

2 Related Work

Peer-to-peer (P2P) overlay networks is a kind of promising architecture to scalability share files [1][2], and audio streams [3]. Recent research further focus on transmitting video streams via p2p. Those works can be divided into two categories: live and recorded. The latter one, e.g. Video-on-dement (VoD), in fact transfers video data in the form of files so existing p2p transmission mechanisms [1][2] can be applied. As any regular files, video files are first split into equal-size pieces and then uploaded to interested peers. These peers further exchange acquired pieces in parallel, known as the swarming procedure. Once enough continuous pieces from the beginning of a video file are downloaded, these pieces are appended and played [4]. However, swarming exchanges pieces randomly, rather than in sequence, and may cause long initial waiting time. To overcome this problem, pieces needed in near future should be downloaded first. Both Vlavianos et al. [5] and Shah and Paris [6] proposed similar piece selection mechanisms. The former category further integrates p2p and video streams by temporally splitting video files into chunks [7][8], e.g. each chunk contains one-second video and/or audio data. Thus, if a chunk is not downloaded in time, video playing can skip or wait for this chunk [9] and provide better user experience than the latter category. They take much effort on the mechanism of P2P protocol to improve the performance of Video Streaming. Layered encoding and multiple description coding are suitable for many applications over heterogeneous networks. SVC, a layered video codec, is an extension of H.264/MPEG-4 AVC [10]. Venkata et al. [11] present a tree-based algorithm to find out path diversity, and a framework

for multiple description codec. Pierpaolo et al. [12] implement SPPM protocol, a prioritized mechanism, to construct a multicast tree with six-way handshake process. Mubashar and Toufik [13] provide smooth media streaming by peer selection methodology. Devices can partially download decodable stream based on their capabilities, such as computation power and network bandwidth. Our work is going to provide a solution for Video Streaming service, and focuses on how to make use of SVC and P2P features to deployment a streaming system. That is, our mechanism can be applied with any P2P protocol.

3 Proposed Mechanism

Relative to other transmission protocol, P2P network is more un-stable. In this section, in order to provide stability of Video Streaming service in an unstable environment, we present a mechanism to take advantage of P2P and SVC, and configure parameters dynamically while downloading video segments. The mechanism is divided into *Channel Server*, and *Client Engine* (Fig. 1).

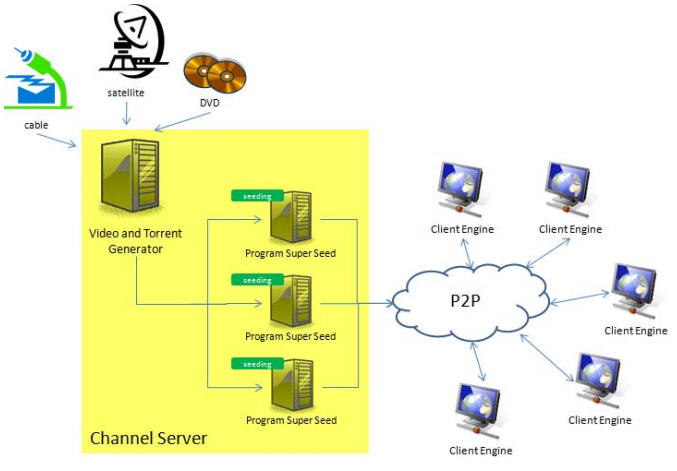

Fig. 1. The mechanism of Channel Server and Client Engine

3.1 Channel Server

In our mechanism, Channel Server receives video signal from different source, and splits signal into Segments and Crumbs with layers based on SVC information; moreover, torrents are made, scheduled and dispatched into Program Super Seed for streaming out. Figure 2 shows main modules and steps from processing video signal to share Crumbs. There may be more than one Program Super Seed to enhance sharing performance. Client Engines download Crumbs with P2P protocol from Program Super Seeds or other Client Engines. With Crumbs,

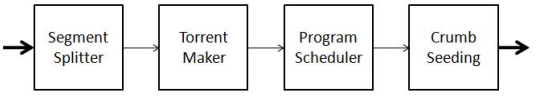

Fig. 2. Main modules from video signal to torrents

users watch video which is composed by SVC. The main steps and modules are shown in the following:

1) `Segment Splitter`: The video signal is received, converted, and split into Segments by GOPs (Fig. 3). According to SVC specification, a Segment is

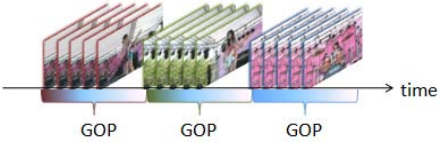

Fig. 3. Video is split into Segments by GOPs

organized with a Header, a Base Layer, and several Enhancement Layers [10] which is various via different video qualities (Fig. 4). Moreover, one Enhance-

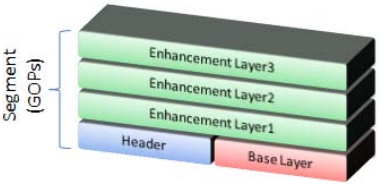

Fig. 4. A Segment is organized according to SVC specification

ment Layer could be partitioned by SVC into dimensions. Let v be the video with n Segments, s_i be the ith Segment of v, and for s_i, let H_i be Header, B_i be Base Layer, and E_{il} be Enhancement Layer l (l is no more than 3 [10]), we know

$$v = \sum_{i=1}^{n} s_i = \sum_{i=1}^{n} \left(H_i + B_i + \sum_{l=1}^{3} E_{il} \right). \tag{1}$$

Different with other protocols, P2P network is an unstable environment, because the behavior of a peer to join and leave the service network in P2P environment may cause service unstable to another peer. On the other hand, buffer time is an important factor of sharing performance in P2P Video Streaming [14]. The more the buffer time is, the more delay-live the Video Streaming is, but the more served users and better streaming quality are. As a result, *dynamic amount of GOPs of a Segment* is required. Let $|G_{max}|$ and

$|G_{min}|$ be the maximum and minimum amount GOPs of a Segment respectively, and $|G_i|$ is the amount of GOPs of s_i, so $|G_{min}| \leq |G_i| \leq |G_{max}|$; let ρ_i denotes the ratio of *successful downloading* s_i among all Client Engines, and we set

$$|G_i| = \begin{cases} |G_{i-1}| + 1, \text{ if } \rho_i \text{ is greater than a threshold} \\ |G_{i-1}| - 1, \text{ else} \end{cases}$$

Based on this dynamic configuration, if $|G_i|$ is smaller, the Client Engine watches TV programs approximate live. However, the Segment which is too small causes an overhead of sharing in P2P, and that is the reason why a lower bound $|G_{min}|$ is required. In Section 3.2, we will define what *successful downloading* is.

2) `Torrent Maker`: Let the dimension of E_{il} is $|E_{il}|$, and the jth dimension of E_{il} is e_{ilj}, $v = \sum_{i=1}^{n} \left(H_i + B_i + \sum_{j=1}^{|E_{i1}|} e_{i1j} + \sum_{j=1}^{|E_{i2}|} e_{i2j} + \sum_{j=1}^{|E_{i3}|} e_{i3j} \right)$ is derived from Equation 1. Because the amount of Enhancement Layers is various with different video qualities, *dil* might be 0 if Enhancement Layer l does not exist. Additionally, SVC is able to compose a video segment as long as both Header and Base Layer exist, so we break borders between Enhancement Layer 1, Layer 2, and Layer 3; in other words, the Enhancement Layers are regarded as a set of dimensions of Layer 1, Layer 2, and Layer 3, and the dimension of Enhancement Layers is $|E_{i1}|+|E_{i2}|+|E_{i3}|$. Since Header and Base Layer are essential for a Segment composition, we group Header and Base Layer as an atomic element, called Crumb, and view each dimension of E_i as a Crumb, too; hence, there are $1 + |E_{i1}| + |E_{i2}| + |E_{i3}|$ Crumbs for s_i. Let c_{ik} be kth Crumb of s_i, we know

$$v = \sum_{i=1}^{n} s_i = \sum_{i=1}^{n} \left(\sum_{k=1}^{1+|E_{i1}|+|E_{i2}|+|E_{i3}|} c_{ik} \right). \tag{2}$$

We make every Crumb into a torrent. Figure 5 is an example to make Crumbs of Segment No.78, s_{78}, into torrents: There are 2 Enhancement Layers in $|s_{78}|$, and dimension of Enhancement Layer 1 and Enhancement Layer 2 is 3 and 5 respectively. Torrent Maker makes 9 Crumbs into 9 torrents.

3) `Program Scheduler`: Channel Server maintains a window which keeps tracks on sequence numbers of available Segments which imply the buffered Segments in Channel Server and Client Engine for P2P sharing. Let $|w|$ be the windows size, and x be the sequence number of the latest Segment, the sequence numbers of available Segments range from $x - |w| + 1$ to x. In other words, only $s_{x-|w|+1}$, $s_{x-|w|+2}$,..., and s_x are available live Segments. Besides, Channel Server maintains another incremental sequence number, x', which is an initial index, for every Client Engine that once connects to this streaming service network. Note that it is a bad idea to set $x' = x$ in Channel Server, because every Client Engine tends to download the same Segment, and the

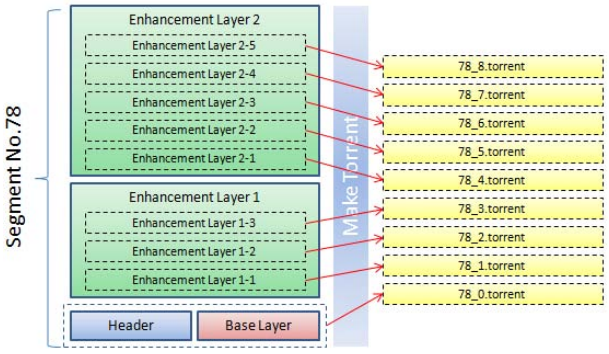

Fig. 5. Crumbs of Segment No.78 are made into torrents

performance of sharing cannot be obvious, but to set $x' = x - |w| + 1$ decreases successful downloading rate as well.

4) `Crumb Seeding`: Initially, only Program Super Seeds have Crumbs of Segments of video. By downloading torrents of Crumbs, Client Engines exchange information with the tracker to download and share Crumbs. After several Client Engines download successfully, the loading of Program Super Seeds is spread, and the advantage of P2P is amplified.

3.2 Client Engine

The idea of P2P network is sharing; on the other hand, the most important feature of live Video Streaming is timing. As a result, to setup a live P2P Video Sreaming system, which is also a timely sharing system, is going to face a trade-off between delay and efficiency of P2P. We design a schema for Client Engines to execute downloading tasks, and the schema comes up with two selection models: a) Segment Selection, and b) Crumb Selection.

a) `Segment Selection`: As mentioned in *Program Scheduler*, Channel Server maintains x' for every Client Engine connecting to this service network, and $x - |w| + 1 < x' < x$. Let $|s_i|$ be the playback time interval of s_i. In initial stage, the Client Engine starts downloading $s_{x'}$ at time $t_{x'}$, $s_{x'}$ has to be downloaded successfully by $t_{x'} + |s_{x'}|$ to make sure the downloaded Segment is live. That is, Client Engine has $|s_{x'}|$ to download $s_{x'}$. After $t_{x'} + |s_{x'}|$, $s_{x'}$ will be expired no matter it is downloaded successfully or not, even Client Engine is sharing $s_{x'}$ with others. Note that the Segment which Client Engine is downloading is the most urgent than any other Segments, so Client Engine focuses on downloading a single Segment at a time. Two situations are separately discussed as follows:

   – If $s_{x'}$ is downloaded successfully at time $t_{x'+1}$, Client Engine starts to download $s_{x'+1}$. Because the previous Segment is downloaded successfully, Client Engine earns a *Time Extension* $t_{x'} + |s_{x'}| - t_{x'+1}$ for current

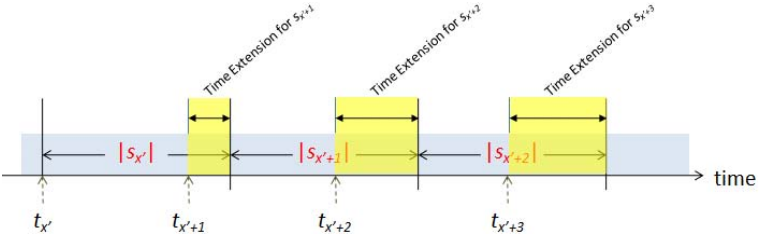

Fig. 6. Time Extensions for downloading $s_{x'+1}$, $s_{x'+2}$, and $s_{x'+3}$

Segment to be downloaded; that is, Client Engine has $|s_{x'+1}| + (t_{x'} + |s_{x'}| - t_{x'+1})$ to download $s_{x'+1}$, and so on for the following Segments. Figure 6 illustrates Time Extensions for downloading $s_{x'+1}$, $s_{x'+2}$, and $s_{x'+3}$. The downloading sequence number $x' + k$ keeps looking forward until it reaches x which is the upper bound of available Segments. This racing behavior makes as many Client Engines synchronize Segments with Program Super Seeds as possible (Note that for a Client Engine, Segments are not always downloaded from Program Super Seeds). Therefore, more and more Client Engines which sharing the same Segments as Program Super Seeds do come up, and can be regarded as new Program Super Seeds.

– If Segment s_i is not downloaded successfully at time t_i, the Client Engine goes back to initial stage and refresh the sequence number x' from Channel Server.

b) `Crumb Selection`: In our mechanism, the definition of a *successful downloading task* is refined due to the characteristic of SVC. As mentioned in *Torrent Maker*, SVC is able to compose a video segment as long as Header and Base Layer exist; hence, if Header and Base Layer downloaded, the downloading task is *successful*. Comparing with a successful downloading task, a *complete downloading task* means all required dimensions of a Segment are downloaded. There are $1 + |E_{x'1}| + |E_{x'2}| + |E_{x'3}|$ Crumbs for Segment $s_{x'}$, and let $c_{x'1}$ denote Crumb of Header and Base Layer. For a particular Client Engine in initial stage, transmission rate between Program Super Seed is evaluated, said $\mathbb{R}$, and dimensions that are required for $s_{x'}$ are from 1 to $\mathbb{D}_{x'}$, where $\mathbb{D}_{x'}$ is the max d satisfying

$$\frac{\sum_{k=1}^{d} c_{x'k}}{\mathbb{R}} \leq |s_{x'}| + TimeExtension(s_{x'}). \tag{3}$$

Any $\mathbb{D}_i$ for s_i sustains $1 < \mathbb{D}_i \leq 1 + |E_{i1}| + |E_{i2}| + |E_{i3}|$. If all $\mathbb{D}_{x'}$ Crumbs are downloaded (which means *completely*), $\mathbb{D}_{x'+1}$ is set to $\mathbb{D}_{x'} + 1$; otherwise, $\mathbb{D}_{x'+1}$ is set to $\mathbb{D}_{x'} - 1$. This *dynamic dimensions* for Crumb downloading is suitable for Video Streaming in P2P network, because $\mathbb{R}$ of every Client Engine is not the same, and a Client Engine is not always in poor transmission rate, not to mention that Segment can be downloaded from any other Client

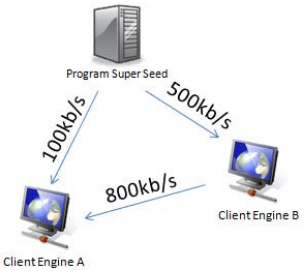

Fig. 7. The downloading rate of a Client Engine is not unchangeable

Engine. Figure 7 illustrates a scenario that Client Engine A tries to download two Segments s, and s' in the same size 256KB from Program Super Seed and Client Engine B separately; theoretically, it takes 20.48 seconds for s, but takes only 6.656 seconds for s' after Client Engine B gets s'.

4 System Deployment and Configuration

In this section, how our Video Streaming system, GaiaSharp, is deployed and configured is presented. Figure 8 is a snap shot of Client Engine of GaiaSharp, which downloads Segments and EPG (Electronic Program Guide) to provide Live or On-demand Video Streaming service.

The P2P engine of GaiaSharp is MonoTorrent, which is an open source of Mono Project [15]. We develop two systems in different video codecs: Windows

Fig. 8. A snap shot of Client Engine of GaiaSharp

Media Codec (without layered structure) and H.264 (with layered structure by SVC). Firstly, Channel Server generates every Segment in WMV format 640x480 resolution, 600kbps, and 29.97 fps. Every Segment size is about 20 seconds, and 1.5MB; based on [14], piece size of torrent are made in 256KB to perform better P2P sharing efficiency. The window size $|w|$ is 5, which means there are 5 Segments seeding at the same time; x' is $x-3$ to act as a buffer for P2P sharing. In LAN, the video displayed smoothly and works fine, but in WAN with one Super Seed of upload bandwidth 2MB/s, it performance is unacceptable. For watching live TV, the lower bound of transmission rate between a Super Seed and a Client Engine is 600kbps which is the minimum requirement to display video without downloading any Segment failed; that is not a workable requirement for our Channel Server and most Client Engines in practice. According to the experiment result, we apply the proposed mechanism and split video into Segments in H.264 format, the resolution is 640x480, and Enhancement Layers are partitioned into 30 Crumbs; the bit rate is 128kbps for Base Layer, 512kbps for Enhancement Layer 1, 2048kbps for Enhancement Layer 2; for a Segment of 20 seconds, the Crumb of Base Layer is about 360KB, Enhancement Layer 1 is partitioned into 4 Crumbs in 320KB, and Enhancement Layer 2 is partitioned into 10 Crumbs in 512KB; we set G_{max} and G_{min} about 20 and 5 seconds respectively. As a result, the best quality is bettered from 600kbps to 2688kbps, and more importantly, the minimum requirement to display video is lowered from 600kbps to 144kbps which is acceptable even for ADSL clients with bandwidth in 256/64.

5 Conclusion

Although P2P file sharing applications have become popular, previous research seldom investigates streaming mechanism and system deployment. A Channel Server and Client Engine strategy for Live Video Streaming based on the advantages of P2P and SVC is developed. By breaking and re-organizing layers of Segments into Crumbs, a successful and complete downloading task is re-defined. Moreover, due to the un-stability of P2P network, GaiaSharp dynamically configures Segment seeding and scheduling to provide scalability for live P2P Video Streaming system. Making use of SVC, the requirement of transmission rate is much smaller than un-layered video codec. There is still much work to be done. For example, there are Channel Management algorithms to reduce the waiting time during switching and provide better user experience, and is it possible to provide Video Streaming service in HD quality?

References

1. Saroiu, S., Gummadi, K.P., Gribble, S.D.: Measuring and analyzing the characteristics of Napster and Gnutella hosts. Multimedia Systems 9(2), 170–184 (2003)
2. Cohen, B.: Incentives build robustness in BitTorrent. In: The First Workshop on the Economics of Peer-to-Peer Systems, Berkeley (2003)
3. Baset, S.A., Schulzrinne, H.G.: An analysis of the Skype peer-to-peer Internet telephony protocol. In: IEEE INFOCOM 2006, Barcelona, Spain, pp. 1–11 (2006)

4. Dana, C., Li, D., Harrison, D., Chuah, C.-N.: BASS: BitTorrent Assisted Streaming System for Video-on-Demand. In: IEEE 7th Workshop on Multimedia Signal Processing, Shanghai, pp. 1–4 (2005)
5. Vlavianos, A., Iliofotou, M., Faloutsos, M.: BiToS: Enhancing BitTorrent for Supporting Streaming Applications. In: INFOCOM 2006, Barcelona, Spain, pp. 1–6 (2006)
6. Shah, P., Paris, J.-F.: Peer-to-Peer Multimedia Streaming Using BitTorrent. In: IPCCC 2007, New Orleans, LA, pp. 340–347 (2007)
7. Xie, S., Li, B., Keung, G.Y., Zhang, X.: Coolstreaming: Design, Theory, and Practice. IEEE Trans. on Multimedia 9(8), 1661–1671 (2007)
8. Pplive, `http://www.pplive.com`
9. Hei, X., Liang, C., Liang, J., Liu, Y., Ross, K.W.: A Measurement Study of a Large-Scale P2P IPTV System. IEEE Transactions on Multimedia 9(8), 1672–1687 (2007)
10. Schwarz, H., Marpe, D., Wiegand, T.: Overview of the Scalable Video Coding Extension of the H.264/AVC Standard. IEEE Transactions on Circuits and Systems for Video Technology (2007)
11. Padmanabhan, V.N., Wang, H.J., Chou, P.A.: Resilient peer-to-peer streaming. In: 11th IEEE International Conference on Network Protocols, Atlanta, USA, pp. 16–27 (2003)
12. Baccichet, P., Schierl, T., Wiegand, T., Girod, B.: Low-delay Peer-to-Peer Streaming using Scalable Video Coding. Packet Video, Lausanne, Switzerland, pp. 173–181 (2007)
13. Mushtaq, M., Ahmed, T.: Smooth Video Delivery for SVC based Media Streaming over P2P Networks. In: 5th IEEE Consumer Communications and Networking Conference, Las Vegas, NV, pp. 447–451 (2008)
14. Tewari, S., Kleinrock, L.: Analytical Model for BitTorrent-based Live Video Streaming. In: 4th IEEE Consumer Communications and Networking Conference, Las Vegas, NV, USA, pp. 976–980 (2007)
15. Mono Project, `http://monotorrent.com/`