IMPROVING VOD P2P DELIVERY EFFICIENCY OVER INTERNET USING IDLE PEERS

Leandro Souza, Xiaoyuan Yang Javier Balladini, Ana Ripoll
Computer Architecture and Operating System, Universitat Autònoma de Barcelona, 08193 Bellaterra, Spain

Fernando Cores
Computer Science and Industrial Engineering, Universitat de Lleida, St. Jaume II, 69, 25001 Lleida, Spain

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

Abstract: This paper presents DynaPeer Chaining, a peer-to-peer Video-on-Demand (VoD) delivery policy designed to deal with high bandwidth requirement of multimedia contents and additional constraints imposed by Internet environment: higher delays and jitter, network congestion, non-symmetrical clients' bandwidth and inadequate support for multicast communications. We consider the scenario where we have multiple ADSL-based peers that stream the same video to multiple receivers. We propose an adaptive scheme to take advantage of idle peers in order to improve system efficiency, even when extreme conditions (low request rates or limited peer resources) are considered. We conducted a performance comparison study of our proposal with classic multicast (Patching) and other P2P delivery schemes, such as P^nP^n and Chaining, improving their performance by 50%, 62% respectively, even when taking into account Internet constraints.

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 multimedia 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 (Guo 2003, Hua 2004, Yang 2005) has been focused on the delivery process of multimedia content, 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 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 (Hua 2004), clients cache the most recently received video information in the buffer and forward it to the next clients using unicast channels. The P2Cast (Guo 2004) and cache-and-relay (Jin 2002) allow clients to forward the video data to more than one client, creating a delivery tree or ALM. However, neither Chaining nor 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 (Hefeeda 2003), CoopNet (Padmanabhan 2002) or BitVampire (Liu 2006) 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 to aggregate. 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 (Yang 2005) we proposed P^nP^n architecture that takes advantage of multicast technology on the client side. This architecture works by exploiting the

Souza L., Yang Javier Balladini X., Ripoll A. and Cores F. (2007).
IMPROVING VOD P2P DELIVERY EFFICIENCY OVER INTERNET USING IDLE PEERS.
In *Proceedings of the Second International Conference on Signal Processing and Multimedia Applications*, pages 293-300
DOI: 10.5220/0002137402930300

clients non-active resources in two ways: first, it allows clients to collaborate with the server in the delivery of initial portions of video, patch 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^n also requires that output bandwidth is 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 (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 have proposed a VoD architecture for an Internet environment, which is called P2PVoDSpread. In order to allow clients (peers) to collaborate with server-delivery process, we have designed a new delivery scheme called DynaPeer, based on a P2P paradigm. DynaPeer policy 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. 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 features founded on Internet, which allows us to design a realistic delivery scheme for VoD services.

In such P2P systems, peer collaboration depends on clients' free resources and the availability of requested multimedia content. If required content is not available, peers cannot then collaborate, and their resources will be wasted, becoming idle peers. In this paper, we focus on a multicast-delivery policy, DynaPeer Chaining, capable of taking advantage of idle peers in order to improve collaboration probability and reduce server load. This scheme allows the improvement of P2P system efficiency, even in the most restricted situations (with low request rates or few peer resources). Additionally, DynaPeer chaining can operate when peers' output-bandwidth is not so restricted, facilitating further performance improvement.

The remainder of this paper is organized as follows. In section 2 we present our system overview. In section 3, the performance evaluation is presented. In Section 4, we indicate the main conclusions and future works.

2 SYSTEM OVERVIEW

The goal of P2PVoDSpread design is to take advantage of client collaboration to decentralize the server-delivery process, eventually shifting the streaming load to peers. Clients make their idle-resources available so as to generate a complete, or partial, stream for incoming clients.

In this section, we first present details of P2PVoDSpread system, which is composed by three entities: VoD Servers, peers and Virtual Servers. Then we show how system operates, emphasizing on collaboration window concept, extended buffer strategy and DynaPeer service schemes.

2.1 System Entities and Concepts

2.1.1 VoD Server

The P2PVoDSpread 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, guarantying the QoS[1].

2.1.2 Peers

In our system, a peer can be a computer or a set-top-box, interconnected by ISPs network. It 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 have to collaborate in order to provide service for a complete streaming session. The number of

[1] In the rest of paper, the system description is done from point of view of a VoD system with a single centralized server, which stores whole system catalogue. However, the system design can also be directly applied for others architectures composed by multiple servers (Proxy or CDN based architectures).

necessary peers to start a stream process to a video i is denoted as N_i, and is defined by the ratio between video i play-rate and peers' output bandwidth. 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.

2.1.3 Virtual Server

All collaborations in DynaPeer are managed by the Virtual Server (VS). A virtual server (Fig. 1), denoted by VS(*i,s,w*), is a logical entity defined as a set of peers that collaborate in a delivery process to offset s of video i, 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 (*Ij*) and output-bandwidth (*Oj*) will determine VS input and output streaming capacity.

Initially, it is assumed that i 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*. Outside [s, s+W], the interval defined by the collaboration window, 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 data. In this way, once the collaborative buffer is full, the following blocks received (s+W, s+W+1, s+W+2,...) replace the oldest blocks (s, s+1, s+2,...).

2.1.4 Extended Buffer

To take advantage on clients' buffer capacity, DynaPeer are configured to set clients' buffer to store proportional information of video data that can be carried out by peers' output-bandwidth (O_j). In this way, peers does not need to store the complete minutes of video, they are coordinated to store different blocks of it (i.e., the data kept for future collaboration for a video i with a play rate Pr_i, will be determined by Pr_i/O_j relation and buffer capacity B). We term this strategy as extended buffer. Expr. 1 determines the extended buffer size for a peer j (B_j).

$$B_j = \begin{cases} B * \frac{\Pr_i}{O_i}, & O_i < \Pr_i \\ B, & otherwise \end{cases} \quad (1)$$

The extended buffer allows peers to store only necessary data for collaboration exploiting efficiently its buffer capacity.

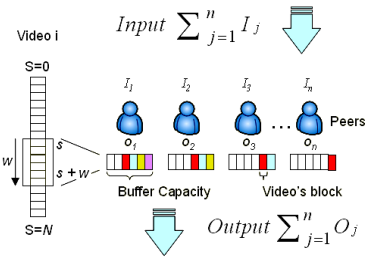

Figure 1: DynaPeer Virtual Server.

2.1.5 Collaboration Window

The collaboration window (W_i) is defined as the extended buffer time capacity of all peers involved in a collaboration process. Due stream patches requirements 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[2]. The W_i is calculated by expression 2.

$$W_i = \begin{cases} B_i - \frac{N_i \cdot (N_i - 1)}{\lambda_i}, & B * \lambda_i \geq \frac{\Pr_i}{O_i} \\ B_i - \frac{N_i \cdot (B * \lambda_i - 1)}{\lambda_i}, & Otherwise \end{cases} \quad (2)$$

The collaboration window defines the maximum time that an incoming request can be served by peers (i.e., period of time that any block remains stored on collaborators' buffer before its replacement). In this way, the mandatory condition for the collaboration process is that the requesting peer arrives inside the collaboration window.

2.2 System Operationally

DynaPeer operates using Virtual Servers to manage clients' collaborations. Each Virtual Server is bound to an existent ongoing channel and all requesting peers for this channel, automatically, become candidate peers inside a new VS in the system.

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

[2] For modeling purposes, we assume the worst collaboration buffer depending on number of involved peers in stream process

proportionally to their service capacity and the server will be involved in the delivery process, sending data to the requesting peer 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 peers and resources, it switches to full-stream mode.

2.2.1 DynaPeer Multicast

Using the multicast policy, DynaPeer allows the streaming process for clients in a multi-source/multi-destination way, better exploiting the network capacity of ISPs. 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 works by allowing a new peer joining an ongoing multicast channel (complete stream) still receiving the entire video data stream. For new requests for the same video, the Virtual Server acts in two different ways: First, if an incoming peer can join an ongoing multicast channel, the server 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 (detonated 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 (W), whose size depends on buffer storage available for collaboration after patching policy. VS only can create a new multicast channel if the next client request arrives inside the collaboration window.

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 VS service capacity cannot be sufficient to fulfill a complete streaming session.

We propose to use the idle peers from over-sized VS to improve the QoS and performance of VoD system. In particular, we propose the utilization of those wasted peers to operate in one of these two ways: as Backup peers or as Helper peers.

The main goal of a Backup peer is to guarantee the necessary *QoS* during a streaming session started by VS. The Backup peer draws an important rule for dealing with peer failures and high Internet network jitter. In this way, Backup peers are used in VS as support for active collaborators' peers. If any peers go down during a streaming session, the Backup peer will be responsible for replacing it. In addition, due to bandwidth variations, an active peer inside a VS, can have its output-bandwidth capacity reduced. In this case the VS can use the Backup peer to help on delivery process in order to supply the necessary quality of service for streaming session.

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 full stream collaboration. However, Helper peers view another video and do not have the video data required to collaborate with different VS. Therefore, to assist a VS, they previously need to receive video data, connecting the Helper peer in the ongoing channel of assisted VS. As result, the Helper peer 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 data 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.

The number of available peers to perform Helper functionality (H_A) is achieved after collaboration is established. This is defined multiplying the total number of candidate peers inside a collaboration group ($G_i=B*\lambda_i$) and that are not involved in the collaboration process (N_i) by number of virtual servers for video i (VS_i). Expression 3 synthesizes the number of available helpers in the system, where M is the total number of videos in system's catalog.

[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 (Cheng, K. 2005)

$$H_A = \sum_{i \in D} ((G_i - N_i - 1) \cdot Cs_i) + \sum_{i \in D'} (G_i \cdot Cs)$$

$$D = \{ j \mid 1 \le j \le M \ \ and \ \ G_j > N_j \}$$

$$D' = \left\{ j \mid 1 \le j \le M \ \ and \ \ G_j < N_j \ \ and \ \ \frac{1}{\lambda_j} \ge W_j \right\} \quad (3)$$

The number of necessary Helpers to participate in a collaboration process for a video i is defined by the number of requested peers to complete VS capacity (N_i–G_i), multiplied by the number of complete streams generated for this video, always provided that the collaboration group needs Helpers peers (G_i<N_i). Expression 4 gives the number of requested helpers for a video i.

$$H_R^i = \begin{cases} (N_i - G_i) \cdot Cs_i, & (G_i < N_i) \wedge \left(\frac{1}{\lambda_i} < W_i \right) \\ 0, & otherwise \end{cases} \quad (4)$$

The number of available helpers is limited. Therefore, we have to decide to which VS the helpers will be assigned and control when helpers will be exhausted. To resolve the first issue, we assign helpers to those VSs that have fewer requisites to accomplish with a complete stream capacity, To control the number of available helpers, we use expr. 3 (available helpers) combined with expr. 5 that evaluates the total number of helper peers required by first k videos (more popular):

$$H_{TR}(k) = \sum_{i=1}^{k} H_R^i, \qquad \forall j \le M \quad (5)$$

Fig. 2 shows a snapshot of the system in multicast configuration. Client arrival rates are shown in figure (time bar). Peer 1 has sent a video i request to the server that has started a multicast channel to attend it. A few minutes later and inside P_1 time, peers 2, 3, 4 and 5 request the same video. Theses clients were joined to multicast channel I and they are incorporated to VS1. 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 i 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 the request. Once channel propagation is made, peer 4 is set as Backup peer for VS1 whilst peer 5 is set as a Helper peer.

In time 5, peer 7 requests video i. It arrives inside P2 time and could be joined to multicast channel II. In time 8, due 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

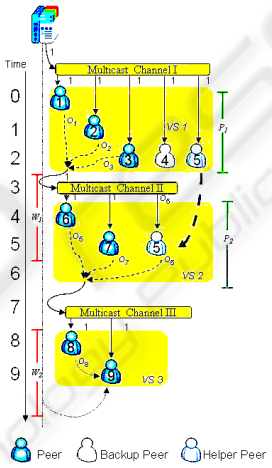

Figure 2: DynaPeer Multicast Snapshot.

2.2.2 DynaPeer Chaining

In DynaPeer, the use of Helpers allows idle clients to participate in the collaboration process. This contribution is useful in providing full stream capacity to a VS, avoiding server participation. However, depending on request load, not all helpers can be used by the DynaPeer policy to collaborate with Virtual Servers. In Fig. 3, we show the number of total helpers, calculated using expr. 3, and the remaining free helpers after applying DynaPeer Multicast (expr. 5). As can be seen, there is an important volume of free helpers, even with low request rates. The main idea behind DynaPeer Chaining is that those free Helpers that are also idle in the system are grouped to generate a new Virtual Server in order to propagate video information for a new period of time. This propagation can be understood as an extension of a previous VS collaboration window. Thus, the VS of popular videos does not need such an extension; in contrast, however, the VS of unpopular videos does. The collaboration process works identically to DynaPeer Multicast. The main difference in this approach is

that there will be special Chain VSs composed only of Helpers peers.

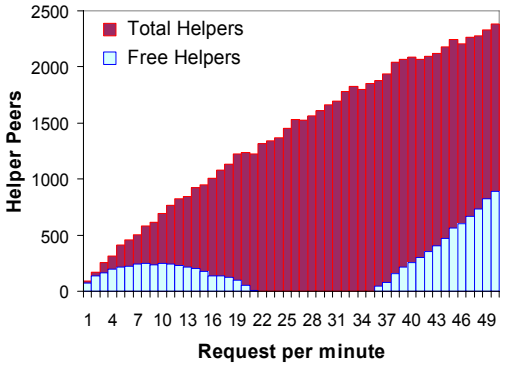

Figure 3: Free helpers after applying DynaPeer.

Fig. 4 shows a snapshot at minute 12 of DynaPeer Chaining for an unpopular video. We have determined that the number of collaborative clients to attend a full stream capacity is equal to 3. Furthermore, client buffer are constant and sufficient to generate two minutes of collaboration window (W_1, W_2 and W_3). There are two requests for this video, the first starting at minute 0 (peer 1) and the last at minute 12 (peer 2). When peer 1 makes its request, a new virtual server is created (VS_1) and three Helper peers are designated to it, in order to guarantee full stream capacity (DynaPeer Helpers). However, DynaPeer Chaining detects that requested-video is set as 'unpopular', and schedules new helpers to create a new VS at the final time of the VS_1 collaboration window (W_1). This process creates VS_2. This new VS will propagate the video for a longer W_2 period of time. As no requests are received in this period, DynaPeer Chaining maintains its function, generating another new VS (VS_3) to once again propagate the video data for future requests. Finally, the request from client 2 arrives at minute 12, being attended by VS_3.

We notice that, in this example, DynaPeer Chaining needed 6 Helpers to attend the client's request (peer 2). Thus, the number of Helpers for this purpose will depend on the number of available Helpers in the system, which is given by the number of requests for the most popular videos. Thus, with DynaPeer Chaining, future clients requesting non-popular videos will have a greater probability of being attended by a VS, alleviating Server load.

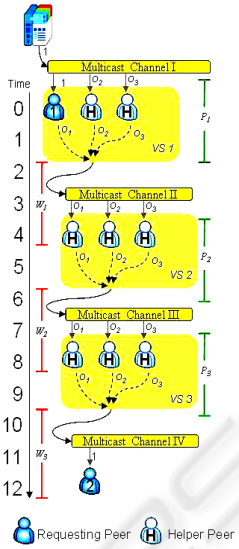

Figure 4: DynaPeer Chaining Snapshot.

3 PERFORMANCE EVALUATION

In this section, we show the analytical model results for the DynaPeer delivery scheme (Souza, 2007), starting by evaluating the performance contrasted with Patching delivery policy and other P2P-based delivery policies. The goal of this experimentation is to analyze DynaPeer Chaining performance andscalability with respect to different workloads.

The 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. We have evaluated this metric over different workloads (requests rate) and client's resources (client-buffer sizes and output bandwidth). These parameters are related with each other and affect the peer-collaboration capacity

3.1 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/λ, where λ is the request rate. We used a Zipf-like distribution to model video popularity. The probability of the ith most popular video being chosen is $1/(i^z.\sum_{j=1}^{M}\frac{1}{j^z})$, where M is the catalogue size and z is the skew factor that adjusts the probability function. For the whole study, the skew factor is fixed to 0.729 (typical video-shop distribution, Hongliang 2006). The time of analysis

and video length was 90 minutes, the output-bandwidth of clients is fixed to 750Kbps and video play rate is 1500Kbps. The analyzed and default values of the parameters are summarized in table 1.

Table 1: Experimentation environment parameters.

Parameter	Default value	Analyzed values
Zipf Skew Factor	0.729	0.729
Video length	90 minutes	90 minutes
Video Catalogue Size	100 videos	100 videos
Client's Output Bandwidth	750 Kbps	50 – 2500 Kbps
Request Rate	10 request/min	2-50 request/min
Client's Buffer Size	5 minutes	1-50 minutes
Play rate	1500 Kbps	1500 Kbps

3.2 Client-request Rate Effect

In this experiment, we have changed client request rate from 2 to 50 requests per minute. The other system parameters are assumed to have the default values shown in table 1.

Fig. 5 shows that DynaPeer Chaining performs DynaPeer Multicast when system load is lesser than 21 req/min. The improvement depends on system load, but on average it achieves a server load reduction of 14%. After this load there are not free helpers for applying DynaPeer Chaining, however, this fact not impact policy performance, achieving the same results than DynaPeer Multicast.

DynaPeer policies are capable of accomplishing the optimal performance of 100 complete streams (by taking in account initial seed video streams required for each catalogue video) + patch streams, with only a request rate of 37 req/min.

Comparing the performance of DynaPeer policies with other approaches, we can noticed than as the requests increases, the amount of available resource of clients also increases, which provides a lower server-load for DynaPeer policies. DynaPeer Chaining policy improves Patching requirements by up to 78%, whilst surpass P^nP^n and Chaining by 50% and 62% respectively.

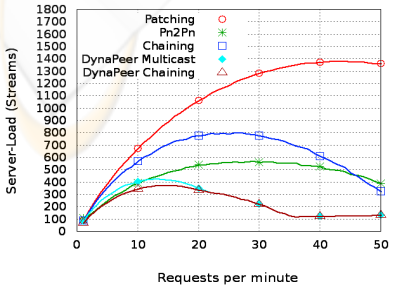

Figure 5: System Performance vs request rate.

Analyzing the results, we can conclude that DynaPeer Chaining policy provides an unlimited scalability, due to it is capable to hold server load in spite of however the request-rate may have grown.

3.3 Client-buffer Size Effect

The goal of this experimentation is to evaluate the influence of peer storage resources (buffer size) on delivery policies performance. To evaluate this, we have changed client buffer size from 1 to 50 minutes, showing the server-load achieved.

In figure 6, we can perceive that client's buffer size have an important impact on system performance. The server-load of all policies decreases in accordance with client-buffer size, due to larger buffer capacity improves sharing capabilities among requests. The results show that also DynaPeer Multicast and DynaPeer Chaining policies are able to achieve better buffer efficiency than the others. This can be shown by swift convergence to optimal performance obtained by DynaPeer policies. DynaPeer achieves lower server requirements with a buffer capacity of 14 minutes. Patching achieves its best performance (320 streams) with a buffer capacity for 28 minutes of video, meanwhile chaining achieves at minute 27.

3.4 Output-Bandwidth Size Effect

In this section, we evaluate the delivery policy's performance in accordance with client output-bandwidth. This parameter only has influence over DynaPeer policies; due to the other approaches do not consider client output-bandwidth limitation.

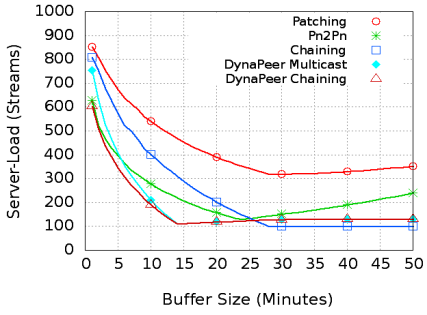

Figure 6: System Performance vs clients' buffer.

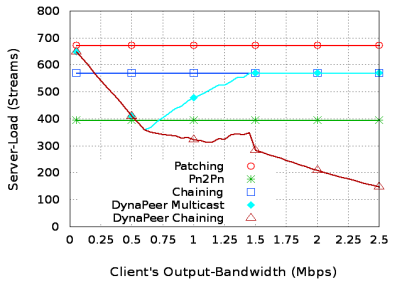

Figure 7: System Performance vs Output-Bandwidth.

Analyzing the figure 7, DynaPeer approaches have different behaviors. DynaPeer Multicast first decreases in server-load; when peer output-bandwidth is over 600 Kbps, it begins to lose performance and so the server-load increases. This is caused by the extended buffer definition, which means higher output-bandwidth creating a lower extended buffer capacity, and consequently resulting in lower performance. In this case, better results are achieved when a tradeoff between extended Buffer capacity and output bandwidth is employed.

On the other hand, DynaPeer Chaining keeps alleviating server-load due to helpers' utilization. This fact can be possible due the creation of chains of helpers that can provide a collaboration window as big as necessary for collaboration. We can infer that DynaPeer Chaining provides extra capabilities to adapt P2PVoDSpread system to a heterogeneous environment. Furthermore, it can take advantage of additional peer resources (i.e. output bandwidth), when they are available, to enhance scalability and performance.

4 CONCLUSIONS

Our concern in this paper is a new VoD system based on a P2P paradigm and multicast communication for Internet VoD services. Instead of independent collaborations between server and client, the proposed DynaPeer and DynaPeer Chaining delivery policies synchronize a group of clients for collaboration to attend new requests, reducing server-resource requirements.

The experimental study with analytical models shows that DynaPeer policies improve VoD system capacity and decrease the server-load, taking major advantage of client resources to decentralize the delivery process. Experimental results have shown that DynaPeer performance depends directly on the number of available collaborators and their resources. DynaPeer Chaining can make use of idle peers to improve system efficiency, even when extreme conditions (low request rate or limited peer resources) are considered. Furthermore, it can take advantage of additional peer resources, to enhance P2PVoDSpread scalability and performance in an heterogeneous environment.

We are extending DynaPeer system, analyzing the impact of dynamic behavior of Internet on our delivery schemes. We are focused on scheduling and peer selection policies capable to itself to a heterogeneous environment. Second, we are working on a dynamic and distributed control mechanism to provide fault-tolerance functionality. All these characteristics will be considered in future work, using simulation tools and a real prototype.

REFERENCES

Cheng, Kan-L., Cheuk K.-W and Chan S.-H.G., 2005. Implementation and performance measurement of an island multicast protocol. *IEEE Int. Conf. on Communications (ICC 2005), vol. 2, pp 1299-1303.*

Guo, Y., Suh, K., Kurose J. and Towsley, D.2003. P2cast: peer-to-peer patching scheme for vod service. *In Proc. of 12th Int. Conf. on World Wide Web, pp 301–309.*

Guo, L., Chen, S., Ren, S., Chen, X. and Jiang, S. 2004. PROP: A Scalable and Reliable P2P Assisted Proxy Streaming System. *In Proc. of the 24th Int. Conf. on Distributed Computing Systems (ICDCS'04), pp. 778-786, Washington, DC.*

Hefeeda, M., Habib, A., Botev, B., Xu, B., and Bhargava, D. B. 2003. PROMISE: Peer-to-peer media streaming using collectcast. *In Proc. of ACM Multimedia' 03, Berkeley, CA, pages 45-54.*

Hongliang Y., Dongdong Z., Ben Y. Z., Weimin Z. 2006. Understanding user behavior in large-scale video-on-demand systems *In Proc. of the 2006 ACM Eurosys Conf., Leuven, Belgium.*

Hua, K. A., Tantaoui, M., and Tavanapong, W. 2004. Video delivery technologies for large-scale deployment of multimedia applications. *In Proc. of the IEEE, volume 92.*

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

Liu, X. and Vuong, S. T. 2006. A Cost-Effective Peer-to-Peer Architecture for Large-Scale On-Demand Media Streaming. *Journal of Multimedia, Vol. 1, Issue 2.*

Padmanabhan, V., Wang, H., Chou, H., and Sripanidkulchai, K. 2002. Distributing streaming media content using cooperative networking. *In Proc. NOSSDAV'02, Miami Beach, USA.*

Souza, L., 2007. DynaPeer: A Dynamic Peer-to-Peer VoD System over Internet. PhD thesis, University Autonoma of Barcelona.

Yang, X. Y., Hernández, P., Cores, F., Ripoll A., Suppi R. and Luque, E. 2005. Dynamic Distributed Collaborative Merging Policy to Optimize the Multicasting Delivery Scheme. *In Proc. of 11th Int. Euro-Par 2005 Conf., Lisbon, Portugal.*