DynaPeer: A Dynamic Peer-to-Peer Based Delivery Scheme for VoD Systems*

Leandro Souza[1], Fernando Cores[2], Xiaoyuan Yang[1], Ana Ripoll[1]

[1] Computer Architecture and Operating System
Universitat Autònoma de Barcelona, 08193 Bellaterra, Spain
[2] Computer Science and Industrial Engineering
Universitat de Lleida, St. Jaume II, 69, 25001 Lleida, Spain

Leandro@aomail.uab.es, Fcores@diei.udl.es, Xiao@aomail.uab.es, Ana.Ripoll@aub.es

Abstract. Supporting Video-on-Demand (VoD) services in Internet is still a challenging issue due to high bandwidth requirement of multimedia contents and additional constraints imposed by such environment: higher delays and jitter, network congestion, non-symmetrical clients' bandwidth and inadequate support for multicast communications. This paper presents DynaPeer, a peer-to-peer VoD delivery policy designed for Internet environment. Our design defines a Virtual Server, which is responsible for establishing a group of peers, enabling service for new client requests by aggregating the necessary clients' resources. Virtual Server operates in both unicast and multicast environments, thereby improving system performance. To demonstrate the effectiveness of DynaPeer, we have developed an analytical model to evaluate its performance, understood as the server-load reduction due to request service distributed among peers. We conducted a performance comparison study of our proposal with classic unicast, multicast (Patching) and other P2P delivery schemes, such as Pn2Pm, Chaining and Promise, improving their performance by 45%, 59%, 74% respectively, even when taking into account Internet constraints.

Keywords: On-Demand Media Streaming, Peer-to-Peer systems, Internet VoD.

1 Introduction

Advances in network technology will provide the access to new generation, full-interactive and client-oriented services such as Video-on-Demand. Through these services, users will be able to view videos from remote sites at any time. However, serving video files to a large number of clients in an "on demand" and "real time" way imposes a high bandwidth requirement on the underlying network and server.

To spread the deployment of VoD systems, much research effort [4][6][7][11] has been focused on the delivery process of multimedia contents, exploiting both unicast and multicast techniques, trying to reduce the bandwidth consumption and provide better system streaming capacity. In spite of the success of these techniques, their

* This research is supported by the MEyC-Spain under contract TIN 2004-03388.

scalability requirements to provide service on a large-scale system, such as Internet, is still limited by server and network resources.

Recently research has looked to the peer-to-peer (P2P) paradigm as a solution to decentralize the delivery process among peers, alleviating the server load or avoiding any server at all. P2P systems for streaming video have generated important contributions. In the Chaining delivery policy [7], clients cache the most recently received video information in the buffer and forward it to the next clients using unicast channels. The P2Cast [4] and cache-and-relay [8] allow clients to forward the video data to more than one client, creating a delivery tree or ALM. However, neither chaining or ALM delivery policies consider client output-bandwidth limitation in collaboration process, which limits their usage to dedicated network environments. Other VoD P2P-based architectures such as PROMISE [1], CoopNet [9] or BitVampire [10] assume that a client does not have sufficient output bandwidth to generate the complete information to other clients, using n clients to send the required bandwidth. However, they assume that clients work as proxies storing whole video information. Furthermore, system scalability is compromised due to unicast communication. To solve the scalability problem, in previous works [11] we proposed P^n2P^m architecture that takes advantage of multicast technology on the client side. This architecture works by exploiting the clients non-active resources in two ways: first, it allows clients to collaborate with the server in the delivery of initial portions of video, patches streams; and second, it establishes a group of clients to store the available information of an existent server multicast channel to eliminate it. P^n2P^m also requires that output bandwidth is, at least, the same as video play-rate.

The Internet environment imposes further restrictions to P2P streaming schemes in order to provide VoD service. First, providing service over non-dedicated network environments implies no QoS guaranties, transmission congestion, packet loss and variable point-to-point bandwidth. Second, non-symmetrical clients' bandwidth involves a careful delivery strategy due to clients' output-bandwidth limitation. Third, Internet Service-Provider[2] (ISP) networks differ on supporting (or not) the IP-Multicast delivery technology. Finally, content copyright protection affects content storage limited to non-persistent devices. Thus, content on peers is only available over a limited period of time.

To solve the above challenges, we propose a new delivery scheme called DynaPeer, based on a P2P paradigm for an Internet environment. DynaPeer differs from the previous P2P schemes in certain key aspects. First, DynaPeer works with unicast and multicast communication techniques, depending on the technology available to the ISP network. The combination of unicast and multicast could allow DynaPeer to dynamically exploit the IP multicast mechanism, achieving better network utilization and providing system scalability. Second, this scheme takes into consideration the non-symmetric characteristics of client bandwidth, which is in accordance with current xDSL technology. Third, our delivery scheme assumes the non-homogeneity characteristics founded on a non-dedicated network such as Internet, which allows us to design a realistic delivery scheme for VoD services. To the best of our knowledge, our proposal is the first VoD delivery scheme that

[2] Currently, certain ISPs provide multicast technology over the xDSL, through DsLAN technology. Authors in [3], have demonstrated its usage and have proposed a connectivity architecture for multicast ISPs.

combines non-dedicated network environment, asymmetrical connection on the client side and multicast delivery technique for client collaborations.

The remainder of this paper is organized as follows. In section 2 we present DynaPeer design. In section 3, an analytical model to evaluate DynaPeer performance is presented. Section 4 shows the performance evaluation through the analytical model. In Sections 5, we indicate the main conclusions and future works.

2 DynaPeer Design

DynaPeer is not a server-less system; rather, it combines a server-based architecture with a P2P delivery scheme. The server holds the entire system catalogue, acting as seeds for the multimedia content. It is also responsible for establishing every client-collaboration process. DynaPeer takes advantage of client collaboration to decentralize the server-delivery process, eventually shifting streaming load to peers.

The explanation of DynaPeer is divided into 3 parts. Section 2.1 describes the collaboration model of DynaPeer and in sections 2.2. and 2.3, we present P2P delivery schemes over unicast and multicast environments.

2.1 Collaboration Model

The principle of DynaPeer is based on clients' collaborations in which clients (peers) make their idle-resources available so as to generate a complete, or partial, stream for incoming clients. In our system, a peer is an active client who plays a given video and is able to collaborate with the system.

Peers' collaboration capacity is limited by peer resources (bandwidth and storage) and available video data. In our case, we consider that peers have an asymmetrical input/output bandwidth (input bandwidth is, at least, the same as video play-rate and output bandwidth is supposed to be lower than video play-rate) and a limited buffer capacity. Having insufficient output bandwidth to transmit a complete video stream implies that several peers (N_i, defined by the ratio between video i play-rate and peers' output bandwidth) have to collaborate in order to provide service for a complete streaming session. Furthermore, due to copyright protection and peers' limited buffer capacity, peers cannot permanently store a complete video. Therefore, they can only serve, on the fly, video data previously received from an active streaming session and temporally stored on clients' buffer.

All collaborations in DynaPeer are managed by the *Virtual Server* (VS). The objective of a VS is to establish a group of peers, aggregating sufficient resources, enabling the service for new clients' requests. Another important function attributed to the VS is to perform distributed control tasks among peers in a distributed way, minimizing server involvement. A virtual server (Fig. 1), denoted by VS(*j,s,w*), is a logical entity defined as a set of peers that collaborate in a delivery process to offset *s* of video *j*, during a period of time *W*. The VS's service capacity is achieved by peers' *resource aggregation* and will depend on the number of peers integrating this. The sum of peers' input (I_i) and output-bandwidth (O_i) will determine VS input and output-stream capacity.

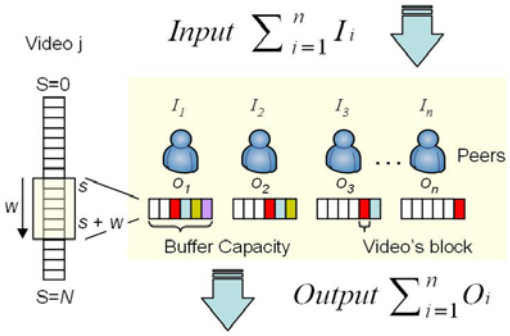

Fig. 1. DynaPeer Virtual Server

Initially, it is assumed that j is the video that all peers forming VS are reproducing. Video data available on VS is defined by *s* (first video block currently stored on VS) and the *collaboration window W* (period of time that any block remains stored on a VS before it is replaced). Outside [*s*, *s+W*], the interval defined by W, the VS is unable to make the collaboration as video data is not available in its buffer. Therefore, to provide full service for a streaming session, DynaPeer policies have to implement a *sliding window* over whole video. In this way, once the collaborative buffer is full, the following blocks received *(s+W, s+W+1,...)* replace the older blocks (*s, s+1,...*).

In DynaPeer, each VS is bound to an existent ongoing channel. Thus, the number of peers integrating a VS will depend on the peer's collaboration window and video request arrival-rate. To enlarge the collaboration window, we need to improve peers' buffer capacity (*B*). DynaPeer manages the peer's buffer by storing only data proportional to the contribution that can be carried out by peers' output-bandwidth (i.e, the video data kept for future collaboration for a video j with a play rate Pr_j, will be determined by Pr_j/O_i relation and buffer capacity). We term this strategy *extended buffer* capacity. The extended buffer allows VS to provide a larger collaboration window, increasing peers' collaboration probability and system efficiency.

The VS manages the collaborations by two different levels: full-stream and partial-stream collaboration. Full-stream collaboration is achieved when the VS has sufficient resources to deliver a full stream to a new client. In this case, the whole video stream will be delivered by the VS. Otherwise, if there are not enough resources, the VS proceeds with the partial stream collaboration. In this case, VS contributes with the new client request proportionally to their service capacity, and the server will be involved in the delivery process, sending data to the client in order to complete the service and to guarantee the QoS. Of course, every VS begins applying partial-stream collaboration and when it has sufficient size and resources, it switches to full-stream.

2.2 DynaPeer Unicast

Assuming that not all ISPs are powered with multicast technology in their access network, our proposal also exploits the delivery scheme by using unicast both from server and client side.

The mandatory condition for the collaboration process is that the requesting peer arrives inside the collaboration window W of the required VS. Following these conditions, if there are no candidate peers available for collaboration in a VS, the

server is responsible for opening a channel to serve the incoming request. If the number of peers inside a VS is not enough to take the collaboration, the Vs performs a partial stream collaboration. If there are sufficient candidate peers in the VS (enough resources) to generate a complete stream, a new channel will be opened from the peers to attend incoming request. All requesting peers, automatically, become candidate peers inside a new VS in the system.

Fig. 2a shows a snapshot of Unicast collaboration mechanism in minute six of DynaPeer stream process. In this example, we assume that a video stream must be served by three clients (Ni=3). The first peer (peer 1) is being directly attended by the server and it defines the collaboration window (W_1) for the VS_1. Peers 2 and 3 are also being attended by the server and both are integrated in VS_1. In minute three, the VS_1 has achieved its delivery capacity for one complete stream and when peer 4 makes a request in minute 4 (inside W_1) it is attended by VS_1 (Fig. 2a I), switching to full stream collaboration mode. Automatically, peer 4 starts another Virtual Server VS_2. As peer 5 arrives, the VS_1 does not have available service capacity to serve it. The VS_2 (composed by peer 4) applies partial stream collaboration with the server in the streaming process to peer 5 (fig. 2a II). The same occurs with peer 6 request.

2.3 DynaPeer Multicast

Using the multicast scheme, DynaPeer allows the streaming process for clients in a multi-source/multi-destination way, better exploiting the network capacity of ISPs.

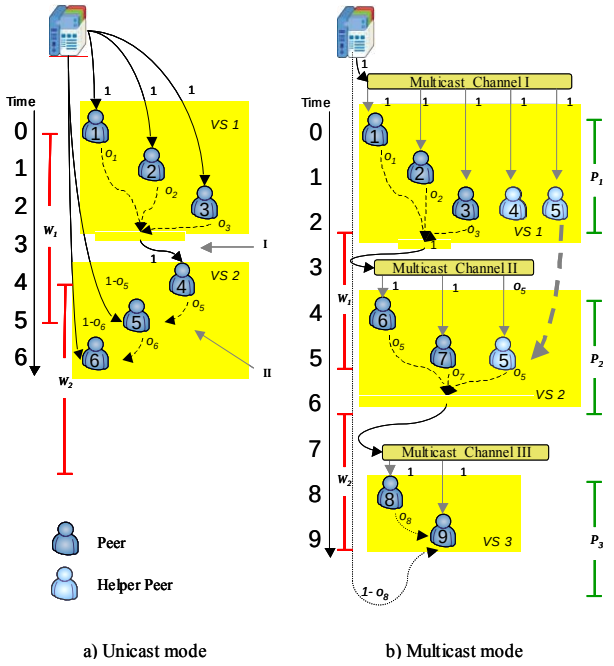

Fig. 2. DynaPeer Snapshot.

The VSs are responsible for creating multicasts channels, from the client's side[3], serving incoming client's requests. In this way, DynaPeer avoids any extra server's resource for serving contents that have already been started by other peers.

The collaboration process for multicast environment works by letting a new peer joining an ongoing multicast channel (*complete stream)* and still receives the entire video data stream. For new requests for the same video, the VS acts in two different ways: First, if an incoming peer can join an ongoing multicast channel, the VS delivers only the missing portion of the requested video in a separate unicast channel, *patch stream*, using the clients' output-bandwidth capacity. The period of time that a peer can join an ongoing multicast channel is called Patching Window (denoted as *P* time), and it depends on client buffer capacity. Second, if a requesting peer does not have sufficient buffer space for joining the ongoing channel (arrival time > *P*), the VS starts a new multicast channel for the incoming peer. Once patching window finishes, DynaPeer begins the multicast collaborative window, whose size depends on buffer available for collaboration after patching policy. VS only can create a new multicast channel if the next client request arrives inside the collaboration window. Different to unicast delivery, in multicast, the peers need their buffer to store patching information arising from the ongoing channel. Thus, extended buffer capacity for collaboration is more limited, since it can be applied only in the unused portion of the peer's buffer.

In multicast mode, the virtual server will be integrated by all the peers that arrive inside patching window. Therefore, depending on a video's popularity and on clients' requests rate, it is possible that the number of peers participating in a VS can be larger than N_i. In this case, as only N_i peers are required to propagate multicast stream, the remaining VS peers for most popular videos will not collaborate in the streaming process. On the other hand, less popular videos VS cannot have sufficient collaborators peers; consequently their service capacity cannot be sufficient to fulfill a complete streaming session in a collaboration process. We propose to use the idle peers on over-sized VSs to improve the QoS and performance of VoD system. In particular, we propose the utilization of those wasted peers to operate as *Helper peers*.

VS service capacity can be improved by the utilization of *Helper peers*. The main function of Helper peers is to allow the VS of non-popular videos to achieve full collaboration capacity, improving DynaPeer performance. *Helper peers* are allocated to collaborate with other VS without sufficient service capacity for carrying out a full stream collaboration. However, helper peers view another video and do not have the video data required to collaborate with a different VS. Therefore, to assist a VS, they previously need to receive video data, connecting Helpers in the ongoing channel of assisted VS. As a result, the Helper downloads video data, proportional to its output-bandwidth, stores it temporally on collaborative buffer and uses its output-capacity to delivery it to another client. The requisite for receiving the new video before serving it, will be wasteful unless the ingoing stream does not require additional resources. This constraint let this approach feasible only with multicast communications.

Fig. 2b shows a snapshot of the system in multicast configuration. Client arrival rates are shown in figures (time bar). Peer 1 has sent a video request to the server that has started a multicast channel to attend it. A few minutes later and inside P_1 time, clients 2, 3, 4 and 5 request video j. Theses clients were joined to multicast channel

[3] The mechanism of generating multicast trees from clients to other clients is orthogonal to the analysis presented in this article. For instance, we assume the mechanism proposed in [1].

and they are incorporated to VS_1. In time 2, patching window finishes and DynaPeer begins the multicast collaboration window (W_1). After P_1 time, but also inside W_1 time, peer 6 requests the same content j from the server. DynaPeer selects peers 1, 2 and 3 (N_i=3) to deliver the video and starts a new multicast channel (Channel II) for attending to the client's request. Once channel propagation is made, peer 4 and 5 is set as a helper peer. In time 5, peer 7 request video j. It arrives inside P_2 time and could be joined to multicast channel II. Due to time constraints, peer 8 request was unable to join either multicast channel I or II. The only possible alternative is to create a new channel. The VS_1 is unable to create this channel due to its collaboration window W_1 is surpassed by peer arrival time. Regardless that VS_2 was also incomplete in its total stream capacity to serve the request, it could achieve its completed service capacity by the utilization of VS_1 helper peer 5. At that moment, VS_2 could start the delivery process to the requesting peer, generating multicast channel III. Finally peer 9 arrives in minute 9, and it can join the ongoing multicast channel III.

3 Mathematic Analysis

In this section we present an analytical model for evaluate the DynaPeer performance. The main objective of our model is to evaluate the performance that can be achieved by DynaPeer and related P2P delivery policies. In this case, performance is understood as the server-load reduction (streams) due to request service distributed among peers (S*).

To perform this analysis some assumptions are made from points of view of architecture, clients and system work-load. The model assumes a VoD system with a single centralized server[4], which stores whole system catalogue. We take in consideration asymmetrical bandwidth behavior for clients. Moreover, this output-bandwidth is not enough to provide video at the required play-rate. Also, the model does not take in consideration clients' or servers' failures during a streaming session.

To undertake the model complexity, we do not handle dynamic behavior of the VoD system (network congestion and jitter, and variable client bandwidth) and we assume average values for clients' output-bandwidths (O) and clients' buffer capacity (B). Furthermore, we suppose that video is encoded with a Constant Bit-Rate and all videos' catalogue has the same length (L) and the same bandwidth requirements (Pr). Symbols used in the analysis are listed in Table 1.

Table 1. Analytical Model Main Parameters.

Symbol	Explanation	Symbol	Explanation
S*	Server Load (streams)	S^i_{C*}	Completed Streams for video i (streams)
M	Video catalog size	S^i_{P*}	Patch Streams for video i (streams)
L	Video Length (min)	W_i	P2P collaboration Window time
B	Peers mean Buffer size in minutes	λ_i	Requests arrival rate video i (req/min)
O	Peer Output Bandwidth (Mbps)	Pr_i	Play rate for Video i
N_i	Number of Peers for serve a stream for video i, $N_i = \frac{Pr_i}{O}$	G_i	Number of Collaborative Peers in a Group

[4] The model can also be directly applied for others architectures composed by multiple servers (Proxy or CDN based architectures).

In model development, we have evaluated the streaming capacity required by the server without the utilization of P2P policies as reference, and afterward we have evaluated the server load reduction resulting of the incorporation of DynaPeer schemes. Due to space limitations we only present DynaPeer multicast model.

3.2 Multicast Performance Analysis

We present two performance models for DynaPeer multicast. First we start by analyzing basic DynaPeer multicast delivery policy. Then we proceed by presenting DynaPeer with Helpers analytical model.

DynaPeer multicast.
In multicast, there are two server costs to evaluate, the full stream cost (complete stream created by the server for incoming clients) and the patch stream cost. Therefore, the total server-load for DynaPeer based policy is given by the sum of total server-load of complete streams and patch streams:

$$S_{DynMul} = \sum_{i=1}^{M} Sc^{i}_{DynMul} + Sp^{i}_{DynMul} \quad (1)$$

To achieve DynaPeer functionality, clients able to participate in a collaboration process are grouped inside a collaboration group (G_i). The collaboration group (G_i) is achieved by the total number of candidate peers arriving in the patching window time, which is made up of a VS:

$$G_i = \begin{cases} B * \lambda_i, & \frac{1}{\lambda_i} < B \\ 1, & \frac{1}{\lambda_i} \geq B \end{cases} \quad (2)$$

Once patching window has finished, DynaPeer begins the collaborative window (W), whose size depends on the buffer storage available for collaboration after patching policy. Peers' available buffer capacity depends on their relative arrival time inside the patching window. For modeling purposes, we assume the worst collaboration buffer depending on candidate peers inside the collaboration group of video i (G_i), which is multiplied by N_i in order to attain the extended collaboration window:

$$W_i = \begin{cases} B_i - \frac{N_i \cdot (N_i - 1)}{\lambda_i}, & G_i \geq N_i \\ B_i - \frac{N_i \cdot (G_i - 1)}{\lambda_i}, & G_i < N_i \end{cases} \quad (3)$$

Then, the full stream cost, for a video i, is achieved by calculating the number of channels that the server must open to serve this video during a period of time (L). In DynaPeer, if there are sufficient peers inside the collaboration group to propagate the requested video i ($G_i \geq N_i$), the server needs only to open first stream. Subsequent streams are managed by peers. Otherwise, peers can collaborate only partially or not

at all with the server (requiring the same streams as a central-server using patching policy, $S_{C^{i}Mul}$). Therefore the server-required stream is defines as follows:

$$Sc^{i}_{DynMul} = \begin{cases} 1, & G_i \geq N_i \\ Sc^{i}_{Mul} - \left(Sc^{i}_{Mul} * \frac{G_i}{N_i} \right), & Gi < N_i \quad and \quad \frac{1}{\lambda_i} \leq W_i \\ Sc^{i}_{Mul}, & otherwise \end{cases} \tag{4}$$

P2P patch streams service for video i is managed using the streaming resources from VS_{j-1} and VS_j (under construction) to serve the incoming client's request. If there is at least one peer in the current VS_j, it collaborates proportionally with the main server to send patch streams (Sp^{i}_{Mu}). The previous VS_{j-1} only helps if it has free streaming resources (G_i-N_i>0):

$$Sp^{i}_{DynMul} = \begin{cases} Sp^{i}_{Mul} - \left[\frac{Sp^{i}_{Mul}}{N_i} + \left(\left| G_i - N_i \right| * \frac{B}{N_i} \right) \right] * \frac{Sc^{i}_{Mul}}{L}, & G_i > 1 \\ Sp^{i}_{Mul}, & otherwise \end{cases} \tag{5}$$

DynaPeer multicast with Helpers.

Helpers are only used when there is at least one incoming request arriving inside the collaboration window time. Thus, the incoming request can take advantage of the virtual server created with helpers.

The number of available peers to perform Helper functionality (H_A) is achieved after collaboration is established. This is defined by the total number of peers inside a collaboration group (G_i) and that are not involved in the collaboration process:

$$H_A = \sum_{i=1}^{M} \left| G_i - N_i \right| \tag{6}$$

The number of requested Helpers to participate in a collaboration process for a video i is defined by the number of necessary peers to serve a stream, always provided that the collaboration group needs Helpers. Expr. 7 gives the number of requested helpers for video i.

$$H^{i}_{R} = \begin{cases} N_i, & \left(G_i < N_i \right) \wedge \left(\frac{1}{\lambda_i} < W_i \right) \\ 0, & otherwise \end{cases} \tag{7}$$

The number of available helpers are limited. Therefore, we have to decide to which Virtual Server the helpers will be assigned and control when helpers will be exhausted. To resolve the first issue, we assign helpers to those Virtual Servers that have fewer requisites. To control the number of available helpers, we use expr. 6 (available helpers) combined with expr. 8, that evaluates the total number of helpers required by first j more popular videos:

$$H_{TR}(j) = \sum_{i=1}^{j} H_R^i, \qquad \forall j \le M \tag{8}$$

Using helpers, the server requirements are of one stream only, provided there are sufficient helpers to complete the requisites of video i and also the previous ones ($H_{TR}(i)<H_A$), and that video-request arrival rates support collaboration ($1/\lambda_i<W_i$). If helpers can only partially fulfill the requirements, then only a portion of video streams will be saved by peers. Otherwise, the central server has to manage all streams:

$$Sc^i_{DynHlpMul} = \begin{cases} 1, & \left(H_{TR}(i) < H_A\right) \wedge \left(\frac{1}{\lambda_i} < W_i\right) \\ Sc^i_{DynMul} - \frac{H_A - H_{TR}(i-1)}{N_i}, & \left(H_{TR}(i-1) < H_A < H_{TR}(i)\right) \wedge \left(\frac{1}{\lambda_i} < W_i\right) \\ Sc^i_{DynMul}, & otherwise \end{cases} \tag{9}$$

4. Performance Evaluation

In this section, we show the analytical model results for the DynaPeer delivery scheme, by evaluating the performance contrasted with traditional Unicast and Patching delivery policies, and with other P2P delivery policies such as Promise[5], Chaining[7], and P^n2P^m[11].

4.1 Workload and Metrics

In our experiments, we assumed that inter-arrival time of client requests follows a Poisson arrival process with a mean of $1/\lambda$, where λ is the request rate. We used a Zipf-like distribution to model video popularity. The probability of the i^{th} most popular video being chosen is $1/(i^z \cdot \sum_{j=1}^{M} \frac{1}{j^z})$, where M is the catalogue size and *z* is the *skew* factor that adjusts the probability function. For the study, the *skew* factor is fixed to 0.729 (typical video-shop distribution [1]). The time of analysis was 90 minutes, the same as a video length, the output-bandwidth of clients is fixed to 750kbps and video play rate is set to 1500Kbps. The analyze values of the parameters are summarized in table 2.

The comparative evaluation is based on the server load metric that is defined as the mean number of streams required by the server at the end of analysis.

Table 2. Experimentation environment parameters.

Parameter	Default Value	Parameter	Default Value
Request Rate	10 requests/minute	Client's Buffer Size	15 minutes
Play rate	1500 Kbps	Client's Output Bandwidth	750 kbps
Video length	90 minutes	Video Catalogue Size	100 videos
Zipf Skew Factor	0.729		

4.2 P2P Delivery Schemes Comparison

Fig. 3 shows the server-load achieved by DynaPeer other P2P approaches. For this test, we have considered the analytical model described in section 3 and the analytical model found in each one of the delivery policies. As we would expect, server load for all P2P architectures is lower than that achieved by pure-Unicast and Patching delivery mechanisms.

Comparing unicast delivery policies, Promise achieves 50% less server-load while Chaining provides 70% compared with central unicast mode. DynaPeer-Unicast, achieves 50% of server-load reduction, as we find with Promise. The gap performance achieved by DynaPeer in unicast modes occurs because both Promise and Chaining have specific requirements for their functionality. Promise requires peers to have, at least, sufficient buffer capacity to store a whole video (90 minutes), while DynaPeer only assumes a buffer of 15 minutes. On the other hand, Chaining assumes that peer output bandwidth is, at least, the same of a video play-rate, while DynaPeer and Promise assume half the output bandwidth.

Comparing multicast mode, Promise is unable to take advantage of server-load reduction while Chaining keeps reducing server-load in the order of 39% compared with Patching's non-P2P policy. Results show that multicast usage, in conjunction with P2P scheme, provides the best solution for system performance. P^nP^m (that assumes that peer output bandwidth is the same as video play-rate) and *DynaPeer+Helpers* achieves the best server-load reduction, being 54% and 75% betters than Patching, and 24% and 59% if compared with Chaining. Finally *DynaPeer+Helpers* improve system performance by 45% and 74% if compared with P^nP^m and Promise, respectively.

5. Conclusions

We have proposed and evaluated a new delivery policy based on a P2P paradigm and multicast communication mechanism for Internet VoD services. Our design defines a Virtual Server, which is responsible for establishing a group of peers, enabling service

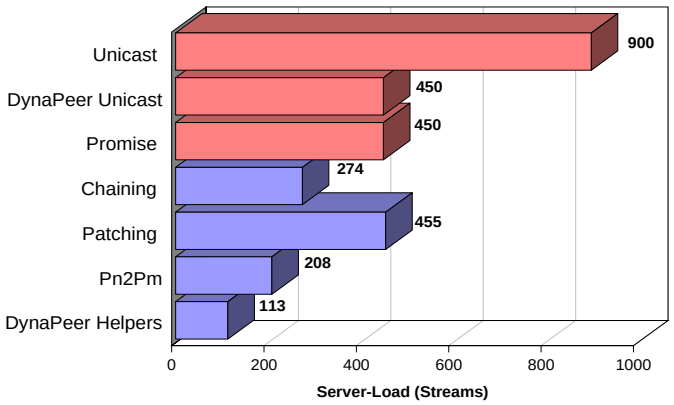

Fig. 3. DynaPeer Comparison

for new client requests by aggregating the necessary resources. The Virtual Server also performs distributed control tasks among peers by saving server control requirements.

The analytical study shows that DynaPeer policies improve VoD system capacity and decrease the server-load, taking major advantage of client resources to decentralize the delivery process. Compared with traditional unicast and patching delivery schemes and also with highly similar proposals found in the literature, we conclude that DynaPeer has the best performance.

We have started several future research projects. We are going to analyze the impact of Internet dynamic behavior on our schemes. In addition, we are studying a mechanism to automatically adapt DynaPeer to a heterogeneous environment and fault tolerance techniques. All these characteristics will be considered in future work, using simulation tools and a real prototype.

References

1. C.C. Aggarwal, J.L. Wolf, and P.S. Yu. The maximum factor queue length batching scheme for video-on-demand systems. IEEE Transactions on Computers, vol. 50, no. 2, pp. 97-109, 2001.
2. K. C. Almeroth, M.H. Ammar. "Multicast group behavior in the Internet's multicast backbone (MBone)". IEEE Communications Magazine, vol. 35, pages 124-129, June 1997.
3. Cisco Systems, "Delivering Multicast Video over Asymmetric Digital Subscriber Line". White Paper available in http://www.cisco.com/en/US/products/ps6598/products_white_paper 9186a00804f9d11.shtml, 2003
4. Y. Guo, K. Suh, J. Kurose and D. Towsley. "P2cast: peer-to-peer patching scheme for vod service". In Proceedings of the 12th Int. Conf. on World Wide Web, pp 301–309, ACM Press, 2003.
5. M. Hefeeda, A. Habib, B. Botev, D. Xu, and D. B. Bhargava. "PROMISE: Peer-to-peer media streaming using collectcast". In Proc. of ACM Multimedia' 03, Berkeley, CA, pages 45-54, 2003.
6. 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.
7. K. A. Hua, M. Tantaoui, and W. Tavanapong. "Video delivery technologies for large-scale deployment of multimedia applications", In Proc. of the IEEE, volume 92, September 2004.
8. S. Jin and A. Bestavros. "Cache-and-relay streaming media delivery for asynchronous clients". In Proceeding of NGC'02, Boston, MA, USA, October 2002.
9. V. Padmanabhan, H. Wang, P. Chou, and K. Sripanidkulchai. "Distributing streaming media content using cooperative networking". In Proc. NOSSDAV'02, Miami Beach, USA, 2002.
10. Xin Liu and Son T. Vuong. "A Cost-Effective Peer-to-Peer Architecture for Large-Scale On-Demand Media Streaming". Journal of Multimedia (JMM), Volume 1, Issue 2. May 2006.
11. 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 Proc. of 11th Int. Euro-Par 2005 Conference, Lisbon, Portugal, August 30-September 2005.