Improving the Availability of Scalable on-demand Streams by Dynamic Buffering on P2P Networks

Chow-Sing Lin
Dept. of Computer Science and Information
National University of Tainan
Tainan, Taiwan (R.O.C.)
[e-mail: mikelin@mail.nutn.edu.tw]

Received April 28, 2010; revised June 4, 2010; accepted June 11, 2010; published August 27, 2010

Abstract

In peer-to-peer (P2P) on-demand streaming networks, the alleviation of server load depends on reciprocal stream sharing among peers. In general, on-demand video services enable clients to watch videos from beginning to end. As long as clients are able to buffer the initial part of the video they are watching, on-demand service can provide access to the video to the next clients who request to watch it. Therefore, the key challenge is how to keep the initial part of a video in a peer's buffer for as long as possible, and thus maximize the availability of a video for stream relay. In addition, to address the issues of delivering data on lossy network and providing scalable quality of services for clients, the adoption of multiple description coding (MDC) has been proven as a feasible resolution by much research work. In this paper, we propose a novel caching scheme for P2P on-demand streaming, called *Dynamic Buffering*. The proposed Dynamic Buffering relies on the feature of MDC to gradually reduce the number of cached descriptions held in a client's buffers, once the buffer is full. Preserving as many initial parts of descriptions in the buffer as possible, instead of losing them all at one time, effectively extends peers' service time. In addition, this study proposes a description distribution balancing scheme to further improve the use of resources. Simulation experiments show that Dynamic Buffering can make efficient use of cache space, reduce server bandwidth consumption, and increase the number of peers being served.

Keywords: Streaming, peer-to-peer, dynamic buffering, video-on-demand, multiple description coding

This research was partially supported by National Science Concuil of Taiwan under contract 5-2221-E-218-015 -MY2.

DOI: 10.3837/tiis.2010.08.003

1. Introduction

With the rapid advances in Internet and networking technologies, there has been a dramatic increase in user demand for various multimedia applications, especially for media streaming services. In general, streaming services can be categorized into live streaming and on-demand streaming. In on-demand streaming systems, the streamed content (such as movies) is pre-recorded and available from the streaming server in the full version. Users require not only continuous but also complete playback of the requested streaming contents, regardless of when a request is made for service. In on-demand streaming systems, peers can relay current video blocks as well as initial blocks to subsequent peers, only the initial blocks remain cached in the buffer. Assuming peers' forwarding bandwidth is sufficient, the ability of peers to relay blocks of a video to other peers depends mainly on whether the initial blocks of the video are held in the cache. In contrast, live streaming content is typically captured from live events, such as ball games and the news. For peers logging on late, the missing initial part of the video is irrelevant to the QoS they perceive, and is ignored. Users watch the streaming content from the point where they first joined the broadcast, instead of from the beginning, and therefore, any peer with sufficient forwarding bandwidth can provide services to later peers. In this paper, we only focus on on-demand video services.

In on-demand media streaming systems, a streaming server is equipped with a limited amount of outbound bandwidth as well as a number of videos. Users issue requests to watch any one of the provided videos. Upon receiving a user's request, on-demand streaming systems must ensure the user receive a continuous and complete playback of the requested video. On-demand media streaming services generally require streaming servers to dedicate a substantial amount of forwarding bandwidth to each user, for a long period of time. Such resource consumption of media streaming has driven most service providers to shift the system design from traditional Client-Server architecture to Peer-to-Peer (P2P) architecture [1][2][3][4]. In P2P systems, a user (or peer) both consume resources from the system, and contributes resources to the system [5]. With the aid of users, the scalability of an on-demand streaming systems can be increased and more users can be served.

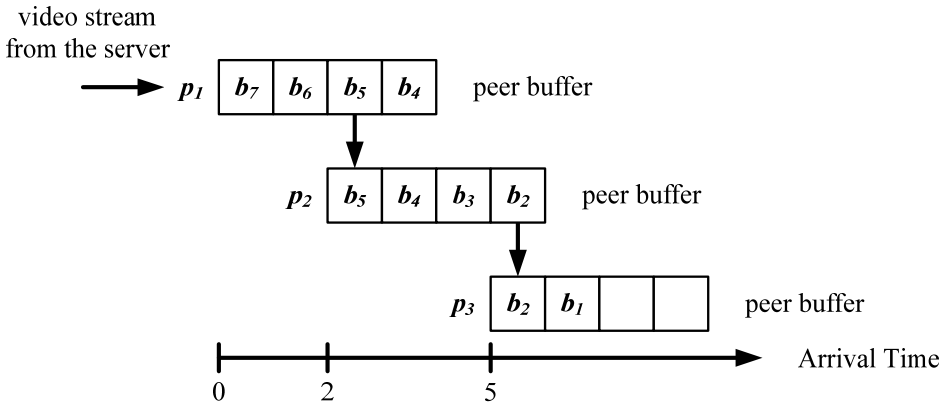


Fig. 1. A snapshot of peers' buffers for the cache-and-relay scheme at time 7.

In the past, there has been a great deal of work done on providing on-demand streaming services in a P2P manner [6][7][8][9][10]. Most of these systems employ the cache-and-relay

schemes [11][12] that require each peer to cache the most recent video stream it receives. As long as the initial part of the video stream remains in its buffer, the peer can then relay the cached stream to late-arriving peers in a pipelining fashion. **Fig. 1** shows a snapshot of peers' buffers at time *7* for the cache-and-relay scheme. To clarify the sequence of a video stream, the video is equally divided into a series of n blocks, denoted as b_1, b_2, …, and b_n. Note that the data cached in the buffer is video blocks that have already been watched. In this example a peer's buffer is capable of caching 4 video blocks at one time; loaded from the left to right in a pipeline fashion. There are three peers p_1, p_2, and p_3, requesting the same video, and arriving at time 0, 2, and 5. Peer p_1 is the first peer to request and receive the video stream from the server. At time 2, p_2 arrives. Since p_1 still holds the initial video block b_1 in its cache, it forwards the cached video blocks to p_2 sequentially. Similarly, when p_3 arrives at time 5, p_2 still holds b_1, and forwards the cached video blocks to p_3. p_2 and p_3 arrive before the preceding peers have filled up the buffers, and are still holding the initial part of the video. Therefore, each peer can receive the entire requested video from its preceding peer(s) in a pipelining fashion. With the cache-and-relay scheme, much of the workload can effectively be shifted from the server to other capable peers, and the server can avoid wasting its limited outbound bandwidth on delivering duplicate video streams to peers with the same request.

In Video-on-Demand (VoD) systems [13][14][15] based on the cache-and-relay scheme described above, the ability of a peer to contribute to the system for serving new peers depends both on the amount of available forwarding bandwidth and the existence of the initial part of a video stream in its buffers. For example, consider peer p_i arriving at time t_i and equipped with a buffer capable of caching s_i time units of the requested video stream. The peer cannot serve (relay blocks to) any peer arriving later than time $(t_i + s_i)$ even though it has adequate available forwarding bandwidth. Because the initial segment of the video stream is not held in its buffer beyond time $t_i + s_i$, it cannot provide subsequent peers with the entire video. In other words, the service capability of p_i is restricted to $[t_i + 1, t_{i1} + s_i]$. Most existing systems based on the conventional cache-and-relay scheme are vulnerable to asynchronous arrival of peers. If the time between peers' arriving is long (in comparison to the buffer size), peers will not be able to cache or relay the video stream efficiently. Thus, the majority of peers have to rely on the streaming server to provide the requested video, leaving the forwarding bandwidth of most peers unused. In such cases, the intention of a P2P streaming system to take advantage of the peer resources (especially forwarding bandwidth) cannot be realized.

To address the issue of delivering video over lossy networks, many Multiple Description Coding (MDC) schemes have been proposed in the last few years. These MDC schemes aim to reduce the effects of transmission errors by coding video signals in two or more correlated sub-streams, call *descriptions*. Normally, descriptions are independently transmitted over separate channels by exploiting path diversity of the multicast tree on P2P networks. Whenever all descriptions arrive without error, the fidelity of a video is maintained. In cases of transmission error due to broken links or a substantial data loss, the lost information can partially be recovered from the received descriptions. In addition, MDC can be used to provide scalable Quality of Service (QoS) for clients with various levels of downloading/forwarding bandwidth. In extreme cases, clients with limited network bandwidth, such as mobile devices, could still enjoy low quality videos by receiving only one description. The perceived QoS of watching videos depends on the number of descriptions a client receives.

To address the aforementioned problems inherent in conventional cache-and-relay schemes, this study seeks to take advantage of the flexible encoding potential of MDC and proposes a novel caching scheme, called *Dynamic Buffering*. In conventional cache-and-relay schemes, the service capability of peers will suddenly disappears after the peers' buffer fills up with

streaming data. In this proposed Dynamic Buffering scheme, the service capability of peer degrades more gradually. We expect the Dynamic Buffering scheme to deal with asynchronous requests more effectively, and thus to achieve a more efficient utilization of peer resources, further alleviating the workload imposed on the server. To the best of our knowledge, this is the first paper to propose the application of dynamic buffering schemes to on-demand MDC streaming for P2P networks. Simulation results show that the proposed dynamic buffering scheme significantly outperforms conventional cache-and-relay schemes such as CoopNet.

Video streaming systems with the proposed dynamic buffering scheme require full knowledge and central management of all peer resource usage to find source peers (parents) and to balance the distribution of descriptions. Such centralized management could be implemented from an existing central node in the system, although this may cause scalability issues when serving a huge number of clients. Generally, regardless of the employed streaming protocol, there must be a certain amount of centralization in P2P streaming systems. For example, administrators of a commercial streaming system may require a central node to collect and manage peer information for the sake of accounting, or statistics. Even in general P2P streaming systems, peers require an entry or bootstrap node to facilitate the joining or subscribing to the desired videos. Our proposed approach just piggybacks on existing central nodes such as these. Additional control and management overhead on the central server could lead to scalability issues. Generally, centralized systems can be scaled up by applying clustering technology to the virtual servers; in that a cluster of servers is well organized, and sets of clients are partitioned and distributed to server nodes to balance the workload. Such centralized architecture has proven practical, and may be found in many current commercial server sites. In CoopNet [16][17] and our previous work [18], a centralized protocol has been proposed as a feasible method to manage peers on P2P streaming networks, so long as the consumption of memory, network bandwidth, and CPU power usage for the centralized management is acceptable, compared to the number of video streams. In future publications, we will report on the pros and cons of fully distributing such a centralized system, using such common techniques as gossiping [19].

The remainder of this paper is organized as follows. In Section 2, we briefly describe the multiple description coding technique and review a typical approach based on the conventional cache-and relay scheme, the CoopNet. Section 3 presents the ideas of the proposed Dynamic Buffering scheme and description distribution balancing. Section 4 describes the simulation results and analyses. Finally, we conclude the paper in Section 5.

2. Related Work

2.1. Multiple Description Coding

Multiple Description Coding (MDC) [20][21][22][23][24] is a flexible encoding technique capable of encoding a video into multiple sub-streams, called *descriptions*. Any subset of descriptions can be received and decoded by a peer to reconstruct the original video content with fidelity commensurate to the number of descriptions they receive. The more descriptions received, the higher the quality the reconstructed video content is. The flexibility of MDC enables peers with heterogeneous downloading capacity to receive a limited number of descriptions, but still enjoy a limited level of viewing quality according to the individual downloading capacity [24]. The proposed Dynamic Buffering scheme takes advantage of the flexibility of MDC in decoding, to enable a more gradual degradation of peer service

capability, thereby achieving a more efficient utilization of resources.

There has been a lot of research into the realization of MDC with existing coding standards. In particular, a great deal of effort has gone into developing an H.264/AVC standard-compliant multiple description (MD) encoder. Such MDC schemes can generally be divided into two categories:

1. Schemes that exploit the *spatial* correlation between each frame of the sequence, such as Polyphase Spatial Subsampling Multiple Description Coding (PSSMDC) [25] and Multiple Description Motion Coding (MDMC) [26].
2. Schemes that take advantage of the *temporal* correlation between each frame of the sequence, such as Multiple State Video Coder (MSVC) [27] and the Motion-Compensated Multiple Description (MCMD) [28] video coding.

The PSSMDC creates four descriptions from the spatially downsampled polyphase components of the original frames. Each description is compressed independently with the recently deeveloped H.264/AVC video coding standard, and then packetized and sent over an error prone network. In case of error in one or more of the descriptions, frames are masked appropriately at the receiving end before inserting the corrected frames into the corresponding frame buffers. The MDMC scheme splits the block-based motion vector field into two parts using quincunx sampling. The resulting motion vector subfields are successively transmitted to the decoder over separate channels. Finally, the residual information of each macro block is multiplexed into two bit-streams.

In MSVC, the input video is split into sequences of odd and even frames, each being coded as an independent description with its own prediction process and state. Even if one description is completely lost, the other one can be independently decoded, and the reconstructed video can be rendered at half the frame rate. It has been suggested that lost frames in a damaged representation could be recovered by utilizing temporally adjacent frames in descriptions, to recover frames for future prediction. In MCMD schemes, the input sequence is sub-sampled into even and odd frames sequences. The MD encoder is made-up of three dependent coders employing separated frame buffers: a central coder that receives even and odd frames; and two symmetric coders on each side that work on even and odd frames. Coder outputs are merged into two equally-important descriptions, and sent over independent channels.

In general, the MDC schemes based on temporal correlation provide better results than spatially-based techniques. However, choosing the optimal multiple description coding scheme is always a trade-off among many factors, such as content, computation costs, efficiency, etc, and is beyond the scope of this paper.

2.2. CoopNet

CoopNet is a typical P2P media streaming approach based on conventional cache-and-relay schemes. The primary characteristic distinguishing CoopNet from previous systems lies in its attempt to enhance the robustness of streaming services by means of MDC. In CoopNet, the streaming server encodes the streaming content into multiple descriptions using MDC, and distributes these descriptions among peers. Each peer is expected to receive descriptions from as many different parents as possible. With such a multi-parent transmission paradigm, peers are able to continue receiving parts of the streaming content and watch them at a lower quality level, even if some of the parents stop forwarding the streaming data due to accidental failure or intentional departure. However, like most conventional cache-and-relay schemes, CoopNet suffers from asynchronous arrival of peers and inefficient peer resource utilization. For example, suppose that the provided video is encoded into multiple descriptions of the same bit

rate r, and peer p_i receives descriptions n_i with a buffer size b_i. In CoopNet, p_i will cache all received descriptions and its buffer will be saturated with the streaming content $b_i/(n_i*r)$ time units after arriving. The initial parts of all the received descriptions are then squeezed out of the buffer by incoming data, and p_i is no longer able to forward the complete video to any newly arrived peer in a pipelining manner. Therefore, when peers subsequent to p_i arrive too late, such as during off-peak time or requesting unpopular videos, the outbound bandwidth of p_i tends to be unused, and the server is burdened with a greater workload. In Section 4, we evaluate the performance of proposed Dynamic Buffering scheme by comparing it with CoopNet to show how effectively Dynamic Buffering scheme addresses the asynchronous arrivals of peers and how much workload can be shifted from the streaming server to peers.

3. Dynamic Buffering

In the design of a P2P on-demand streaming systems based on conventional cache-and-relay schemes, a key issue is how to maintain peer service capability for as long as possible. "Service capability," refers to the ability of peers to serve subsequent peers with the entire video stream they request. More specifically, a peer p_i is considered to have service capacity for another peer p_j only if:

1) p_i keeps the initial part of the video stream it receives in its buffer so that it can provide p_j with the entire video stream in a pipelining manner,
2) p_i has enough forwarding bandwidth to deliver the video stream,
3) the video stream p_i is able to provide is required by p_j.

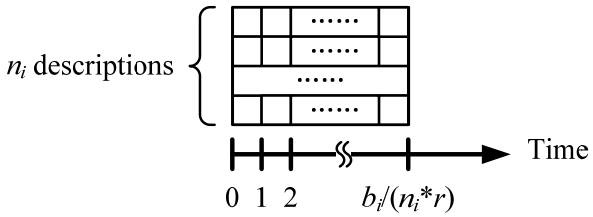
In this section, we describe proposed Dynamic Buffering scheme and show how it assists peers to retain the initial part of the received video stream, thereby prolonging the expiration time of the peer's service capability. Moreover, the Dynamic Buffering scheme attempts to maintain the diversity of descriptions of a video cached among peers to ensure fair provision within the system.

3.1 Basic Idea

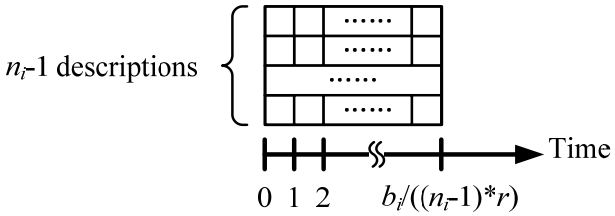
Despite the assumptions made here, the Dynamic Buffering scheme presented in this paper can be easily adapted to situations where more than one video is provided and encoded into multiple descriptions of different bit rates. For simplicity sake, however, let us assume that there is only one video provided for users to select and watch. The video is encoded into descriptions m of equal bit rate r, denoted as d_1, d_2, ..., d_m. Similar to conventional cache-and-relay schemes, the Dynamic Buffering scheme requires each peer to cache all the descriptions it receives until its buffer is filled up. Once the buffer of a peer is saturated with the streaming data, the peer must stop caching one of the descriptions it receives and discard the cached streaming data of the selected description to make room for subsequent streaming data of other descriptions. To avoid disrupting the video playback of a peer's children, a peer may stop caching and discard only those descriptions that have not been forwarded to any peers. Each peer is required to repeat the above process every time its buffer is filled up, and until there are no descriptions left that the peer may discard, or until only one description is left in its buffer.

Fig. 2 illustrates a general example of the Dynamic Buffering scheme. In **Fig. 2**, a peer p_i arrives at time t_i and receives descriptions n_i from other peers and/or the server. Suppose p_i is equipped with a buffer of size b_i. The buffer fills up with streaming data of descriptions n_i at time $t_i + b_i/(n_i * r)$, as shown in **Fig. 2(a)**. Instead of losing the initial parts of all the n_i descriptions as in the conventional cache-and-relay scheme, p_i chooses to discard one

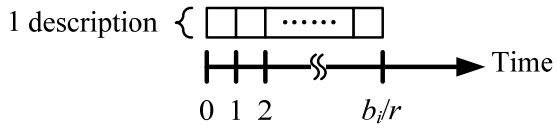
description to preserve the integrity of the other (n_i - 1) descriptions. Note that discarding one description does not degrade p_i's viewing quality. p_i still enjoys n_i-description viewing quality since it merely chooses not to cache one of the n_i descriptions it receives after playback. As shown in **Fig. 2(b)**, after sacrificing one description, p_i is able to hold onto the initial parts of the other (n_i - 1) descriptions until time $t_i + b_i/(n_i * r) + (b_i/(n_i * r)) * r/((n_i - 1) * r) = t_i + b_i/((n_i - 1) * r)$, where $(b_i/(n_i * r)) * r)$ indicates the room made by discarding one description and $(b_i/(n_i * r)) * r/((n_i - 1) * r)$ indicates the additional time period for that the other (n_i - 1) descriptions can be preserved. If no peer arrives to request descriptions from p_i, only one description will be left in p_i's buffer. In this case, p_i will not be able to preserve the initial part of the description that it has received and will thus lose its service capability after time $t_i + b_i / r$, as shown in **Fig. 2(c)**.



(a) p_i's buffer map at time $t_i + b_i/(n_i * r)$that indicates that descriptions and blocks p_i has cached.



(b) p_i's buffer map at time $t_i + b_i/((n_i - 1) * r)$.



(c) p_i's buffer map at time $t_i + b_i / r$.

Fig. 2. An illustration of the dynamic buffering scheme.

From **Fig. 2(a)** to **2(c)**, it can be observed that the Dynamic Buffering scheme allows a peer to degrade it's service capacity gradually, rather than completely lose its service capability after its buffer is filled up with descriptions. More specifically, the conventional cache-and-relay scheme prohibits p_i from supplying any description in its entirety to subsequent peers after time $t_i + b_i/(n_i * r)$, whereas the Dynamic Buffering scheme enables p_i to supply (n_i - k) descriptions in time period $[t_i + b_i/((n_i - k + 1) * r), t_i + b_i/((n_i - k) * r)]$, where $1 \leqq k \leqq n_i - 1$. Since the expiration time of p_i's service capability is largely extended from $t_i + b_i/(n_i * r)$ to $t_i + b_i / r$, p_i has a better chance to fully utilize its forwarding bandwidth, thereby alleviating the workload of the streaming server. The Dynamic Buffering scheme is expected to outperform conventional cache-and-relay schemes, especially when peers are equipped

with a small buffers or the inter-arrival time is great.

3.2 Description Distribution Balancing

So far, we have shown how a Dynamic Buffering scheme can enable peers to manage and preserve the descriptions they receive, in an attempt to maintain service capability. However, the service capability of peers is determined by providing complete descriptions and the descriptions it can provide. If the descriptions that a peer can provide are widely available in the system, that is, if the peer holds the same descriptions as many other peers, it may be unable to serve subsequent peers and alleviate the workload of the server, since it cannot provide newcomers with descriptions that they could not get from other peers. Such a skewed distribution in the availability of various descriptions would diminish the cooperation level among peers in on-demand streaming approaches employing MDC, including the proposed Dynamic Buffering scheme.

In the paper, we focus our efforts on achieving a balanced description distribution among peers and incorporating such considerations into the design of the Dynamic Buffering scheme as to further improve its performance. This goal can be achieved by focusing on either of two aspects: 1) description dropping and 2) parent and description selection. This study shows how arbitrary description dropping, as well as unadvised parent and description selection, may skew distribution in the availability of various descriptions, and describes how the Dynamic Buffering scheme alleviates those effects.

With the Dynamic Buffering scheme, a peer has to discard one description to preserve the others when its buffer is filled up. In the worst case, arbitrarily choosing one description to discard may lead peers to preserve the same descriptions and thus limit the number of descriptions held. If this were the case, newly arrived peers, especially the ones requesting high viewing quality, would barely be able to acquire all the descriptions they needed from existing peers. In order to achieve the desired viewing quality, they would have to rely on the server to provide the descriptions that were absent from existing peers. As a result, the resources of peers could not be fully and efficiently utilized to alleviate the workload of the server, despite of the fact that many peers preserved some descriptions for the provision of streaming services. Even worse, arbitrary description dropping may cause certain descriptions to entirely vanish from peers. In P2P on-demand streaming systems, there often exist peers whose service capability is about to expire. These peers still preserve some descriptions in the buffer but cannot continue to keep any of them unabridged once the buffer is filled up, because there is only one description left, or there are no kept descriptions. Arbitrary description dropping may lead to some peers controlling the entire supply of some descriptions. If this were the case, the descriptions kept by those peers would vanish once the buffers were filled up, and the server would be the only source for recovering the lost descriptions. To avoid such widespread and simultaneous disappearance of descriptions, informed description dropping is necessary.

As with informed description dropping, judicious parent and description selection of new peers is equally crucial in achieving balanced description distribution and improving peer resource utilization. If new peers were allowed to request descriptions from arbitrarily chosen parents, they might request descriptions that would be widely available for a long duration. Requests from new peers for widely available descriptions would shrink the diversity of descriptions among peers, leading to a skewed description distribution as occurred from unadvised description dropping. Selecting parents with a long expiration time does not favor the efficient utilization of peer resources. The detrimental effects described above could be avoided through judicious parent and description selection.

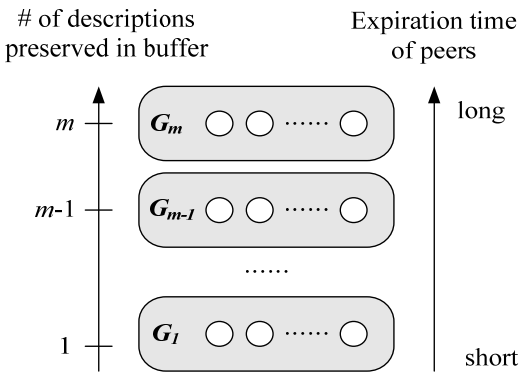


Fig. 3.The hierarchical organization formed by classifying peers into groups according to the number of descriptions preserved in the buffer.

Ideally, we would like peers to perform the description dropping procedure in such a way that peers whose service capability would expire at near or the same time could preserve as many different descriptions as possible, and the availability of the descriptions they preserve would be as widely available as possible. With respect to the parent and description selection, a new peer should first consider the peers whose service capability would expire the soonest as well as those holding the least widely available descriptions. In order to do this, we need a method to evaluate the expiration time of a peer's service capability and the availability of descriptions. The expiration time of a peer's service capability is defined as, "the time after that the peer will no longer be able to keep any description it has received." The time remaining before a peer's service capability expires can be roughly estimated by the number of descriptions it preserves in its buffer. A greater number of descriptions preserved in the buffer usually implies that more descriptions can be sacrificed and discarded if necessary, thus signifying a longer service capability. Accordingly, in the Dynamic Buffering scheme, the streaming server distinguishes between the expiration times of different peers by means of the number of descriptions they hold. The streaming server classifies peers into groups according to the number of descriptions preserved in the buffers. For example, all the peers preserving i descriptions in the buffer will form a group, denoted as G_i. **Fig. 3** shows the hierarchical organization formed by classifying peers into groups according to the number of descriptions they preserve in the buffers. The service capability of peers in the same group tends to expire at the same or similar time. The server should attempt to select peers in G_1 as the parents of new peers since they would soon lose the service capability. Within each group G_i, peers should keep the descriptions as widely variable, and as equally available as possible. The availability of a description, d_j, in a group G_i is defined as,

$$A_i(d_j) = \sum_{p_k : p_k \in G_i, d_j \in H(p_k)} F(p_k),$$

where $H(p_k)$ denotes the set of descriptions that peer p_k keeps in its buffer and $F(p_k)$ denotes the amount of available forwarding bandwidth at p_k, measured by the number of descriptions it can provide. In other words, the availability of a description in G_i indicates the maximum potential supply of the description that peers in G_i can collectively offer.

To construct the aforementioned hierarchical organization and evaluate the availability of various descriptions in each group, the server needs to collect necessary information from

peers, including the available forwarding bandwidth of each peer and the descriptions preserved in the buffer. Peers generally trigger any updates of that information. It is noteworthy that only those peers that have available forwarding bandwidth and still preserve at least one description in the buffer are involved in the hierarchical organization.

Description Dropping. By maintaining a hierarchical organization as shown in **Fig. 3**, the streaming server could force peers to perform the description dropping procedure. Because we would like peers to perform the description dropping procedure in such a way that the descriptions within each group are equally available, a peer p_i in group G_j would be required to drop the cached description with the most availability in G_{j-1}. The availability of each description preserved by p_i could be acquired by issuing a lookup query to the streaming server. **Fig. 4** shows the description-dropping algorithm of the Dynamic Buffering scheme.

---

p_i: the peer about to perform the description dropping procedure

$H(p_i)$: the set of description p_i preserves in its buffer

$T(p_i)$: the set of description p_i preservesand has forwarded to other peers

G_j: the group p_i belongs to

Description Dropping (p_i)

1 **for** each description $d_k \in (H(p_i) - T(p_i))$

2 $\quad A_{j-i}(d_k) \leftarrow \sum_{p_x : p_x \in G_{j-1}, d_k \in H(p_x)} F(p_x)$

3 find a description d_k,

$\quad$ s.t. $d_k = \arg\max_{d_n \in (H(p_i) - T(p_i))} A_{j-i}(d_n)$

4 $H(p_i) \leftarrow H(p_i) - d_k$

5 $G_j \leftarrow G_j - p_i$

6 **if** $j > 1$

7 $\quad G_{j-1} \leftarrow G_{j-1} \cup p_i$

Fig. 4. The description dropping algorithm of the dynamic buffering scheme.

Parent and Description Selection. To watch the desired video, a peer, p_i, issues a request to the streaming server, and specifies its available forwarding bandwidth and requested viewing quality, denoted as $F(p_i)$ and $Q(p_i)$. Since the provided video is assumed to be encoded into descriptions of equal bit rate r, $F(p_i)$ and $Q(p_i)$ can be measured by the number of descriptions. $F(p_i)$ indicates how many descriptions p_i can be forwarded simultaneously, and $Q(p_i)$ indicates how many descriptions p_i are required to achieve its specified viewing quality. On reception of a request from p_i, the streaming server has to determine which $Q(p_i)$ descriptions p_i it should receive from which peers. As mentioned before, in selecting parents for p_i, the streaming server should first consider those peers whose service capability will soon expire to improve peer resource utilization. In determining which descriptions p_i should receive from the selected parents, the server should give the highest priority to the description with the least availability in the group that the selected parents belong to, in order to enhance

the diversity of descriptions in the system. **Fig. 5** shows the algorithm for parent and description selection.

---

p_i: a newly arrived peer
t_i: the arrival time of p_i
$R(p_i)$: the set of descriptions p_i receives
$H(p_i)$: the set of descriptions p_i preserves
$Q(p_i)$: the viewing quality required by p_i
$F(S)$: the available forwarding bandwidth of the server

Parent and Description Selection (p_i)

1 $j \leftarrow 1$
2 $R(p_i) \leftarrow 0$
3 $H(p_i) \leftarrow 0$
4 **repeat**
5 find a peer $p_k \in G_j$,
s.t. $p_k = \underset{p_x:(H(p_z)-R(p_i))\neq 0}{\arg\min} \left(t_x + \frac{b_x}{|H(p_x)| * r} \right)$
6 **if** $p_k = NULL$
7 $j \leftarrow j+1$
8 go to step 4
9 find a description $d_x \in (H(p_k) - R(p_i))$
10 s.t. $d_x = \underset{d_y:d_y\in(H(p_k)-R(p_i))}{\arg\min} A_j(d_y)$
11 $F(p_k) \leftarrow F(p_k) - 1$
11 **if** $F(p_k) = 0$
12 $G_j \leftarrow G_j - p_k$
13 $R(p_i) \leftarrow R(p_i) \cup d_x$
14 $H(p_i) \leftarrow H(p_i) \cup d_x$
15 $Q(p_i) \leftarrow Q(p_i) - 1$
16 **until** $Q(p_i) = 0$ **or** $j > m$
17 **if** $Q(p_i) > 0$
18 **if** $F(S) \geq Q(p_i)$
19 make the server randomly select and forward $Q(p_i)$ descriptions that p_i lacks
20 **else**
21 reject p_i

Fig. 5. The parent and description selection algorithm of the dynamic buffering scheme.

4. Simulation Results and Analysis

4.1 Simulation Setting

In this section, we evaluate the performance of the Dynamic Buffering scheme through simulation experiments. The experiment environment is described as follows. The server is

capable of concurrently forwarding 5000 descriptions. Only one video of length 120 minutes is provided, and it is encoded into 16 descriptions. The buffer size of peers follows the uniform distribution $N(2 * r, 32 * r)$. The arrival of peer requests follows the Poisson distribution with a mean of λ, where λ ranges from 0 to 100 (requests/min.). The video quality required by a peer, in terms of the number of descriptions, follows the normal distribution $N(9, 3)$. Additionally, a peer's forwarding capacity is set to be equal to its downloading capacity. In other words, a peer is capable of forwarding as many descriptions as it receives. The parameters used for the experiment are summarized in **Table 1**.

In the following experiments, we evaluate the performance of the proposed Dynamic Buffering scheme (called Dynamic Buffering-Balance hereafter) by comparing it with CoopNet and its two variations: Dynamic Buffering-Sequence and Dynamic Buffering-Random. The main differences among the four approaches are summarized as follows.

- **CoopNet.** CoopNet employs the conventional cache-and-relay scheme. When the buffer of a peer is full, the peer will lose the initial parts of all the descriptions it receives and thus, will no longer be able to serve any subsequent peers.
- **Dynamic Buffering-Balance.** A peer performs the Dynamic Buffering scheme to prolong the expiration time of its service capability once its buffer is full.
- **Dynamic Buffering-Sequence.** The difference between this approach and Dynamic Buffering-Balance lies in the description dropping procedure. In this approach, a peer discards descriptions in sequence, instead of based on the description availability.
- **Dynamic Buffering-Random.** When the buffer is filled up, a peer randomly chooses one description to discard from all those held.

Table 1. Parameters used for the experiment

Parameter	value
Video length (min)	120
Server bandwidth (descriptions)	5000
Number of descriptions	16
Arrival rate (requests/min)	0-100
Buffer size	2-32

4.2 Performance Evaluation

This study investigated the server bandwidth consumption of the four approaches at various arrival rates. The results are shown in **Fig. 6**. It may be observed that the server bandwidth utilization of CoopNet sharply increased with the arrival rate and all the server bandwidth was used up at the arrival rate of 70 (requests/min.), whereas all the other three approaches employing the dynamic buffering scheme consumed much less server bandwidth. For example, at the arrival rate of 10 (requests/min.), the server bandwidth consumption in CoopNet and in Dynamic Buffering-Balance was 24.28% and 1.72%. As the arrival rate increased to 100 (requests/min.), the server bandwidth consumption in CoopNet and in Dynamic Buffering-Balance were 100% and 3.76%. This proves the superior scalability of the dynamic buffering scheme over the conventional cache-and-relay scheme. Furthermore, whether a peer should choose a description to discard based on the description availability seems trivial in this experiment.

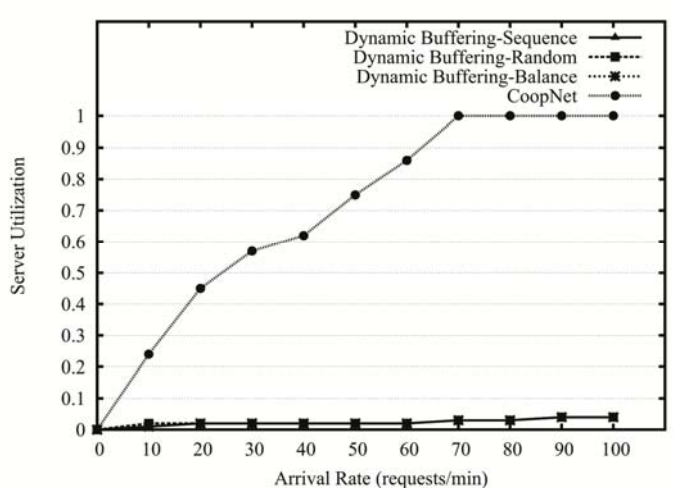


Fig. 6. With buffer size of 2-32 minutes, the server bandwidth utilization of CoopNet, dynamic buffering-sequence, dynamic buffering-random, and dynamic buffering-balance at various arrival rates.

To clarify the difference between Dynamic Buffering-Balance, Dynamic Buffering-Sequence, and Dynamic Buffering-Random, we restricted the maximum buffer size of a peer to 10 minutes, and evaluated the server bandwidth utilization of the three approaches at various arrival rates as shown in **Fig. 7**. In CoopNet, since the buffer size of a peer was grossly insufficient, the requests of new peers were usually served by the server. Thus, at an arrival rate of 20 (requests/min.), the server in CoopNet was already overwhelmed and unable to provide new peers with video services. On the other hand, with Dynamic Buffering, a peer could execute the dynamic adjustment process to increase its availability to keep the initial part of the video within a limited buffer. The smaller the buffer size, the more frequently the peer adjusted its buffer. Dynamic Buffering-Sequence and Dynamic Buffering-Random did not take into account the availability of descriptions in description dropping procedure, and consequently led to a skewed distribution in the availability of various descriptions. When a new peer requested the provided video, it was unable to acquire a diverse variety of descriptions from candidate parent peers. In Dynamic Buffering-Random, because the dropped description had been selected randomly, the server bandwidth consumption was between Dynamic Buffering-Balance and Dynamic Buffering-Sequence.

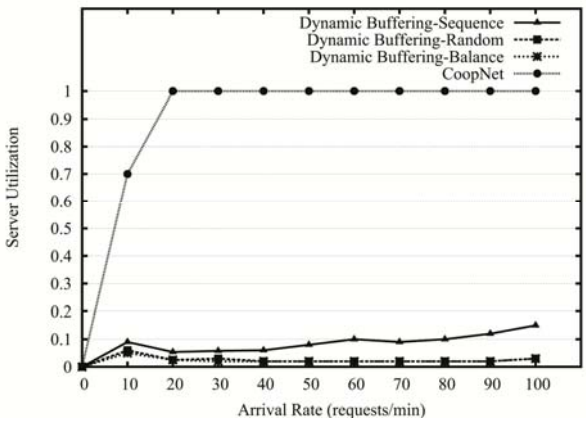


Fig. 7. With a buffer size of 2-10 minutes, the server bandwidth utilization of CoopNet, dynamic buffering-sequence, dynamic buffering-random, and dynamic buffering-balance at various arrival rates.

At an arrival rate of 10 (requests/min.), the server bandwidth utilization in CoopNet, Dynamic Buffering-Sequence, Dynamic Buffering-Random, and Dynamic Buffering-Balance were 69.26%, 8.32%, 6.48%, and 5.38%. In addition, when the arrival rate increased to 100 (requests/min.), the server bandwidth utilization in CoopNet, Dynamic Buffering-Sequence, Dynamic Buffering-Random, and Dynamic Buffering-Balance were 100%, 14.38%, 3.84%, and 3.8%. This result showed the superior scalability of Dynamic Buffering-Balance over the other approaches. The overall scalability of Dynamic Buffering-Balance was greater than CoopNet. In the other words, executing dynamic adjustment processes for peers may be of crucial importance for increasing the availability and peer resource utilization. On the other hand, it could be observed that Dynamic Buffering-Balance outperformed Dynamic Buffering-Sequence and Dynamic Buffering-Random. The results indicated that incorporating the resource balancing strategy into peers' buffer management was beneficial for attaining maximum description utilization. Dynamic Buffering effectively improved the description allocation to keep resources in balance. It could strain to meet the various requests of video quality for new peers even if the capacity of buffer and arrival rate were reduced. This result was caused primarily by the interworking of MDC technique and Dynamic Buffering. For example, if a peer possessed 16 descriptions and the buffer had the storage capacity for caching 32-minute initial part of the video. In the case of CoopNet, the peer could only keep the first 2-minute part of the video in its buffer. At the third minute, the peer would not have sufficient storage to cache the initial part of the video. The peer who did not execute dynamic adjustment would make the peer incapable of utilizing the buffer's storage space and providing the service for a new peer. However, in Dynamic Buffering, peers are allowed to drop descriptions through a dynamic adjustment process and reduce the number of descriptions in the buffer, thereby releasing redundant buffer storage space to prolong the time required to keep the initial part of the video in the buffer. Dynamic Buffering gradually adjusted the number of descriptions to prevent the buffer storage space from losing all the descriptions at one time. When the arrival rate was increased, it was still able to alleviate the server loading by executing dynamic adjustment and resource balancing significantly. In CoopNet, since the buffer size of a peer was insufficient to wait for the arrival of new peers, the requests of new peers would be provided by the server. Thus, when the arrival rate was 20(requests/min), the server in CoopNet had already been overwhelmed, and were unable to provide new peers with video services. On the other hand, in Dynamic Buffering, in order to keep the initial part of the video in a limited buffer space, a peer could execute a dynamic adjustment process to increase its availability.

Fig. 8 shows the server bandwidth utilization of the four approaches when the arrival rate was fixed to 10 (requests/min.), and the buffer size of peers varied between 7 and 32 minutes. As shown in **Fig. 8**, while buffer was long, the waiting time a peer could afford increased accordingly. In Dynamic Buffering, the length of the initial part of the video could be lengthened so that the probability of serving a new peer by other peers would increase. Taking advantage of peer service capacity may reduce the requests of a new peer to the server. **Fig. 9** is a detailed illustration of the three description-dropping approaches shown in **Fig. 8**. It illustrates the diversity of Dynamic Buffering-Sequence, Dynamic Buffering-Random, and Dynamic Buffering-Balance. When the buffer size was restricted, the Dynamic Buffering was able to reduce the bandwidth consumption of the server. In **Fig. 8** and **Fig. 9** when the maximum length of the buffer was 7 minutes, the server bandwidth consumptions of CoopNet, Dynamic Buffering-Sequence, Dynamic Buffering-Random, and Dynamic Buffering-Balance were 75.52%, 15.02%, 10.56%, and 9.94%. Furthermore, when the maximum length of the buffer was 32 minutes, the server bandwidth consumptions of CoopNet, Dynamic

Buffering-Sequence, Dynamic Buffering-Random, and Dynamic Buffering-Balance were 26.64%, 2.8%, 1.52%, and 1.46%. The results show that the buffer size was highly pertinent to the service capability of a peer and the scalability of the system. The larger the buffer storage space, the more robust the dynamic adjustment would be. Applying Dynamic Buffering succeeded in prolonging the availability of peers.

Dynamic Buffering was more adaptive to distinct capacity of peers than CoopNet. Even though the storage space of the buffer was insufficient to cache video content, Dynamic Buffering was still able to alleviate the server loading effectively. It had higher system scalability and robustness in different environments, and alleviated the influence caused by the limited capacity of a peer on the server.

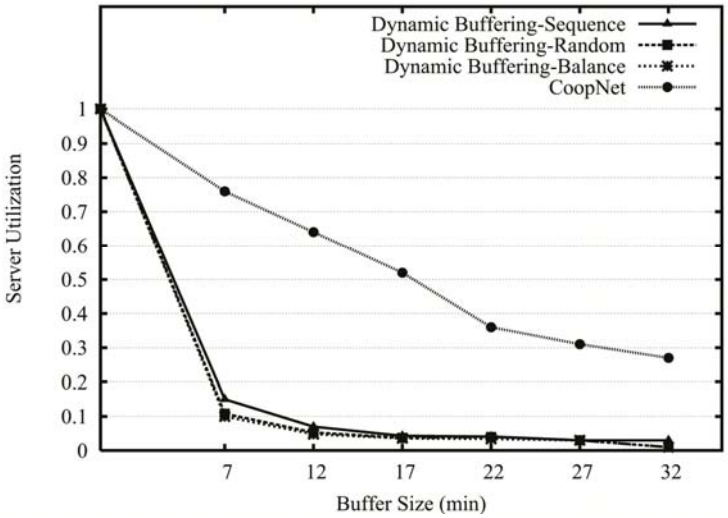


Fig. 8. With the arrival rate of 10(requests/min), the server bandwidth utilization of CoopNet, dynamic buffering-sequence, dynamic buffering-random, and dynamic buffering-balance at various buffer sizes.

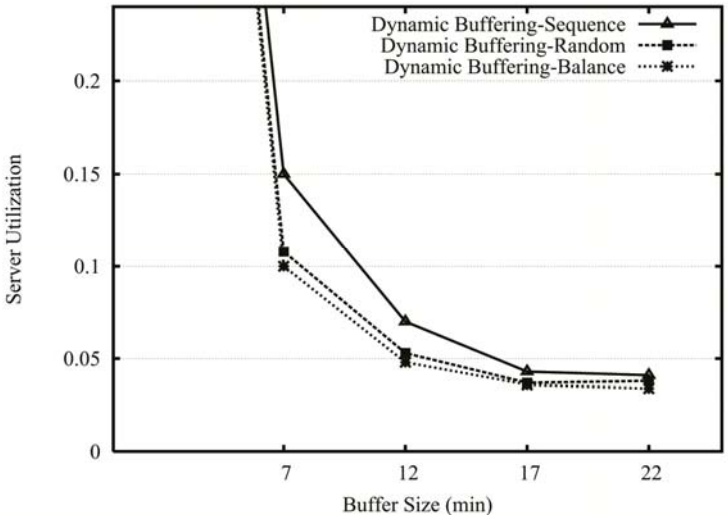


Fig. 9. The server bandwidth utilization of dynamic buffering-sequence, dynamic buffering-random, and dynamic buffering-balance.

5. Conclusions

In this paper, we proposed a novel caching scheme for P2P on-demand media streaming, called *Dynamic Buffering*. With the proposed approach, a peer would be able to prolong the existence of the initial part of the video by dynamically adjusting cached descriptions. The dynamic adjustment process was designed to increase the service capacity of a peer and alleviate the bandwidth consumption of a server at various request arrival rates. The Dynamic Buffering was suitable for videos with various degrees of popularity. When peers require an extremely popular video with high arrival rate, or likewise an unpopular movie with low arrival rate, a peer could implement dynamic adjustment process to extend the available time of the initial part of the video. Applying the Dynamic Buffering technique in a P2P VoD system could prolong the available time of peers to wait for the arrival of new peers, and hence improve the scalability of a server.

The main restriction of this study is that Dynamic Buffering was designed for P2P VoD system without considering the departure of peers. In an on-demand P2P streaming system, peers are able to cooperate with each other to acquire video content. When a peer departs from the system, the video quality of other peers may be affected. Hence, the streaming server must deal with unexpected peer departures caused by the interrupted transmission. As mentioned earlier, the issue of failure recovery will be investigated in our future publication. In addition, we are currently working on the implementation of P2P MDC streaming system with dynamic buffering based on the H.264/AVC coding standard, and these details will also be reported in our future publications.

References

[1] S. Androutsellis-Theotokis and D. Spinellis, “A survey of peer-to-peer content distribution technologies,” *ACM Computer Survey*, vol. 36, no. 4, pp.335–371, 2004.

[2] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, “Chord: a scalable peer-to-peer lookup protocol for Internet applications,” *IEEE/ACM Trans. on Networking,* vo.11, no.1, pp.17-32, 2003.

[3] A. Rowstron and P. Druschel, “Pastry: Scalable, distributed object location and routing for large scale peer to peer system,” in *Proc. of ACM Distributed Systems Platforms（Middleware）*, pp.329-350, 2001.

[4] M. Ripeanu, “Peer-to-Peer architecture case study: Gnutella network,” in *Proc. of IEEE Peer-to-Peer Computing*, pp.99-100, 2001.

[5] R. Schollmeier, “A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications,” in *Proc. of First International Conf. on Peer-to-Peer Computing*, pp.101–102, 2004.

[6] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, “Splitstream: high-bandwidth multicast in cooperative environments,” in *Proc. of ACM Conf. on SOSP*, pp.298–313, 2003.

[7] Y. Cui and K. Nahrstedt, “Layered peer-to-peer streaming,” in *Proc. of ACM Conf. on NOSSDAV*, pp.162–171, 2003.

[8] 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 Conf. on Communications*, pp.1467–1472, 2004.

[9] Y. Guo, K. Suh, J. Kurose, and D. Towsley, “P2cast: peer-to-peer patching scheme for vod service,” in *Proc. of ACM Conf. on WWW*, pp.301–309, 2003.

[10] E. Kusmierek, Y. Dong, and D. H. Du, “Loopback: exploiting collaborative caches for large-scale streaming,” *IEEE Transactions on Multimedia*, vo.8, no.2, pp.233–242, 2006.

[11] S. Jin and A. Bestavros, “Cache and relay streaming media delivery for asynchronous clients,” In

Proc. of the 4th International Workshop on Networked Group Communication, 2002.
[12] Y. Cui, B. Li, and K. Nahrstedt, "ostream: asynchronous streaming multicast in application-layer overlay networks," *IEEE Journal on Selected Areas in Communications*, vo.22, no.1, pp.91–106, 2004.
[13] A. Dan, D. Sitaram, and P. Shahabuddin, "Batching: Scheduling policies for an on-demand video server with batching," in *Proc. of ACM Multimedia*, pp.15-23, 1994.
[14] K. A. Hua, Y. Cai, and S. Sheu, "Patching: a multicast technique for true video-on-demand services," in *Proc. of ACM Multimedia*, pp.191-200, 1998.
[15] A. Hu, " Video-on-demand broadcasting protocols: A comprehensive study, " in *Proc. of IEEE INFOCOM*, pp.508-517, 2001.
[16] V. N. Padmanabhan, H. J. Wang, P. A. Chou, and K. Sripanidkulchai, "Distributing streaming media content using cooperative networking," in *Proc. of ACM Conf. on NOSSDAV*, pp.177–186, 2002.
[17] V. N. Padmanabhan and K. Sripanidkulchai "The case for cooperative networking," *Lecture Notes in Computer Science*, vo.2429/2002, pp.178-190, 2002.
[18] C.-S. Lin "Enhancing P2P live streaming performance by balancing description distribution and available forwarding bandwidth," *International Journal of Communication Systems*, to be published.
[19] A. J. Ganesh, A.-M. Kermarrec, and L. Massoulié, "Peer-to-Peer membership management for gossip-based protocols," *IEEE Trans. on Computers*, vo.52, no.2, pp.139-149, 2003.
[20] V. K. Goyal, "Multiple description coding: compression meets the network," *IEEE Signal Processing Magazine*, vo.18, no.5, pp.74–93, 2001.
[21] Z. Lu and W. A. Pearlman, "An efficient, low-complexity audio coder delivering multiple levels of quality for interactive application," in *Proc. Workshop on Multimedia Signal Processing*, pp.529–534, 1998.
[22] A. Albanese, J. Blomer, J. Edmonds, M. Luby, and M. Sudan, "Priority encoding transmission," *IEEE Trans. on Information Theory*, vo.42, no.6, pp.1737–1744, 1996.
[23] Y. Wang, A. Reibman, and S. Lin, "Multiple description coding for video delivery," in *Proc. of the IEEE*, vo.93, no.1, pp.57–70, 2005.
[24] X. Tan and S. Datta, "Building multicast trees for multimedia streaming in heterogeneous P2P networks," in *Proc. of Systems Communications*, pp.141–146, 2005.
[25] R. Bernardini, M. Durigon, R. Rinaldo, L. Celetto, and A. Vitali, "Polyphase spatial subsampling multiple description coding of video streams with H264," in *Proc. of IEEE International Conf. on Image Processing*, vo.5, pp.3213-3216, 2004.
[26] O. Campana and S. Milani, "A multiple description coding scheme for the H.264/AVC coder," in *Proc. of the International Conf. on Telecommunication and Computer Networks*, pp.191-195, 2004.
[27] J. Apostolopoulos, "Reliable video communication over lossy packet networks using multiple state encoding and path diversity," in *Proc. of Visual Communications: Image Processing*, pp.392-409, 2001.
[28] N. Zandon`a, S. Milani, and A. De Giusti, "Motion-compensated multiple description video coding for the H.264/AVC standard," in *Proc. of IADAT International Conf. on Multimedia, Image Processing and Computer Vision*, pp.290-294, 2005.

Chow-Sing Lin received the Ph.D. degree in Computer Engineering from the University of Central Florida, Florida, USA, in 2000. He is currently an Associate Professor in the Department of Computer Science and Information Engineering, National University of Tainan, Taiwan. His research interests include wired/wireless media delivery, peer-to-peer media streaming, distributed multimedia systems, and sensor networks.