Providing VCR in a Distributed Client Collaborative Multicast Video Delivery Scheme*

X.Y. Yang[1], P. Hernández[1], F. Cores[2], A. Ripoll[1], R. Suppi[1], and E. Luque[1]

[1] Computer Science Department, ETSE, Universitat Autònoma de Barcelona, 08193-Bellaterra, Barcelona, Spain
[2] Computer Science & Industrial Engineering Department, EPS, Universitat de Lleida, 25001, Lleida, Spain

Abstract. In order to design a high scalable video delivery technology for VoD systems, two representative solutions have been developed: multicast and P2P. Each of them has limitations when it has to implement VCR interactions to offer true-VoD services. With multicast delivery schemes, part of system resources has to be exclusively allocated in order to implement VCR operations, therefore the initial VoD system performance is considerably reduced. The P2P technology is able to decentralize the video delivery process among all the clients. However, P2P solutions are for video streaming systems in Internet and do not implement VCR interactivity. Therefore, P2P solutions are not suitable for true-VoD systems. In this paper, we propose the design of VCR mechanisms for a P2P multicast delivery scheme. The new mechanisms coordinate all the clients to implement the VCR operations using multicast communications. We compared our design with previous schemes and the results show that our approach is able to reduce the resource requirements by up to 16%.

1 Introduction

Recent advances in high-performance network and video codification technology have made it feasible for the Video on Demand (VoD) server to implement the interaction capabilities of a classic Video Cassette Recorder (VCR), offering the true-VoD service. However, VCR interactions, such as pause or fast forward, increase the resource requirements in the delivery process and reduce the VoD system performance.

Certain researchers have proposed delivery policies that take advantage of the multicast feature. A multicast scheme allows clients to share delivery channels and decrease the server and network resource requirements. Patching multicast policy [3], for example, dynamically assigns clients to join on-going multicast channels and patches the missing portion of video with a unicast channel. The disadvantage of a multicast solution is the complexity of implementing interactive operations, because there is not a dedicated channel per client.

* This work was supported by the MCyT-Spain under contract TIC 2004-03388.

W.E. Nagel et al. (Eds.): Euro-Par 2006, LNCS 4128, pp. 777–787, 2006.

In [2], the authors introduce techniques to implement jump operations. The techniques are able to join the client with another on-going multicast channel (B-Channels), after a jump operation. The B-Channel has to be delivering the desired new playback point of the client. Emergency channels (I-Channels) are used in case no such B-Channel exists. *Split and Merge* (SAM) in [7], is a protocol to merge I-Channels with B-Channels, reducing the cost of a VCR operation. In SAM, the merge process is performed by synchronized buffers that are statically allocated in the central access nodes. The centralized buffer management of SAM is not scalable (the VCR request blocking probability grows linearly in accordance with the VCR frequency). In [1], the authors decentralize buffer among clients to reducing the merging time and the VCR operation resource requirements. In [4], the authors analyze the optimal number of I-Channels that a multicast VoD system has to allocate in order to implement VCR operations. Despite the fact that multicast policies are able to offer true-VoD with VCR, part of the server resource is, exclusively, allocated for VCR interactions. Consequently, VCR operations considerably reduce the system global performance.

Most recently, the peer-to-peer (P2P) paradigm has been proposed to decentralize the delivery process to all clients. Delivery schemes like Chaining, DirectStream o P2Cast [6] are the most representatives. In these schemes, clients cache the most recently received video information in the buffer and forward it to the next clients using unicast channels. Other P2P architectures such as CoopNet and PROMISE[5] assumes that a single sender does not have enough outbound-bandwidth to send one video and use n senders to aggregate the necessary bandwidth. All previous P2P schemes use unicast communications between clients producing a high network overhead. Furthermore, since a client just sends data to only one client, the unicast P2P schemes achieve poor client collaboration efficiency. In [9], we proposed a P2P delivery scheme, called DDCM. In DDCM, each client (one peer) is able to send video information to a set of m clients using only one multicast channel. Furthermore, the DDCM is able to synchronize a set of clients (n peers) to create one collaboration group to replace the server in order to send video information to m peers, providing a collaboration mechanism from n-peers to m-peers. In [10], the collaboration mechanisms of DDCM are incorporated in a distributed VoD architecture with multiple service nodes. Each service node is able to create an independent P2P system to extend the global scalability.

In this paper, we propose mechanisms to implement VCR operations in the DDCM scheme. In our design, any client actively collaborates with the server to implement the join-back process of VCR operations. The new mechanisms are able to create local channels to send different playback points of one video. The jump operations could be implemented by the local channels without requiring the server resource. In a pause operation, a client extends the service time and, consequently it also extends the collaboration time. The mechanisms are able take advantage of pause operations to increase the client collaboration efficiency and reduce the system resource requirement. The reduced resource requirement by pause operations is used implement fast-forward and reverse operations.

In our study, we evaluated DDCM performance with new VCR mechanisms, compared with Patching policy. With a normal VOD system workload and with new VCR mechanisms, the experimental results show our approach is able to reduce the requirement up to 16%.

The remainder of this paper is organized as follows: we dedicate the section 2 to show the key ideas behind our VCR operation mechanisms. Performance evaluation is shown in section 3. In section 4, we indicate the main conclusions of our results and future studies.

2 P2P VoD Architecture to Provide VCR Operations

Our VoD architecture is based on multicast communications and client collaborations that decentralize the video information delivery process. Our design is not a server-less P2P architecture; the server has every video in the service catalogue. The server is responsible for establishing every client collaboration process. The collaboration scheme is designed as two policies: Patch Collaboration Manager (PCM) and Multicast Channel Distributed Branching (MCDB). The objective of PCM is to create multicast channels to service groups of clients, and allows clients to collaborate with the server in the delivery of portions of video. The objective of MCDB, however, is to establish a group of clients to eliminate on-going multicast channels that have been created by PCM.

In the explanation, we assume that video is encoded with a Constant Bit-Rate (CBR). The video information is delivered with invariable size network packets and a video is composed by *L video blocks*. Furthermore, we assume that each client has local buffers to cache a limited number of *video blocks*.

Different VCR operations, such as slow motion, are considered in [1], but we concentrate our explanation on jump forward, jump backward, fast forward, fast reverse and pause which are the most typical VCR operations. We will first explain the jump operation and then we will comment on other VCR operations that could be implemented with jumps. The explanation of our new VCR mechanisms consists of 4 points. We dedicate first two points (Section 2.1 and 2.2) to overview the client collaboration mechanism, third point to presenting details about the jump operation and the last point to discuss remaining VCR operations.

2.1 New Client Admission by PCM

Fig.1a shows the main idea of PCM. When the server is sending video block 3 to the channel $Ch3$, the client (C4) sends a control message to the server, indicating the video of interest and the size of the local buffer. The PCM tries to find an on-going multicast channel that is sending the requested video to service the client. Only on-going multicast channels ($Ch3$) with offset ($O(C3h)$) smaller than client-buffer size could be used to service the client request. The $O(Ch3)$ of channel $Ch3$ is the number of the video block that the channel $Ch3$ is currently sending. The multicast channel is called Complete Stream. The client will not receive the first portion of the video from the Complete Stream, since these

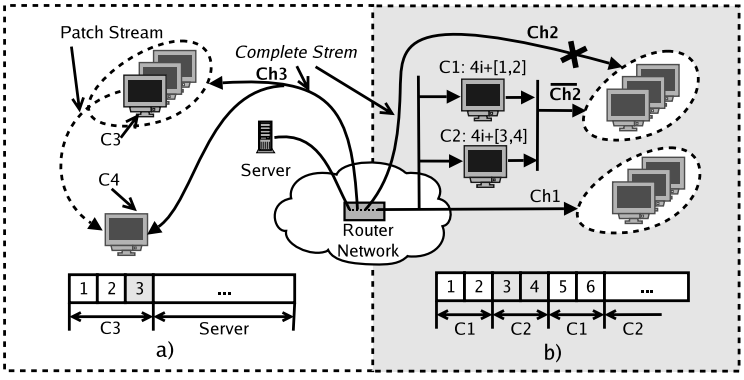

Fig. 1. a) PCM Collaboration Process. b)MCDB Collaboration Process.

video blocks (1, 2 and 3) have already been delivered by the server; therefore the server needs another channel to send the first portion of the video (the channel is called Patch Stream). In order to create the Patch Stream, the server finds a *collaborator* and requests the client's collaboration. The *collaborator*($C3$) has to have the desired video blocks and enough bandwidth to create the Patch Stream. If there is no any available *collaborator*, the server creates the Patch Stream using server resources. Finally, the client joins the Complete Stream, establishes communication with the *collaborator* and starts receiving video blocks.

In the establishment of client collaboration by PCM, the policy needs information about clients' availability for collaboration. In our design, the server uses a table structure (*Collaborator_Table*) to register the information about *video blocks* that are cached by each client in its buffer. Each client is responsible for announcing their availability to collaborate with server. The announcement message is sent only once after the client admission or after a VCR operation; the network overhead is therefore negligible.

2.2 Branching Process of MCDB Policy

Fig.1b shows the main idea behind the MCDB. The MCDB periodically checks the *Collaborator_Table* in order to find out if there are enough *collaborators* to replace some on-going multicast channels. Every channel having another channel that is sending the same video, but with a larger offset, is a candidate to be replaced by MCDB. The MCDB replaces an on-going multicast channel ($Ch2$) with a local multicast channel $\overline{Ch2}$. In order to create the local channel, a group of collaborative clients are synchronized to cache video blocks from another multicasting channel ($Ch1$). The cached blocks are delivered by the collaborative clients to generate the channel $\overline{Ch2}$. When $Ch2$ is replaced by $\overline{Ch2}$, we say that $Ch2$ is branched from $Ch1$ and $\overline{Ch2}$ is a *branch*-channel of $Ch1$.

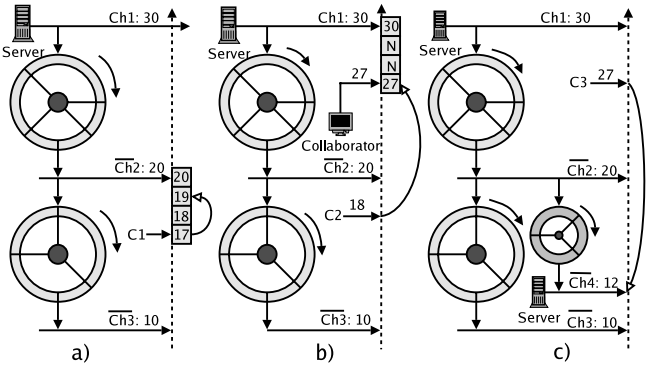

Fig. 2. Jump Operation: Serviced by a) the Client Buffer. b) by PCM. c) by MCDB.

In the example of Fig.1b, we have two channels ($Ch1$ and $Ch2$) that are separated by a gap of 2 video blocks[1]. In order to replace channel $Ch2$, MCDB selects clients C1 and C2 to create a group of collaborators. Clients C1 and C2 are both able to cache 2 blocks and a total of 4 video blocks can be cached by these two collaborators. The C1 caches every block of $Ch1$ whose block-number is $4i+1$ or $4i+2$, being $i = [0..L/4-1]$. For example, C1 has to cache blocks 1, 2, 5, 6 and so on. C2 caches every block of $Ch1$ whose block-number is $4i+3$ or $4i+4$ (3, 4, 7, 8 and so on). All the cached blocks have to be in the collaborator's buffer for a period of time. In this case, the period of time is the playback time of 2 video-blocks. After the period of time, the cached blocks are delivered to channel $\overline{Ch2}$ which is used to replace $Ch2$. It is not difficult to see that the MCDB is able to create multiple local multicast channels with different offsets to collaborate the delivery process of several points of a video. In the case of Fig.1b, the offset of the local multicast channels ($\overline{Ch2}$) is 2 video-blocks lower than $Ch1$'s.

In the branching process, two parameters are determined by the MCDB: 1) The client collaboration buffer size (B_{C_i}). It is the buffer size of client C_i used by MCDB. 2) Accumulated buffer size (BL) is the total size of the collaborative buffer ($\sum_{C_i \in CG} B_{C_i}$, being CG the group of clients). The value of these two parameters is determined by MCDB under 2 constraints: a) A client cannot use more buffer than it has. b) A client only uses one channel in the collaboration process. For more details, see [9].

2.3 Jump Forward/Backward Operation

The Fig.2a shows how the server delivers minute 30 of a video with a multicast channel ($Ch1$). The *branch*-channel ($\overline{Ch2}$) is created by MCDB and is delivering

[1] The separation or gap ($G(Ch2, Ch1)$) between two channels is calculated as $O(Ch1) - O(Ch2)$, being $O(Ch1) \geq O(Ch2)$.

minute 20 of the video. The local multicast channel ($\overline{Ch3}$) is branched from $\overline{Ch2}$. There are 3 situations depending on the new playback position after a jump.

Situation 1: The client wants to jump to a position whose information is already in the client buffer. In Fig.2a, client C1 is receiving information from $\overline{Ch2}$ and has cached video information from 17 to 20 in the buffer. Client C1 is playing minute 17 when it performs a jump forward operation to minute 19. In this case, the video information in the buffer is enough to perform the jump operation. The client skips minute 17 and 18 and immediately starts playing minute 19.

Situation 2: In this case, the information in the buffer is not able to perform the jump operation, so the client contacts with the server and the server uses the PCM policy to service the jump operation. The PCM policy finds an on-going multicast channel in which the offset is bigger than the new playback point, and finds a *collaborator* to send the Patch Stream. Fig.2b shows the delivery process after the jump forward operation of C2 to minute 27. In this example, the client will join multicast $Ch1$ to receive information from minute 30. A *collaborator* will send the video of minute 27, 28 and 29. If the server is not able to find a *collaborator*, the server creates a unicast channel to deliver the Patch Stream.

Situation 3: In this case, the PCM policy is not able to service the client with a new playback position. The server opens a new multicast channel to send the video information and triggers the MCDB policy. The MCDB policy forms a new collaborative group and the new multicast channel will finally be replaced by a local *branch*-channel. Fig.2c shows C3's jump backward action to minute 12. Notice that the PCM is not able to service the jump action because we assumed that C3 is not able to cache more than 5 minutes of video.

In three situations, a client could need the portion of buffer that is currently collaborating with the server. In such a case, the client notifies the server and stops the collaboration. Furthermore, if the buffer is not completely used in the VCR operation, the client starts caching the video information from the new playback point. Once the buffer is filled, the client sends a control message to the server to announce the new collaborative buffer capacity.

2.4 Pause and Fast Forward/Reverse Operations

The behaviour of the pause operation is quite similar to a jump. In this case, the new playback position is determined by the time that the client makes the pause. During a pause, the client continues buffering the video information. If the client is collaborating with the server, the client stops the collaboration and use all the buffer to cache more video information. However, the client-buffer could eventually overflow. In such a situation, the client temporarily stops the service. After the pause operation, the client consumes all the video blocks in the local buffer and then performs a jump action to the point where the video information is no longer in the local buffer. The jump action is then managed as a normal jump operation.

In order to implement fast forward and reverse, we assume that the server has a VCR-version for each video. The VCR-version of a video requires the same

bit-rate as a normal video but with a shorter playback time and lower frame rate. In the case of fast reverse, the video is encoded in the reverse order. The advantage of this technique is its flexibility to implement any speed of fast playback and it does not need more client network incoming bandwidth. However, it has more storage requirements. When a client issues a fast forward, the client contacts with the server to start the delivery process of the VCR-version of the video. After the VCR operation, the new playback point of the normal video is calculated and the client issues a jump operation.

3 Performance Evaluation

In this section, we show the simulation results. We have designed and implemented an object-oriented VoD simulator. Patching and PCM+MCDB with new VCR mechanisms are incorporated in the simulator that is also used in [10]. The experimental study, we evaluated VCR operations' influence and calculated the resource requirements to offer a true-VoD service. We took into account different type as well as the frequency and the duration of VCR operations. The comparative evaluation is based on the *Server Stress* and *Client Stress*. They are defined as the average amount of server and client bandwidth (Mbps) used to service all client requests.

3.1 Workload and User Behaviour Model

We assumed that the inter-arrival time of client requests follows a Poisson arrival process with a mean of $\frac{1}{\lambda}$, where λ is the request rate. We used Zipf-like distribution to model video popularity. The probability of the i^{th} most popular video being chosen is $\frac{1}{i^z \cdot \sum_{j=1}^{N} \frac{1}{j^z}}$ where N is the catalogue size and z is the *skew* factor that adjusts the probability function. For the whole study, the default *skew* factor and the video length were fixed at 0.729 (typical video shop distribution) and 90 minutes, respectively. We assumed that each video was encoded

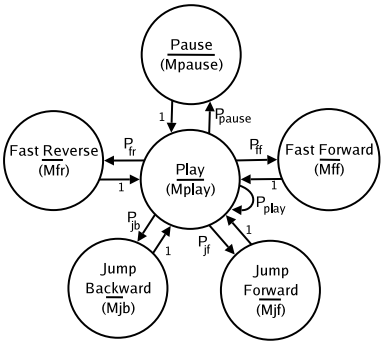

Fig. 3. Client Behaviour Model

in MPEG-2 and requires 1.5 Mbps and the service time is fixed at 24 hours. We assumed that each client is able to cache 5 minutes of the MPEG-2 video.

We assumed that the client interaction behaviour follows the model in Fig.3 which was used in [8]. The model does not try to reflect reality but includes the most common VCR operations and is able to capture two parameters: 1) VCR operation frequency. 2) the duration of each VCR operation. The VCR operation frequency is modelled by P_i's and the duration is evaluated by $\overline{M_i}$'s. The value of $\overline{M_{ff}}$ and $\overline{M_{fr}}$ indicates the average time, in seconds, that a client uses Fast Forward/Reverse. The amount of original video, in units of time, that is visualized in a Fast Forward/Reverse action depends on the fast visualization speed and the duration of the VCR operation. We assumed that the speed of any fast visualization is 2X. In the case of Jump Forward and Jump Backward, the value of $\overline{M}$ indicates the average length of video, in seconds, that will be skipped. Following this model, a client starts to visualize the video during a mean of $\overline{M_{play}}$ seconds, after this, the client issues a VCR operation with a probability of $1 - P_{play}$ or continues with playback. We assumed that the client always returns normal playback after a VCR action and that the duration of each VCR action is uniformly distributed in the interval $[\overline{M_i} \times 0.5 - \overline{M_i} \times 1.5]$.

3.2 VCR Interaction Effect

In this section, we evaluate PCM+MCDB performance with VCR operations. All the probabilities (P_i) of the clients behaviour model are fixed at 0.1 except P_{play}, which is 0.5. The time of each VCR interaction is set at 5 minutes, including playback. With these values of P_i and $\overline{M_i}$, each client could perform an average of 18 VCR operations during playback. We changed the client request rate from 1 to 40 requests per minute. The video catalogue is fixed at 200 videos.

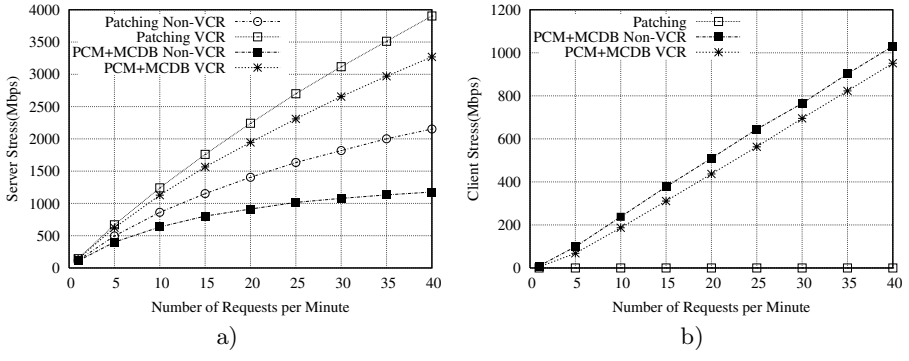

Fig. 4. VCR Interaction Effect on: a) Server Stress. b) Client Stress.

Fig.4a shows the server stress in servicing 200 videos. Without any VCR operation, the Patching policy demands 1818Mbps in order to serve 30 requests per minute. Compared with a Patching policy, PCM+MCDB reduces it by

29% (1282Mbps vs. 1818Mbps). With VCR operations and with Patching, the server stress increase up to 3117Mbps. With our mechanism, the server stress is reduced to 2654Mbps; 15% lower.

Fig.4b shows the client stress generated by PCM+MCDB policy. The Patching policy does not introduce any local overhead. With VCR operation, the client stress of our approach reduce about only 59Mbps(10%).These results indicate that the VCR operations do not affect the client collaboration capacity to decentralize the system load. The explanation for this result is:

1. The pause operations increase the service time of clients and, consequently, the time of the collaboration.
2. With jump forward operations, client service time is lower, so the client will reduce the collaboration time. However, part of the video information is also skipped, requiring less server resource.
3. In jump backward operations, the increase in server resource due to the replay of part of the video information could be rewarded with an increase in collaboration time.
4. The same explanation of jump forward/backward is applicable to fast forward/reverse. Even though, these VCR operations require extra server stress to send VCR-version of video.

3.3 VCR Interaction Frequency and Duration Effect

In this section, we evaluate the effect of VCR frequency and duration on server stress. We set the client request rate at 20 requests per minute.

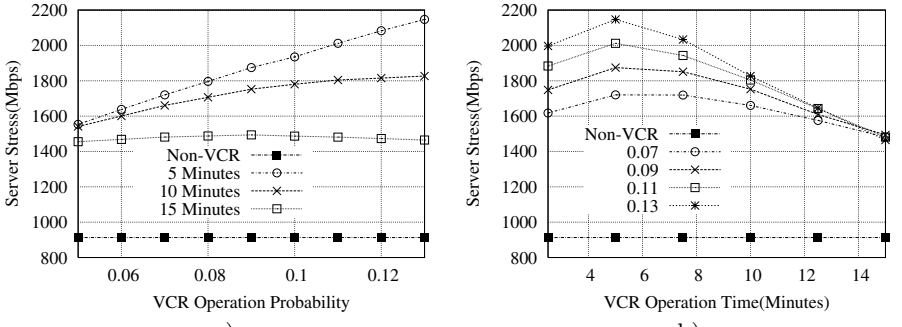

Fig. 5. a) VCR Interaction Probability Effect. b) VCR Interaction Duration Effect.

Fig.5a shows server stress in accordance with the P_is value. Different lines indicate the different durations of each VCR operation (5, 10 and 15 minutes). The value of P_i determines the number of fast forward/reverse operations that introduce extra server stress. As we can see, the server stress increases according with the VCR frequency. However, the increase also depends on the VCR durations. With $\overline{M_i} = 15Min$ the value of P_i does not affect the server stress.

These results suggest that when the duration of fast view operations is long, our approach is able to create client collaborations to decentralize these operations.

Fig.5b shows server stress in accordance with the $\overline{M_i}$s value. Different lines indicate different VCR interaction frequencies. The duration of VCR operations affects the server resource requirements for implementing VCR operations and client collaboration times. On the one hand, a longer duration of VCR operations means that the server has to send more VCR-version of video information to implement fast forward and reverse. On the other hand, a longer duration of a pause operation increases the client collaboration time, thus reducing server stress. As we can see in Fig.5b, there are two different tendencies. The first tendency is when the duration is longer than buffer size. In this case, a longer duration means less server stress and suggests that the VCR mechanism is able to efficiently use the increase in client collaboration time to reduce server stress. The second tendency happens with $\overline{M_i} < 5Min$. In this case, a longer duration increases server stress because the short jumps could be implemented with local buffer of the clients.

4 Conclusions and Future Works

We have proposed and evaluated distributed VCR mechanisms to provide true-VoD in a P2P architecture. Our mechanisms enable clients to efficiently collaborate with VoD servers to implement VCR operations.

Offering multiple videos, experimental results show that PCM+MCDB P2P delivery scheme achieves a reduction in server resource of up to 29%, compared with Patching. Several common VCR operations are analyzed in the experimental study. The experimental results show our mechanisms are very suitable to implement VCR operations with long durations because PCM+MCDB P2P delivery scheme is able to take advantage of the extra client collaboration time, introduced by VCR operations. Comparing with the Patching policy, our mechanisms are able to reduce server resource requirements up to 16%.

We have started several future research projects. First, we are developing a VoD system prototype with the P2P VCR mechanisms. Even though the partial experimental results in laboratory with the prototype have demonstrated the validity of the simulation results, we have to continue working on its implementation. Secondly, we are studying a mechanism to encourage clients to collaborate with the server even they are not playing any video.

References

1. E.L. Abram-Profeta and K.G. Shin. Providing Unrestricted VCR Functions in Multicast Video-on-Demand Servers. In *Proceedings of the IEEE International Conference on Multimedia Computing and Systems, ICMCS*, 66–75, 1998.
2. K.C. Almeroth and M.H. Ammar. The Use of Multicast Delivery to Provide a Scalable and Interactive Video-on-Demand Service, In *IEEE Journal of Selected Areas in Communications*, vol. 14, pages 1110–1122, 1996.

3. Y. Cai, W. Tavanapong and K. A. Hua. Enhancing patching performance through double patching. In *Proceeding of 9th Intl Conf. On distributed Multimedia Systems*, pages 72–77, September 24-26 2003.
4. N.L.S. Fonseca and H.K. Rubinsztejn. Channel allocation in true video-on-demand systems. In *Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE*. Pages:1999-2004 vol.3, 2001
5. Mohamed Hefeeda, Ahsan Habib, Boyan Botev, Dongyan Xu, and Bharat Bhargava. Promise: peer-to-peer media streaming using collectcast. In *Proceedings of MULTIMEDIA '03*, 45–54, New York, USA, 2003.
6. K. A. Hua, M. Tantaoui, and W. Tavanapong. Video delivery technologies for large-scale deployment of multimedia applications. In *Proceedings of the IEEE*, volume 92, September 2004.
7. Wanjiun. Liao and Victor O.K. Li. The Split and Merge (SAM) Protocol for Interactive Video-on-Demand Systems In *INFOCOM '97*, pp 1349, 1997.
8. M.A. Tantaoui, K.A. Hua and S. Sheu A Scalable Technique for VCR-like Interactions in Video-on-Demand Applications. In *The 4th International Workshop on Multimedia Networks Systems and Applications (MNSA 2002) in conjunction ICDCS 2002 Vienna, Austria*, pages 246-251, July 2-4, 2002
9. X. Y. Yang, P. Hernández, F. Cores, A. Ripoll, R. Suppi and E. Luque. Dynamic distributed collaborative merging policy to optimize the multicasting delivery scheme. In *EuroPar'05*, Lisbon, Portugal, 879–889, August 2005.
10. X.Y. Yang, P. Hernández, F. Cores, L. Souza, A. Ripoll, R. Suppi, and E. Luque. *dvodp*^{2}*p*: Distributed p2p assisted multicast vod architecture. In *Proceedings of the IEEE International Parallel & Distributed Processing Symposium (IPDPS'06)*, Rhodes, Greece, April 2006.