$DVoDP^2P$: Distributed P2P Assisted Multicast VoD Architecture [*]

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

[1]Universitat Autònoma de Barcelona
ETSE, Computer Science Department
08193-Bellaterra, Barcelona, Spain
xiao.yuan@aomail.uab.es

[2]Universitat de Lleida
Computer Science & Industrial Engineering, EPS
25001-Lleida, Spain
fcores@diei.udl.es

Abstract

For a high scalable VoD system, the distributed server architecture (DVoD) with more than one server-node is a cost-effective design solution. However, such a design is highly vulnerable to workload variations because the service capacity is limited. In this paper, we propose a new and efficient VoD architecture that combines DVoD with a P2P system. The DVoD's server-nodes is able to offer a minimum required quality of service (QoS) and the P2P system is able to provide the mechanism to increase the system service capacity according to client demands. Our P2P system is able to synchronize a group of clients in order to replace server-nodes in the delivery process. We compared the new VoD architecture with DVoD architecture based on classic multicast and P2P delivery policies (Patching and Chaining). The experimental results showed that our design is better than previous solutions, providing higher scalability.

1 Introduction

Video on Demand (VoD) is a service that will generate one of the largest revenues for private network companies, which needs performance-effective architecture solutions that provide high scalability. Depending on the number of server-nodes and their relationships, there are four representative approaches: centralized, independent server nodes, proxy and distributed servers. The *centralized* architecture uses only one server-node and it has high costs and a lack of scalability. The *independent server node* architecture uses n isolated server-nodes that move the contents close to the clients by the video replication technique. This solution is, however, poor in terms of fault-tolerance and has high storage requirements. The *proxy* approach [2] exploits the existing web proxies' capacity to cache media data, which is cost-effective. The scalability of this approach is limited by bandwidth availability of the centralized server. The *distributed server* architecture (DVoD)[3][16] eliminates the centralized server and links n server-nodes to create a distributed storage system. Each server-node only manages a video catalogue subset and video replication is only for the most popular videos, which is highly cost-effective.

In DVoD architecture, there are two types of service: local service and remote service. In a local service, a client directly receives video information from its local server-node, requiring only local-network resources and server-node resources. In a remote service, the local server-node does not have the requested video in the storage and another node is required. Thus, a remote service also requires the resource of inter-connection network. In order to minimize the resource requirements of remote services, researchers have developed video mapping strategies [3] to calculate the optimal number of replicated popular videos in each node.

Although the DVoD architecture with video mapping strategy has been shown to be effective for the VoD system, the performance it can provide is still not considered satisfactory by network companies for the following reasons. First, the limited bandwidth of server-nodes and server inter-connection network will restrict the number of clients to be served simultaneously. Second, the server-node's storage capacity is limited, and therefore an increase in video catalogue size will also increase the number of remote-services, saturating the inter-connection network. Furthermore, the server-nodes and inter-connection network are highly vulnerable to workload variations, which easily lead to system bottlenecks when there is

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

a peak workload.

In this paper, we propose a new distributed P2P assisted multicast architecture ($DVoDP^2P$). In our design, server-nodes are inter-connected just like a classical DVoD. Clients of each server-node, however, are coordinated to create a P2P system. Each server-node uses a multicast channel to send video information on one point of video. The P2P system is able to generate multiple local multicast channels that simultaneously send different points of video. In such a system, the resources of bandwidth, storage and CPU of the clients are actively used during the video delivery process, achieving high scalability. In our study, we evaluated our proposed distributed architecture according to server-node, inter-connection network and client parameters. The whole study was performed in comparison with DVoD architectures that use the well-known Patching multicast delivery policy and the well-known Chaining P2P policy.

The rest of this paper is organized as follows: in section 2, we outline related works. In section 3, we reveal our $DVoDP^2P$ architecture design. Performance evaluation is shown in section 4. In sections 5, we indicate the main conclusions and future works.

2 Related Work

Sophisticated video delivery techniques based on multicast have appeared, such as Batching, Patching [21][12], Adaptive Piggybacking [6] and Merging[4]. Even though all these multicast policies could be utilized in DVoD architecture, the performance of a multicast scheme in reducing resource consumption is limited. One of the most analyzed multicast policies in DVoD is Patching that is able to dynamically assign clients to join on-going multicast channels and patches the missing portion by unicast channels.

Most recently, the peer-to-peer (P2P) paradigm has been proposed to decentralize the delivery process to all clients. In delivery schemes such as Chaining (reviewed in [13]) or DirectStream [10], each client caches the most recently received information in the buffer and forwards it to just one client using one unicast channel. Other policies such as cache-and-relay [14] and P2Cast [9] allow clients to forward video information to more than one client, creating a delivery tree or application level multicast (ALM) tree. Other P2P architectures such as CoopNet[19] or PROMISE[11] assume that a single sender does not have enough outbound-bandwidth to send one video and use n senders to aggregate the necessary bandwidth. Nevertheless, these two architectures are also based on unicast communications. All previous P2P architectures are designed for streaming systems in global Internet without IP multicast support. In [20], we proposed the first P2P delivery scheme to use multicast communication for a centralized VoD architecture. Other P2P architecture, such as NICE[1] or Zigzag[18] adopt hierarchical distribution trees in order to provide VoD service in dynamic networks where both bandwidth and client availability for collaboration are unpredictable. In contrast, our VoD architecture is targeted at commercial media providers with large metropolitan networks. In such a system and in an enterprise network environment, the IP multicast has been successfully deployed[5]. Furthermore, client behaviour is more limited and more predictable. One of the most similar works is PROP[8]. In PROP, authors present a proxy architecture where clients are organized as a P2P system to help proxies to cache segments of video information. Our architecture distinguishes PROP at several points. First, our architecture does not require a centralized server. Second, we use the multicast communication method for client collaborations. Third, our architecture does not require a distributed content lookup protocol [15][17]. To the best of our knowledge, our proposal is the first VoD architecture that combines a multicast DVoD architecture with a P2P multicast system.

3 Distributed P2P Assisted Multicast VoD Architecture Design

3.1 Architecture Overview

In a video service, video information is sent to clients through different elements of the server and the network between the server and the client. Figure 1 shows our $DVoDP^2P$ architecture model that contains 4 main components: server-nodes, inter-connection network, clients and enterprise networks.

The **server-node** is designed in order to satisfy the soft real-time requirements of the video delivery and is only able to provide a limited *service bandwidth*. The distributed-architecture design adds a video placement policy that decides which videos are saved to each node. We addressed this problem using the results of [3] that divide the storage into 2 parts: cache and mirror. Cache is used to replicate the most popular videos and mirror is used to create a distributed storage with capacity for the entire catalogue.

Server-nodes are connected by routers that provide independent and isolated communication between server-nodes and clients. All of the routers are interconnected to create the **inter-connection network**. The inter-connection network establishes a logical topology that defines the neighbour relationship

We define the service bandwidth as the outbound bandwidth that a server-node is able to support.

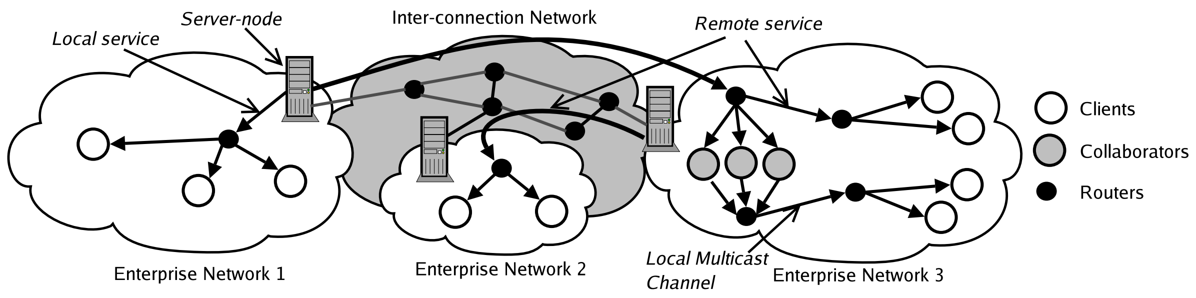


Figure 1. Distributed P2P assisted multicast VoD architecture. Enterprise Net.1 has a local service; Net.2 has a remote service; Net.3 has remote service and one channel created by P2P collaboration.

of nodes. In this logical topology, each node has a number of neighbours that could be used to create remote services. Topologies with redundancy are used to achieve high resource utilization and load-balancing. For example, with hyper-cube logical topology, each node has n neighbour server-nodes, being n the dimension of hyper-cube. If one neighbour node fails, the server-node has $n-1$ more nodes to forward a client request to.

The **client** receives, decodes and displays the video information. Throughout the process, the client includes a buffer and is able to cache video information for collaboration. Clients are connected with the server-node by the local **enterprise network**. We assume that the clients are able to deliver video information to the local network using the multicast technique and that each local client can support outbound-bandwidth for one video stream and inbound-bandwidth for two video streams. These requirements are quite compatible with such current telecommunication networks as xDSL.

Video information is assumed to be encoded at a Constant Bit-Rate(CBR). The video information is delivered in network packets; packet size is invariable and each packet contains a block of video-information bytes. We call this block a *video block*. We enumerate the video blocks of a video from 1 to L, where L is the size of a video in *video blocks*.

In our architecture, each client is responsible for announcing their availability to collaborate with the server. Each client has to send an announcement message to the local server-node and indicates the video blocks that are in the client's buffer. The announcement message is sent only once, so the network overhead is negligible. The announcement is registered by each server-node and creates a local collaborator-table that provide the content localization mechanism. Notice that each local server-node only knows the collaboration capacity of local clients.

Utilizing the collaborator-table information, our architecture defines two client collaboration policies to organize each local enterprise network in a P2P system. The two policies are called Patch Collaboration Manager (PCM) and Multicast Channel Distributed Branching (MCDB). The PCM and MCDB policies complement each other and are able to create local multicast channels utilizing client collaborations. PCM is executed when a client requests a video service and MCDB is periodically performed by each server-node.

3.2 Service Admission Process by PCM

The objective of PCM is to create multicast channels to service groups of clients in the service request admission process. PCM policy defines the communication protocol between server-nodes to establish the delivery process and allows clients to collaborate with the server in the delivery of portions of video. With PCM, clients receive video information from both a multicast and unicast channel. The multicast channels are created by a server-node (local or remote), while unicast channels could be created by a server-node or clients. Multicast channels deliver every block of a video while unicast channels only send the first portion of a video. We call the multicast channel a Complete Stream and the unicast channel a Patch Stream.

Figure 2 shows the sequence diagram of a client video request process under PCM. In a service request, a client sends a *1: Request(BufferSize)* control message to the local server (NodeA). The PCM tries to find a local on-going multicast channel sending the requested video to service the client. Only on-going multicast channels (Ch) with offset ($O(Ch)$) smaller than *BufferSize* can be used to service the client request. If there is no such on-going channel, the server tries to create a new multicast channel.

If the requested video is not in local storage, the PCM decides upon the best remote server (NodeB) and sends a *2: Forward(BufferSize)* message. The client will not receive the first portion of the video from the

The offset of channel Ch ($O(Ch)$) is the number of the video block that the channel Ch is currently sending.

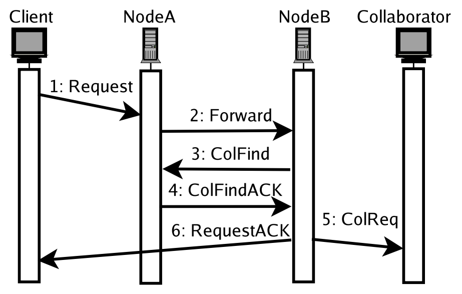


Figure 2. PCM Communication Protocol

multicast channel, since these video blocks have already been delivered by the channel. With message *3: ColFind* and *4: ColFindACK*, the remote server finds a local *collaborator* that is able to send the Patch Stream. The remote server-node sends a message *5: ColReq(PatchStreamLength)* to the *collaborator* in order to request its collaboration. If there is no available *collaborator*, the remote server creates the unicast channel to send the Patch Stream.

In the last PCM step, the PCM sends a *6: RequestACK(ServerId, MulticastChannelId, collaborator)* message to the client to start the delivery process. The client joins the multicast channel (Complete Stream), establishes communication with the *collaborator* and starts receiving video blocks. The video blocks that are received from the Patch Stream are immediately displayed and blocks from the Complete Stream are cached for later display.

3.3 Multicast Channel Distributed Branching by MCDB

The objective of **MCDB** is to establish a group of clients to eliminate on-going multicast channels that have been created by PCM. 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 $Ch1$.

Figure 3 shows the main idea behind the MCDB. In this example, we have two channels ($Ch1$ and $Ch2$) that are separated by a gap of 2 video blocks. In order to replace channel $Ch2$, MCDB selects clients C1 and C2 to create the 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

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

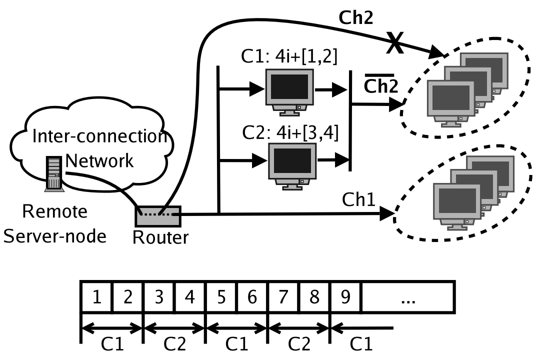


Figure 3. MCDB Collaboration Process

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 Figure 3, 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 the [20].

3.4 Channel Selection by MCDB

MCDB is able to replace every channel that is sending information to a local client. In the event of MCDB being able to replace more than one channel, our policy establishes an order of replacement. The first-replaced channel is the one that releases the most network resources and requires less client buffer. We define two parameters: 1) $NetResource(Ch)$: the network resource associated with a channel Ch to send information to the local clients. The value of $NetResource(Ch)$ is 1, if the channel is from the local server. In another case, the value is calculated according to the distance between the remote server and the local network. 2) $Cost(Ch)$: is the gap (G), in video blocks, between channel Ch and the other chan-

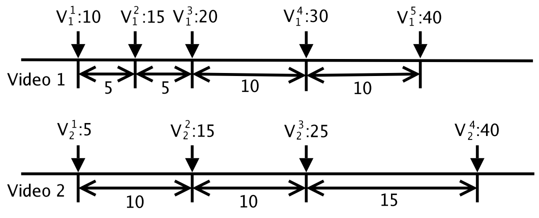


Figure 4. MCDB Ramification Order Example

Branching Pair($j \to i$)	Net. Resource(i)	Cost(i)	Performance(i)	Order
$V_1^5 \to V_1^4$	1	10	0.1	7
$V_1^4 \to V_1^3$	1	10	0.1	6
$V_1^3 \to V_1^2$	1	5	0.2	5
$V_1^2 \to V_1^1$	1	5	0.2	4
$V_2^4 \to V_2^3$	3	15	0.2	3
$V_2^3 \to V_2^2$	3	10	0.3	2
$V_2^2 \to V_2^1$	3	10	0.3	1

Table 1. MCDB Ramification Order

nel that Ch will be branched from. The MCDB calculates $Performance(Ch) = \frac{NetResource(Ch)}{Cost(Ch)}$ for each channel and chooses the multicast channel with the highest $Performance(Ch)$ value as the channel to replace. In the case of there being more than one channel with the same value in $Performance(Ch)$, the MCDB chooses the channel with the smallest offset.

Figure 4 shows an example of the delivery process of video 1 and video 2. Assume that video 1 is delivered by the local server-node and video 2 is sent by a remote server-node. The local server-node uses 5 multicast channels (V_1^i) to simultaneously deliver blocks 10, 15, 20, 30 and 40 of video 1. In the case of the remote server-node, there are 4 channels (V_2^i) to send video 2. Assuming also that the distance between the local network and the remote server-node is 3, the Table 1 shows the value of $NetResource(Ch)$, $Cost(Ch)$ and $Performance(Ch)$ for each possible pair of multicast channels. The ramification order is in the fifth column. In this case, the MCDB will try to replace channel V_2^1 of video 2 with a *branch*-channel from V_2^2.

3.5 Collaborator Selection by MCDB

Each server-node has a list of clients that are willing to collaborate in the MCDB branching process. The objective of the collaborator-selection mechanism is that of choosing a group of clients that produces low network overhead in the collaboration process. Our intuition is to try to select clients that are very close together. Our selection strategy is based on partial acknowledgment of collaborators. Each collaborator randomly selects k collaborators to estimate the distance in number of network-hops. The distance information is collected by the server in the collaborator-table.

Figure 5 shows a possible local network configura-

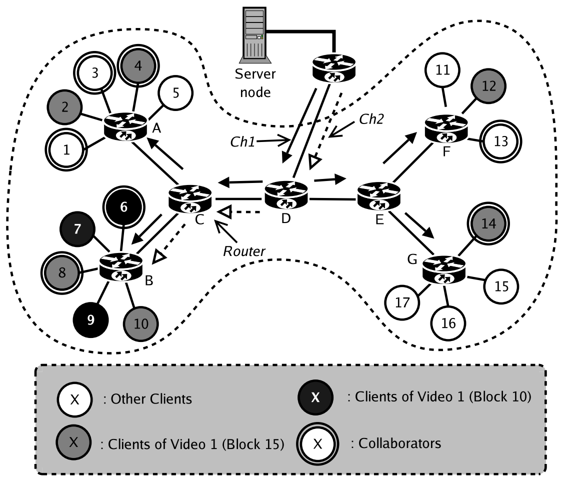


Figure 5. Net. Configuration with 17 Clients

	Distances between collaborator i and j		
Collaborator i	Distance 1	Distance 2	Distance 3
1 (Ch?)	$1 \to 3 = 2$	$1 \to 8 = 4$	$1 \to 13 = 6$
3 (Ch?)	$3 \to 1 = 2$	$3 \to 14 = 6$	$3 \to 6 = 4$
4 (Ch1)	$4 \to 1 = 2$	$4 \to 6 = 4$	$4 \to 14 = 6$
6 (Ch2)	$6 \to 8 = 2$	$6 \to 13 = 6$	$6 \to 14 = 6$
8 (Ch1)	$8 \to 3 = 4$	$8 \to 4 = 4$	$8 \to 13 = 6$
13 (Ch?)	$13 \to 3 = 6$	$13 \to 14 = 4$	$13 \to 6 = 6$
14 (Ch1)	$14 \to 13 = 4$	$14 \to 8 = 6$	$14 \to 3 = 6$

Table 2. Distance Information, being k=3

tion. In this case, there are 7 routers (A-G) and 17 clients. There are 3 clients (black nodes) that are receiving block 10 of video 1 by channel $Ch2$. There are 6 clients (grey nodes) are receiving block 15 of video 1 by channel $Ch1$. There are also 6 clients (1, 3, 4, 6, 8, 13 and 14) that are willing to collaborate with the server-node, represented by nodes surrounded by a cycle. The white nodes represent clients that are receiving other videos by channel $Ch?$. We assume that the value of k is 3 and Table 2 shows the distance information that is estimated by the 7 collaborators. For instance, collaborator 1 estimated the distance to the 3 collaborators 3, 8 and 13. The distance between collaborator 1 and collaborator 13 is 6 (represented as $1 \to 13 = 6$); one message from 1 to 13 has to perform 6 hops. If $Ch2$ is the channel that we are interested in replacing, based on the distance information, MCDB follows 4 steps to select the list of collaborators:

Step 1: Initialize the $CollaboratorList$ with all the collaborators that are also receiving from channel $Ch2$.

Step 2: If there are enough collaborators, End.

Step 3: $C \leftarrow$ the collaborator at the shortest distance from any collaborator of $CollaboratorList$

Step 4: $CollaboratorList \leftarrow CollaboratorList + C$ and jump to Step 2.

The reasoning behind step 1 is that if a collaborator is already receiving from channel $Ch2$, the collaborator will not consume any more network resource to send the information to the same channel. In each iteration, steps 3 and 4 insert a new collaborator into the *CollaboratorList*. In accordance with Table 2, the strategy will select collaborator 6 to be the first collaborator. With 4 iterations, the *CollaboratorList* will be [6],[6,8],[6,8,3],[6,8,3,1]. As we can see, collaborators [6,8,3,1] are connected to router A and B, therefore, the branching process only consumes the network resource of links between A-C and B-C.

3.6 MCDB Policy Complexity

In order to select the best multicast channel to perform the branching process, the MCDB policy has to calculate the gap (in number of blocks) between every pair of multicast channels and the distances (in number of network hops) of every remote server-node. The gap between channels depends on the time that two channels are created by PCM. Furthermore the gap between two channels is constant because we use CBR videos. Therefore, the gap computation only happens once. In order to estimate the distance between two nodes, we could use ping-pong messages to measure the message latency and therefore infer the number of hops between two points in the network.

The collaborator selection mechanism not only introduces network overhead associated with collaborator distance estimation, but also computation overhead in server-node. Being k the number of collaborators that have to estimate the distance, m the total number of collaborators and q the number of messages associated with each distance estimation, the network overhead is $q \times k \times m$. In each iteration i, the selection mechanism has to compare $i \times k$ values in order to select a new collaborator. Thus, the computation complexity of the selection mechanism to select n collaborators is $\sum_{i=1}^{n} i \times k = k \times \frac{n \times (n+1)}{n} \cong O(n^2)$. Note that the value of n is usually small; about 5.

3.7 Discussion

In spite of the fact that the two collaboration policies of $DVoDP^2P$ architecture are complementary to each other, the main client contribution in the delivery process is from MCDB because the multicast channels consume most of the resources. Both PCM and MCDB have to create multicast channels. We assumed that the underlying network provides QoS-aware multicast mechanisms that include routing, resource reservation, reliability, and flow and congestion control protocols. See more details in [5]. Even though all these mechanisms are available in enterprise networks, the IP multicast service model is extremely complex to implement on the Internet. The $DVoDP^2P$ architecture only needs IP multicast support in each local network and multicast service for remote service could be implemented with software-based overlay multicast.

The MCDB collaborator selection mechanism is based on network topology information. Such information, however, is very difficult to obtain and make it be impossible to get the optimal collaborator selection. The MCDB design is not able to get the optimal selection because it is based on partial acknowledgements. The main advantage of the MCDB selection mechanism is the low complexity and overhead. However, the mechanism performance closely depends on the value of k which should be determined according with the local network topology complexity.

The general tendency in P2P system design is heading towards server-less architecture. However server-less architecture for VoD has several inconveniences: 1) Peers are responsible for initializing the video content, which could involve copyright problems. 2) The system cannot provide a minimum required service capacity, if peers decide not to collaborate. 3) Due to fault tolerance questions, the information is duplicated in peers. Since the majority of videos (80%) in a VoD system are unpopular, a server-less architecture may achieve poor peer efficiency. The two first questions are very important for commercial applications and the last question is related to architecture performance. None of these 3 problems are present in $DVoDP^2P$ architecture.

4 Performance Evaluation

In this section, we show the simulation results for $DVoDP^2P$ architecture, Contrasted with DVoD architecture that utilizes a Patching multicast policy and Chaining P2P policies. The Patching policy in DVoD is widely analyzed in [3]. However there is no study that analyses the performance of Chaining policy in a DVoD. Furthermore, there is no comparative study of Chaining and Patching[13]. We adapted the Chaining technique in a DVoD architecture in order to make comparisons. In our modification of the Chaining, clients collaborate with the server-node (local or remote) in order to deliver video information to local clients. Like the PCM+MCDB, under the adapted Chaining policy, a client is not allowed to send video information to clients in another local network.

We evaluated parameters associated with server-nodes and clients. In terms of server-nodes, we changed the total storage size and the number of the most popular videos replicated on each server. These two parameters especially affect the number of remote client requests. In terms of clients, we evaluated $DVoDP^2P$ with different client-buffer sizes that affect the client collaboration capacity. We also evaluated $DVoDP^2P$ with different client-request rates and

Parameter	Default	Analyzed Values
Zipf Skew Factor	0.729	0.729
Video Length	90	90 Minutes
Simulation Time	7500	7500 Seconds
Video Replication	50%	20-80%)
Video Catalogue Size	100	50-400 Videos
Client-buffer size	5	(2,3,...15) Minutes
Request Rate	30	(5-40) $\frac{Request \times Server}{Minute}$
Storage Size	22	(10-100) Videos
Topology	Hyper-Cube	Hyper-Cube, Torus, P-Tree, Double P-Tree

Table 3. Simulator Environment Parameters

numbers of videos in the catalogue in order to show the scalability of our architecture.

4.1 Workload and Metrics

In our experiments, 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 $\frac{1}{i^z \cdot \sum_{i=1}^{N} \frac{1}{j^z}}$ where N is the catalogue size and z is the *skew* factor that adjusts the probability function[7]. For the whole study, the *skew* factor and the video length are fixed at 0.729 (typical video shop distribution) and 90 minutes, respectively. The simulation time is fixed at 7500 seconds (125 minutes), which is enough to determine performance metric values. We evaluated 4 inter-connection network topologies: Hyper-cube, Torus, P-Tree and Double P-Tree. P-Tree and Double P-Tree are proposed in [3] and are tree-based topologies. Due to the space limitation, we only show the results for Hyper-Cube topology with 32 server-nodes. The analyzed values of the parameters are summarized in Table 3. Default values are indicated in the second column.

The comparative evaluation is based on 3 metrics calculated at the end of the simulation. 1) the *Server-node load* is defined as the mean number of streams active in each server-node at the end of the simulation. 2) The *Inter-connection network load* is defined as the mean number of streams crossing one network link in the inter-connection network. 3) the *Local-network overhead* is the average number of streams opened by delivery policies to allow for client collaboration. Each metric is evaluated as the number of streams representing the sum of multicast channels and unicast channels.

4.2 Server Node Storage Capacity Effect

We change the storage size of each server node from 10 to 80 videos (10-80% of the catalogue size). All other parameters have default values (Table 3). Figure 6.a shows the average load in each server node. Compared with Patching, $DVoDP^2P$ produces 33-57% less server load. Compared with a Chaining policy, both $DVoDP^2P$ and Patching are far better solutions if storage size is lower than 30%. If storage capacity is low (10-20%), the distributed system has only one copy of each unpopular video. In this situation, all client requests for an unpopular video are forwarded to one server-node. Client requests from different nodes can share one multicast channel in multicast policies (Patching or $DVoDP^2P$). With a storage size of 50% of the catalogue, a Chaining policy is better than Patching. However, a DVoD+Chaining is still improved by $DVoDP^2P$, which requires 30% less of the server node bandwidth.

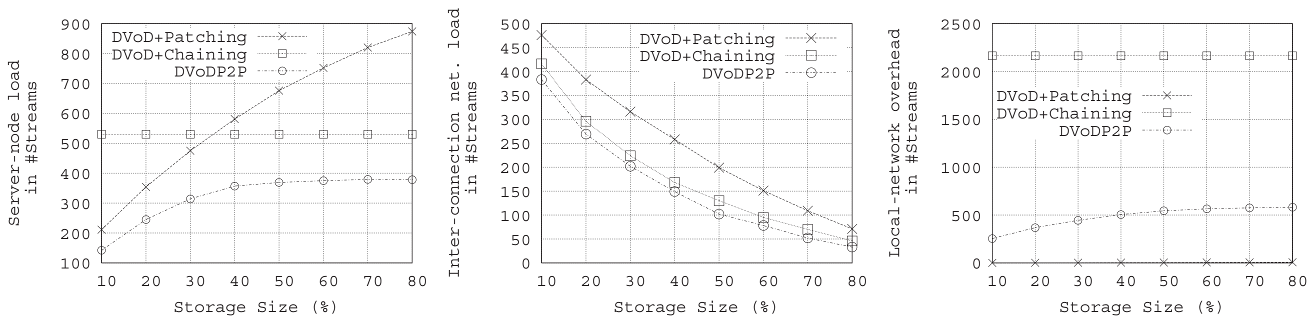
In terms of inter-connection network load (Figure 6.b), $DVoDP^2P$ is 19-53% lower than DVoD+Patching policy. Compared with a DVoD+Chaining policy, the $DVoDP^2P$ decreases the network load by 8-27%. These results indicate that P2P collaboration can reduce the inter-connection network bandwidth requirements, independently of the amount of server-node storage capacity.

Figure 6.c shows the local network overhead generated by Chaining and PCM+MCDB policies. The Patching policy does not introduce any local overhead. The Chaining policy generates a constant network overhead, regardless of storage size. These results are to be expected, because clients collaborate only with the server-node in delivering the most popular videos. The number of collaborations of Chaining depends on the client-request rate and video-popularity distribution. The local-network overhead of PCM+MCDB is 73-88% lower than a Chaining policy.

We also changed the percentage of storage that each server-node dedicates to video replication from 20% to 80% of storage capacity. The results show that the requirement reduction provided by $DVoDP^2P$ in terms of server-node load and local-network load is not changed by the number of replicated popular videos, suggesting that $DVoDP^2P$ will be better than DVoD+Patching and DVoD+Chaining, independently of the video mapping policy. Compared with DVoD+Patching and DVoD+Chaining, $DVoDP^2P$ is able to reduce server-node load by 24-30% and 51-54%, respectively, when we change the storage capacity for video replication.

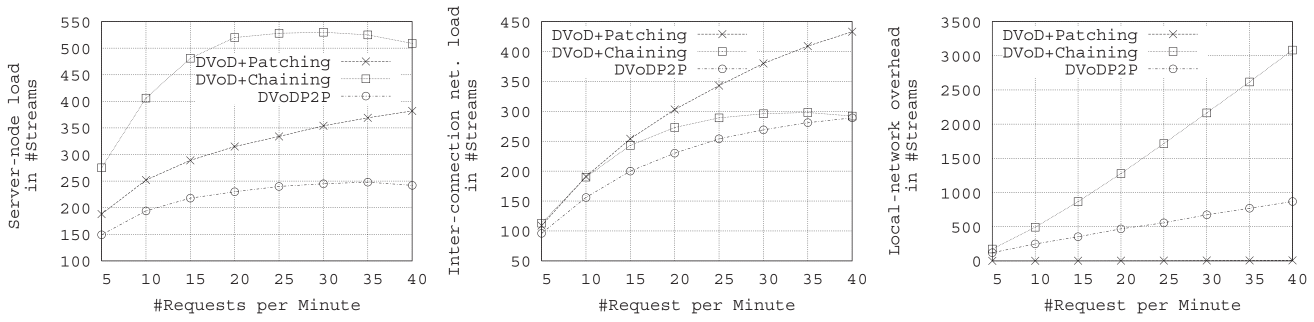
4.3 Client-request Rate Effect

In this experiment, we changed the client-request rate from 5 to 40 requests per minute in each server-node. The other system parameters are assumed to have the default values in Table 3. Figure 7 shows the results of the 3 performance metrics with 3 delivery



a) Server-node Load b) Inter-connection Network Load c) Local-network Overhead

Figure 6. Server-node Storage Size Effect on Hyper Cube



a) Server-node Load b) Inter-connection Network Load c) Local-network Overhead

Figure 7. Client-request Rate Effect on Hyper Cube

policies. The results show that our architecture significantly reduces the sever-node and inter-connection network load. Compared with the DVoD+Patching, $DVoDP^2P$ is able to reduce average server-node load by 37%, and inter-connection network load by 33%.

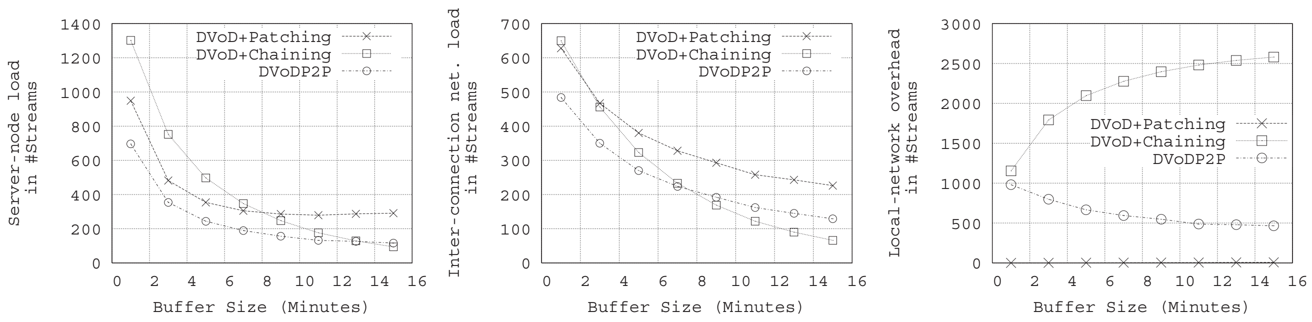
Compared with DVoD+Chaining, our design can reduce server-node and inter-connection network load by 56% and 18%, respectively. In terms of server-node load, DVoD+Chaining is worse than DVoD+Patching because the client collaboration with the DVoD+Chaining occurs only at a very high request rate. Furthermore, unpopular videos are serviced by unicast channels. In terms of local-network overhead, and compared with DVoD+Chaining, $DVoDP^2P$ reduces such overhead by 72%. The low local-network overhead is due to PCM+MCDB P2P policy design replacing only multicast channels and the number of multicast channels per video being independent of the client-request rate.

4.4 Client-buffer Size Effect

All system parameters are fixed at default values, except for client-buffer size, which changes from 1 to 15 minutes. Figure 8.a shows the mean server node load of 3 architectures. The server-node load of DVoD+Patching, DVoD+Chaining and $DVoDP^2P$ decreases in accordance with client-buffer size. With 11-13 minutes of buffer, DVoD+Chaining achieves the same performance as $DVoDP^2P$. Compared with DVoD+Chaining, the server-load of $DVoDP^2P$ is 47-25% lower (if the buffer size is lower than 11 minutes). Compared with DVoD+Patching, the server-load of $DVoDP^2P$ is 27-60% lower.

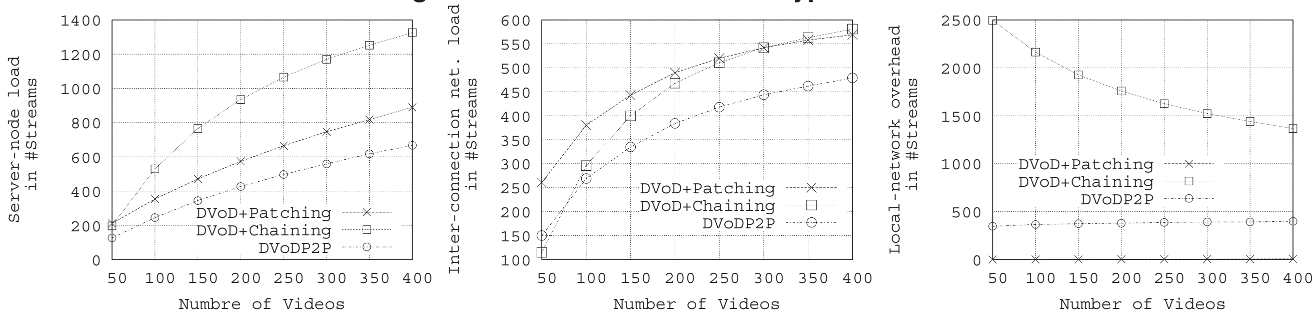
The average inter-connection network load is plotted in Figure 8.b. With 1 minute of buffer, the network load for DVoD+Chaining and DVoD+Patching is almost the same. With more buffer, DVoD+Patching and DVoD+Chaining performance increases. With 8 minutes, the DVoD+Chaining generates a lower inter-connection network load than $DVoDP^2P$.

Analysis of the effect of client-buffer size on the local-network overhead is shown in Figure 8.c. In DVoD+Chaining, the local-network overhead increases along with a higher buffer size, while the $DVoDP^2P$ is able to keep overhead levels at between 466 and 981 channels. Compared with DVoD+Chaining, $DVoDP^2P$ introduces 15-82% less local-network overhead. These results suggest that $DVoDP^2P$ is able to achieve better performance than DVoD+Chaining with low buffer requirements (1-10 minutes). Furthermore, the low local-overhead of $DVoDP^2P$ indicates that PCM+MCDB are low cost P2P delivery policies to organize P2P systems.



a) Server-node Load b) Inter-connection Network Load c) Local-network Overhead

Figure 8. Buffer Size Effect on Hyper Cube



a) Server-node Load b) Inter-connection Network Load c) Local-network Overhead

Figure 9. Video Catalogue Size Effect on Hyper Cube

4.5 Video Catalogue Size Effect

The number of videos in the catalogue is modified from 50 to 400 videos in this experiment. Each server node is assumed to have sufficient storage size to save up to 20% of the catalogue, and 50% of the storage is dedicated to the replication of the most popular videos. Other system parameters are fixed with default values in Table 3. Figure 9.a shows the mean server-node load in DVoD+Patching, DVoD+Chaining and $DVoDP^2P$. Regardless of the delivery policy, the server load increases with more videos. However, the increase in DVoD+Chaining is much higher than 2 multicast architectures. The explanation of such a high increase is that a Chaining policy uses unicast channels to send unpopular videos. More videos in the catalogue necessarily imply more unpopular videos and consequently, the server load drastically increases with a Chaining policy. Compared with a DVoD+Chaining, the $DVoDP^2P$ is able to reduce server load by 55%. This reduction is around 24-40%, if we compare $DVoDP^2P$ with DVoD+Patching.

In terms of inter-connection network (Figure 9.b), $DVoDP^2P$ is able to reduce network load by 16-42%, compared with DVoD+Patching. DVoD+Chaining introduces nearly the same network load as $DVoDP^2P$ when the number of videos is 50. With a catalogue of 300 videos, DVoD+Chaining requires the same network resource as a DVoD+Patching.

Figure 9.c shows the local-network overhead. A decrease in the network overhead of DVoD+Chaining indicates that Chaining-policy performance decreases with catalogue size. $DVoDP^2P$ is able to maintain the number of local multicast channels when the catalogue is large, suggesting that $DVoDP^2P$ performance will not degrade in accordance with the number of videos. Furthermore, the local-network overhead of $DVoDP^2P$ is 71-86% lower than DVoD+Chaining.

5 Conclusions and Future Work

We have proposed and evaluated new distributed VoD architecture that combines the distributed server architecture (DVoD) with multicast P2P system. In $DVoDP^2P$, every client is able to collaborate with server-nodes, regardless of the video they are watching. Instead of independent collaborations between the server and a client, our P2P system design synchronizes groups of clients in order to create *branch* local multicast channels that replaces on-going channels, reducing both server-node load and inter-connection network load. The $DVoDP^2P$ design is able to delegate clients to send both popular and unpopular videos.

The experimental results show that $DVoDP^2P$ needs fewer resource requirements than a DVoD with multicast delivery policy, managing to reduce the server-node load and inter-connection network load by 37% and 33%, respectively. These results corroborate the high scalability of our design when there are a large number of client requests. Furthermore, the P2P system design has a lower local-network overhead than a Chaining P2P system. A reduction of up to 72% in local-network overhead is observed in the experimental results. All these results demonstrate that $DVoDP^2P$ is a highly scalable and cost-effective distributed VoD architecture.

We are extending $DVoDP^2P$ in several directions. First, clients can also collaborate with the server to implement VCR operations. Second, $DVoDP^2P$ has to implement some kind of fault tolerance schemes; backup clients could be used in the collaboration process. Third, with current broadband network technology, clients could be connected to the network even if they are not watching any video. All these “non-active” clients could also collaborate in the delivery process.

References

[1] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable application layer multicast. In *Proceeding of SIGCOMM '02*, pages 205–217, New York, NY, USA, 2002. ACM Press.

[2] S. Chen, H. Wang, X. Zhang, B. Shen, and S. Wee. Segment-based proxy caching for internet streaming media delivery. *IEEE MultiMedia*, 12(3):59–67, July-September 2005.

[3] F. Cores, A. Ripoll, X. Y. Yang, B. Qazzaz, R. Suppi, P. Hernández, and E. Luque. Improving bandwidth efficiency in distibuted video-on-demand architectures. *Parallel Processing Letters*, 13 No. 4:589–600, 2003.

[4] D. Eager, M. Vernon, and J. Zahorjan. Bandwidth skimming: A technique for cost-effective video-on-demand. In *IS&T/SPIE Conf. on Multimedia Computing and Networking (MMCN 2000)*, pages 206–215, San Jose, January 2000.

[5] A. Ganjam and H. Zhang. Internet multicast video delivery. *Proceedings of the IEEE*, 93(1):159–170, 2005.

[6] L. Golubchik, J. C. S. Lui, and R. R. Muntz. Adaptive piggybacking: a novel technique for data sharing in video-on-demand storage servers. *Multimedia Syst.*, 4(3):140–155, 1996.

[7] C. Griwodz, M. B., and L. C. Wolf. Long-term movie popularity models in video-on-demand systems: or the life of an on-demand movie. In *Proceedings of the fifth ACM international conference on Multimedia*, pages 349–357. ACM Press, 1997.

[8] L. Guo, S. Chen, S. Ren, X. Chen, and S. Jiang. Prop: A scalable and reliable p2p assisted proxy streaming system. In *24th International Conference on Distributed Computing Systems (ICDCS'04)*, 2004.

[9] Y. Guo, K. Suh, J. Kurose, and D. Towsley. P2cast: peer-to-peer patching scheme for vod service. In *WWW '03: Proceedings of the twelfth international conference on World Wide Web*, pages 301–309. ACM Press, 2003.

[10] Y. Guo, K. Suh, J. Kurose, and D. Towsley. A peer-to-peer on-demand streaming service and its performance evaluation. In *2003 IEEE International Conference on Multimedia & Expo (ICME 2003), Baltimore, MD*, July 2003.

[11] M. Hefeeda, A. Habib, B. Botev, D. Xu, and B. Bhargava. Promise: peer-to-peer media streaming using collectcast. In *Proceedings of the eleventh ACM international conference on Multimedia*, pages 45–54, New York, NY, USA, 2003. ACM Press.

[12] K. A. Hua, Y. Cai, and S. Sheu. Patching: A multicast technique for true video-on-demand services. In *ACM Multimedia Conf., Bristol, U.K.*, September 1998.

[13] 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.

[14] S. Jin and A. Bestavros. Cache-and-relay streaming media delivery for asynchronous clients. In *NGC'02, Boston, MA, USA*, October 2002.

[15] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A scalable content-addressable network. In *Proceedings of SIGCOMM'01*, pages 161–172, New York, NY, USA, 2001. ACM Press.

[16] C. Shahabi and F. Banaei-Kashani. Decentralized resource management for a distributed continuous media server. *IEEE Trans. Parallel Distrib. Syst.*, 13(7):710–727, 2002.

[17] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 149–160, New York, NY, USA, 2001. ACM Press.

[18] D. A. Tran, K. A. Hua, and T. T. Do. A peer-to-peer architecture for media streaming. *IEEE Journal on Selected Areas in Communications*, 22(1), January 2004.

[19] P. Venkata, N., H. J. Wang, P. A. Chou, and K. Sripanidkulchai. Distributing streaming media content using cooperative networking. In *Proceedings of the 12th inter. workshop on Network and operating systems support for digital audio and video*, pages 177–186, New York, NY, USA, 2002. ACM Press.

[20] X. Yang, P. Hernández, F. Cores, A. Ripoll, R. Suppi, and E. Luque. Distributed p2p merging policy to decentralize the multicasting delivery. In *31st EuroMicro Conference on Sofhtware Engineering and Advanced Applications*, pages 322–329, Porto, Portugal, Agust-September 2005.

[21] W. T. Ying Cai and K. A. Hua. Enhancing patching performance through double patching. In *9th Intl Conf. On distributed Multimedia Systems*, pages 72–77, September 24-26 2003.