LETTER

Cooperative Recording to Increase Storage Efficiency in Networked Home Appliances

Eunsam KIM†, Jinsung KIM††, Nonmembers, and Hyoseop SHIN††††, Member

SUMMARY This paper presents a novel cooperative recording scheme in networked PVRs based on P2P networks to increase storage efficiency compared with when PVRs operate independently of each other, while maintaining program availability to a similar degree. We employ an erasure coding technique to guarantee data availability of recorded programs in P2P networks. We determine the data redundancy degree of recorded programs so that the system can support all the concurrent streaming requests for them and maintain as much availability as needed. We also present how to assign recording tasks to PVRs and playback the recorded programs without performance degradation. We show that our proposed scheme improves the storage efficiency significantly, compared with when PVRs do not cooperate with each other, while keeping the playbackability of each request similarly.

key words: networked PVRs, cooperative recording, storage efficiency, P2P streaming

1. Introduction

With the advances in networks and computer technologies, home appliances connected to the Internet have become commonplace. In particular, digital TVs including PVRs (Personal Video Recorders) can store broadcast TV programs in their built-in storage devices [1], [2]. Thus, users can watch TV programs they want at any time after having recorded them during their broadcasting times. That is, they can record their favorite programs they may miss for personal reasons or because of events with time differences such as World Cup soccer games.

However, it can be seen that, when more than one PVRs record a particular program in a situation where they operate independently of each other, each of them must store the program on its own storage devices even though the program has the same video data. This implies that most participating PVRs can waste their storage spaces because a much smaller number of PVRs with much less storage spaces can process all playback requests for the program since the requests are spread out over time until the program is deleted from the system. It is obvious that, the more popular program the PVRs record, the greater the waste of storage spaces is. Thus, if PVRs cooperate with each other to record and playback a particular program after contributing their storage spaces to the system, it is not necessary for all of them to store the same program.

To increase storage efficiency, we therefore propose a novel cooperative recording scheme in networked PVRs connected with each other. We construct the system based on P2P networks to reduce the cost of operating central servers by taking advantage of their high scalability and low cost. Since PVRs tend to join and leave frequently, we employ an erasure coding technique to ensure data availability of recorded programs by storing data redundantly. We thus determine the data redundancy degree of recorded programs so that the system can support all the concurrent streaming requests for them as well as maintain necessary data availability. We then describe how to assign recording tasks to PVRs and playback the recorded programs without performance degradation.

Through extensive simulations, we show that our proposed scheme improves the storage efficiency significantly compared with when PVRs do not cooperate while keeping similar playbackability of each request.

The present paper is organized as follows: Sect. 2 describes work related to this paper. Section 3 describes our proposed cooperative recording scheme in detail. Section 4 presents the experimental results of the cooperative recording scheme in comparison with a standalone recording scheme. Finally, Sect. 5 offers conclusions.

2. Related Work

Overcoming the limited storage space problem of standalone PVRs, networked PVRs connected to broadband have enabled users to watch TV programs they want from a network. In particular, recent researches on PVR services based on P2P sharing technologies have been conducted to reduce service-dedicated bandwidth by receiving data from neighbor PVRs instead of central servers [3]–[5]. Some of them have developed PVR systems suitable for mobiles [6] or for vehicles [7]. However, these studies have focused mainly on suggesting structures suitable for PVRs to share data among them. No studies have considered the storage space efficiency of participating PVRs when they perform the recording of broadcast programs.

On the other hand, erasure coding techniques increase data availability in a network. They split one whole file into many data blocks, divide each block into $k$ fragments,
Fig. 1 The way to divide and encode each block of an original program into fragments through erasure coding in our system.

encode additional \((n - k)\) fragments, and distribute all of fragments to nodes. Even though some fragments are lost in the network for some reason, the original block can be reproduced through erasure coding as long as only \(k\) fragments among \(n\) fragments are received. Thus, these techniques have been widely employed to develop distributed storage systems in the network environment where packets are likely to be lost\[8\], \[9\]. However, existing distributed storage systems are focused on providing reliable storage space by increasing data availability. No systems have tried to support streaming services for TV programs whose blocks are divided and distributed to nodes through erasure coding.

### 3. Cooperative Recording of TV Programs

#### 3.1 A System Architecture for Cooperative Recording

The system architecture for cooperative recording proposed in this paper consists of PVRs, a shared storage pool contributed by PVRs, and a recording coordinator. When requested by the recording coordinator, PVRs store some fragments assigned to themselves among all the fragments of a program in its storage space reserved for cooperative recording. When a user wants to playback the recorded program, the corresponding PVR plays it back after receiving and decoding a certain number of fragments from several PVRs. The shared storage pool is a virtual storage space for cooperative recording. This space combines all the storage spaces donated by participating PVRs. To receive and transmit necessary fragments, PVRs form P2P networks based on a mesh-pull structure. The recording coordinator manages the storage and transmission of recorded programs among PVRs, maintaining all necessary information including connection status and available storage space information of each PVR, and storage location and size information of fragments of each recorded program.

3.2 Determining the Redundancy Degree of Recorded Programs

In our cooperative recording system, recorded programs are encoded with erasure coding, as shown in Fig. 1 where \(c\) represents the total number of blocks belonging to each program. The original video program is divided into fixed-sized blocks from \(B_1\) to \(B_c\), and each block is also divided into \(k\) fragments and \((n - k)\) fragments are then redundantly encoded. For example, \(B_1\) is divided into \(k\) original fragments from \(OF_{1}\) to \(OF_{k}\) and \((n - k)\) additionally encoded fragments from \(CF_{k+1}\) to \(CF_{n}\). The \(n\) fragments per each block are then distributed to PVRs to reproduce it later.

Our system should support video streaming for all concurrent program playback requests, with less storage space to achieve high storage efficiency, while maintaining similar data availability to when the requests are processed from local storage devices. Thus, we investigate the minimum redundancy degree of a program to serve the given number of concurrent playback requests without quality disruption, i.e., how many fragments per block in total are required at least. Let \(n_{(k,p,a)}\) denote the total number of fragments required to guarantee that program availability should be at least \(a\%\), given that each block is divided into \(k\) fragments and the availability of each PVR is \(p\). In fact, as the \(k\) value increases, more computation and data transmission overhead are required to reproduce each block. This is because more PVRs must be involved in receiving and combining fragments to obtain each block. Conversely, with the increased \(k\) value, load balancing among PVRs can improve in the situation where the system is serving many different programs because fragments are likely to be spread more evenly among PVRs. Thus, \(k\) value should be determined based on the trade-off between system overhead used for erasure coding and the degree of load balancing among PVRs depending on system environments.
In our system, the same number of PVRs as the total number of fragments is needed since each fragment is stored on a distinct PVR. This means that \( n_{k,p,o} \) PVRs on average participate in transmitting fragments for program playback. However, the total playback rate of concurrent requests at a particular time may exceed the aggregated upload capacity that \( n_{k,p,o} \) PVRs can provide, which may result in service disruption. To avoid this, the system must provide sufficient redundant fragments distributed to additional PVRs.

Let \( R_{\text{max}}, B_t, \) and \( B_\alpha \) represent the expected maximum number of concurrent playback requests before it is deleted from the system, the upload bandwidth each PVR contributed to the system for cooperative recording, and video playback rate of each program, respectively. We can obtain the effective upload bandwidth that each PVR with PVR availability of \( p \) can provide by calculating \( p \times B_\alpha \). We can then evaluate the average number of playback requests that one PVR with \( p \) can support by dividing \( p \times B_\alpha \) by \( B_t \). Thus, the number of PVRs required to support \( R_{\text{max}} \) requests can be calculated by dividing \( R_{\text{max}} \) by \( \frac{B_t}{B_\alpha} \). However, if \( n_{k,p,o} \) PVRs can provide the sufficient upload bandwidth to support \( R_{\text{max}} \) requests, more fragments than \( n_{k,p,o} \) are not needed. Therefore, we can obtain Eq. (1) that determines the total number of fragments, satisfying the above conditions.

\[
 n = \max \left\{ n_{k,p,o}, \left( \frac{R_{\text{max}} \times B_t}{p \times B_\alpha} \right) \times \alpha \right\} \tag{1}
\]

In the above equation, the \( \alpha \) value is used to reflect the number of different programs being served since others are evaluated based on one program. It can also be used to compensate for performance degradation factors, including the delays needed to switch the tasks between PVRs and caused by the load unbalancing among PVRs.

### 3.3 Allocating Recording Tasks to PVRs

Before any program scheduled to be recorded starts to broadcast, the recording coordinator should determine which PVRs will perform the recording tasks, including storing fragments on their storage spaces and carrying out erasure coding. Since the resources for recording are limited in the system, recording tasks should be efficiently assigned to PVRs. To do so, the recording coordinator thus selects \( n \) PVRs to store the same number of fragments produced per block through erasure coding in the order of the following priorities:

1. the ones that have reserved the recording of the program
2. the ones that are currently turned on
3. the ones that are currently turned off

To further increase the possibility of improving data availability of recorded programs during playback, it first selects the PVRs that have reserved the recording of the program because they should be turned on when playing it back. The recording coordinator can remotely turn on PVRs that are turned off using special network signals such as Wake on LAN (WOL). It then selects the PVRs already turned on because they consume less resources used to start recording than those turned off. Note that, if more than the required number of PVRs are selected in any of the above processes because some PVRs have the same priority, the PVRs with more remaining storage space reserved for recording have higher priorities.

The recording coordinator also need to select one PVR to receive the broadcast program, carry out erasure coding, and distribute encoded fragments to the selected PVRs. To reduce the delay caused by encoding computations, the recording coordinator chooses the PVR with the best performing CPU among the selected PVRs. When any PVR already assigned to the recording task is turned off, the recording coordinator repeats the selection process to find another PVR to take over the task of the PVR to be turned off.

### 3.4 Playing Back Recorded Programs

To playback a recorded program, a PVR should obtain its original data by reproducing each block through erasure coding after receiving at least \( k \) fragments among \( n \) fragments distributed to \( n \) PVRs. It is important for PVRs to playback recorded programs seamlessly, just like when they do from local storage devices. To do so, our system supports video streaming among PVRs over P2P networks based on a mesh-pull structure so that PVRs can receive blocks in the order they are required for playback.

As soon as a PVR requests to playback a recorded program, the recording coordinator informs it of a list of neighbor PVRs. The PVR then receives necessary fragments from neighbor PVRs simultaneously after requesting them. Even though the PVR’s neighbor leaves P2P networks, the PVR can still receive data from its remaining neighbors while finding another PVR.

### 4. Experimental Results

To demonstrate the effectiveness of our proposed cooperative recording scheme, we performed extensive simulations using the PeerSim P2P simulator. We set the download and upload bandwidth of each PVR to 10 and 5 Mbps, respectively [10]. Each program has a 4 Mbps playback rate with 1 GB size and is divided at a rate of 3 blocks per second.

To investigate the impact of availability of each PVR, i.e., \( p \), on the system performance, we varied it from 0.3 to 0.9, increasing it by 0.2. The inter-arrival and inter-leaving rates follow a Poisson distribution with a mean of 300 to 900 sec., depending on \( p \). We also set \( k \) to 8 so that load among PVRs can be quite widely distributed and we calculated \( n \) by Eq. (1). We predetermined \( n_{k,p,o} \) values from our experiments so that the availability of each program in the system can be maintained at over 95%. We obtained the following results: \( n_{8,0.3,95} = 41, n_{8,0.5,95} = 23, n_{8,0.7,95} = 16, \) and \( n_{8,0.9,95} = 8 \).
Comparison of required storage capacity between SRS and CRS

Fig. 2

Note that, to the best of our knowledge, the user watching time distributions for recorded programs have not been known in the literature. Thus, we assume that the maximum number of requests are made one hour after the broadcast time of the program and the number of requests to be processed simultaneously at the peak time, i.e., $R_{\text{max}}$, is 20% of the total number of requests. It then decreases by 5% per hour.

Figure 2 shows how much our cooperative recording scheme (CRS) improves storage efficiency compared with a standalone recording scheme (SRS) where each PVR independently store programs on its own storage device. That is, the storage space required to support all the playback requests without quality disruption in CRS is much smaller than that in SRS for all $p$ values while varying the number of recording requests from 50 to 125. That is, when the number of requests is 50, 75, 100, and 125, CRS on average requires 5.7, 3.9, 3.5, and 3.4% of SRS’s storage space, respectively. In particular, when the number of requests is 125 and $p$ is 0.9, CRS requires only 1.9% of SRS’s storage space. This considerable improvement is because CRS reduces the required redundancy degree of each program by providing the aggregated upload bandwidth only enough to support the maximum number of concurrent playback requests at a peak time.

Figure 3 demonstrates that, even though CRS requires much less storage space to store each recorded program, its playbackability in all $p$ values exceeds 97.5%. This is because it can provide sufficient PVRs to support the given number of concurrent playback requests, achieving high storage efficiency. We can also see that the playbackability increases slightly as the number of requests increases.

It is also noted that we measure the performance in terms of playbackability 2 hours after the peak time. This is because playbackability is over 99% after that time since $k$ is determined to support the number of requests at the peak time. The playbackability indicates the ratio of playing back time to the whole measuring time including waiting time for necessary fragments from other PVRs. Note that, even when some blocks have not arrived on time in our system, PVRs can receive them even if they wait for a while, unlike the conventional P2P streaming systems where some blocks may be eventually unable to be obtained due to lack of available PVRs. Thus, we employ the playbackability as a performance metric to show the playback quality and program availability simultaneously.

We set $\alpha$ to 1.0 because we carried out all the experiments for only one recorded program so that their results cannot interfere with those for other programs. We show the average values of all experimental results.

Figure 3 shows how much our cooperative recording scheme (CRS) improves storage efficiency compared with a standalone recording scheme (SRS) where each PVR independently store programs on its own storage device. That is, the storage space required to support all the playback requests without quality disruption in CRS is much smaller than that in SRS at all $p$ values while varying the number of recording requests from 50 to 125. That is, when the number of requests is 50, 75, 100, and 125, CRS on average requires 5.7, 3.9, 3.5, and 3.4% of SRS’s storage space, respectively. In particular, when the number of requests is 125 and $p$ is 0.9, CRS requires only 1.9% of SRS’s storage space. This considerable improvement is because CRS reduces the required redundancy degree of each program by providing the aggregated upload bandwidth only enough to support the maximum number of concurrent playback requests at a peak time.

Figure 3 demonstrates that, even though CRS requires much less storage space to store each recorded program, its playbackability in all $p$ values exceeds 97.5%. This is because it can provide sufficient PVRs to support the given number of concurrent playback requests, achieving high storage efficiency. We can also see that the playbackability increases slightly as the number of requests increases.

It is also noted that, deviating from a consistent trend in Fig. 3, CRS’s playbackability is on average 0.8% higher when the number of requests is 50 compared with when it is 75. This is only the case where the left term value of the max function in Eq. (1) is greater than right one. This is because more PVRs are provided than needed in this case since the number of participating PVRs in guaranteeing program availability of 95% when $k$ is 8, i.e., $R_{\text{crs},0.95}$, is greater than that required to support only 50 playback requests.

On the other hand, if we vary the $k$ value, we expect that it affects the performance by adjusting the degree of load-balancing among PVRs. As the $k$ value increases, the number of fragments belonging to each block also increases and the fragments are spread to more PVRs. Thus, the performance can improve further since the system is more likely to avoid the situation where only a few PVRs on which requests are concentrated have already used up their resources even though the resources of others are sufficiently available. However, the performance can be degraded slightly due to more computing and data exchange overheads required to reproduce each block.

It is also expected that the playback quality improves with the increased $\alpha$ value since the $n$ values should also...
increase to guarantee the higher program availability. On the contrary, the storage efficiency decreases since more fragments per block should be generated and stored among PVRs.

The experimental results therefore indicate that our proposed CRS considerably improves storage efficiency by storing only sufficient fragments to support the given concurrent streaming requests as well as maintains program availability at a similar degree to that in SRS.

5. Conclusions

In this paper, we proposed a cooperative recording scheme in networked PVRs based on P2P networks. It can increase storage efficiency considerably compared with when each PVR works alone. To this end, we employed an erasure coding technique to maintain similar program availability. We also determined the total number of fragments required to support all the concurrent streaming requests and presented the way to assign recording tasks to PVRs and playback the recorded programs seamlessly. The simulation results show that our proposed scheme performs much better than when PVRs do not cooperate with each other.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2019R1A2C1002221 & 2019R1F1A1063698) and 2019 Hongik University Research Fund.

References

[1] El H. Khabbiza, R. El Alami, and H. Qjidaa, “Peer-assisted content delivery to reduce the bandwidth of TSTV service in IPTV system,” International Journal of Digital Multimedia Broadcasting, vol.2019, Article ID 2495310, pp.1–12, 2019.
[2] P. Neumann, “Analyzing the effects of time-shifted TV on the energy consumption of personal video recorders,” Proc. IEEE Fourth International Conference on Consumer Electronics Berlin, pp.304–308, 2014.
[3] E. Kim and C. Lee, “An on-demand TV service architecture for networked home appliances,” IEEE Commun. Mag., vol.46, no.12, pp.56–63, 2008.
[4] S. Oh, J.-S. Kim, K.-S. Kong, and J. Lee, “Closed P2P system for PVR-based file sharing,” IEEE Trans. Consum. Electron., vol.51, no.3, pp.900–907, 2005.
[5] E. Kim, T. Kim, and C. Lee, “An adaptive buffering scheme for P2P live and time-shifted streaming,” Applied Sciences, vol.7, no.2, pp.1–14, 2017.
[6] J. Dai, K. Xia, Z. Hua, and Z. Kou, “Personal video recorder function based on Android platform set-top box,” Telkomnika Indonesian Journal of Electrical Engineering, vol.12, no.1, pp.550–557, 2014.
[7] Y. Kim and D. Shin, “Improving file system performance and reliability of car digital video recorders,” IEEE Trans. Consum. Electron., vol.61, no.2, pp.222–229, 2015.
[8] C. Aatish and M. Avinash, “Data management in erasure-coded distributed storage systems,” Proc. 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing, pp.902–907, 2020.
[9] Y. Qiao, X. Kong, M. Zhang, Y. Zhou, M. Xu, and J. Bi, “Towards in-network acceleration of erasure coding,” Proc. Symposium on SDN Research, pp.41–47, 2020.
[10] D. Wu, Y. Liu, and K. Ross, “Modeling and analysis of multichannel P2P live video systems,” IEEE ACM Trans. Netw., vol.18, no.4, pp.1248–1260, 2010.