A Pull-Based Approach for a VoD Service in P2P Networks

Anwar Al Hamra, Ernst W. Biersack, Guillaume Urvoy-Keller

Institut Eurécom
B.P. 193 - 06904 Sophia Antipolis - FRANCE
{alhamra, erbi, urvoy}@eurecom.fr

Abstract. We study a new approach to provide an efficient VoD service to a large client population in P2P networks. Previous work has suggested to construct a multicast tree to distribute the video to clients. However, a multicast tree usually requires the central server to perform complex algorithms to add new clients to the tree. In this paper, we show how to simplify the algorithm performed by the server and, at the same time, achieve an efficient use of the system resources. For this purpose, we present a new pull-based approach, called PBA. The basic idea of PBA is quite simple. When a new client wishes to receive a video, the new client contacts first the server. If there is enough left over bandwidth along the path to the new client, the server transmits the video to the new client. Otherwise, the server provides to the new client a list of candidate servants chosen at random. These servants are clients that have received or are currently receiving the video. The new client then searches for an available servant to download the video from.
We compare PBA to P^2cast, a multicast-tree based approach proposed previously. We investigate via intensive simulations the efficiency of both approaches in terms of percentage of rejected clients and of bandwidth consumption. PBA does not only simplify the algorithm performed at the server, but also consumes less bandwidth and allows more clients to access the service.
keywords: Video on-demand service, Peer-to-peer networks, Multicast tree, Performance evaluation

1 Introduction

Providing a VoD service to a large client population over the Internet requires efficient and scalable architectures. Many systems have been explored in the past to tackle scalability for VoD services. These systems largely fall into open-loop systems [1, 2, 3, 4], closed-loop systems [5, 6], and prefix caching assisted periodic broadcast systems [7, 8]. Yet, in the above systems, clients are passive in the sense that they only receive the video and do not anticipate resources. A new paradigm, called Peer-to-Peer (P2P), has emerged in the past few years. In P2P systems, clients do not only receive the content but they also provide resources by serving other clients (peers). Thus, when the arrival rate of clients is high, there are more clients that contribute to resources, which in turn increases the overall capacity of the system and therefore the number of clients that can be served.

Previous work has advocated to construct a multicast tree to deliver the video in P2P systems. However, a multicast tree usually requires the server to perform complex algorithms to allow new arrivals to join the tree. In this paper, our goal is to show that we can achieve both, a simple server algorithm and an efficient resources management. For this purpose, we introduce a new pull-based approach, called PBA. The basic idea of PBA is quite simple. When a new client wishes to receive a video, the new client contacts first the server. If the server is not overloaded and there is enough left over bandwidth along the path to that new client, the server feeds the new client with the video. Otherwise, the server responds to the new client with a list of servants (peers) chosen at random. These servants are clients that have received or currently receiving the video. The new client then searches for an available servant from which it downloads the video.

We compare PBA to P^2cast [9], a multicast-tree based approach proposed previously. We investigate via intensive simulations the efficiency of PBA and P^2cast. Our conclusion is that PBA does not only simplify the server as compared to a multicast-tree based approach, but it also consumes less bandwidth, which allows more clients to be serviced. The rest of this paper is organized as follows. Section 2 presents related work. Section 3 introduces PBA model. Section 4 presents the simulation settings. Section 5 provides results. Section 6 addresses service disruptions. Section 7 concludes the paper.

2 Related Work

Previous work on VoD can be largely classified into open-loop schemes, closed-loop schemes, and prefix-assisted periodic broadcast schemes.

Concerning open-loop schemes, *Staggered broadcasting* [1] is the most straightforward one where the server allocates for each video C channels each with a bandwidth equal to the playback rate b of the video. On each channel, the whole video is broadcast at rate b. The starting points of the transmission on the different channels are shifted in time to guarantee a start-up delay of no more than L/C, where L is the length of the video.

More efficient and complicated schemes have been proposed later on [2, 3, 4]. The key idea is that each video is divided into many segments. The server transmits each segment periodically and infinitely each at its own rate. These schemes differ in the way they set the rate and the size of each segment. For more details on open-loop schemes see [10].

While open-loop schemes broadcast the video regardless of the clients request pattern, closed-loop schemes serve the video in response to clients requests. Hierarchical merging [6] is the most efficient closed-loop scheme. As its name indicates, this scheme merges clients in a hierarchical manner. When a new client arrives, the server initiates an unicast stream to that client. At the same time, the client starts listening to the closest (in time) stream (target) that is still active. When the client receives via unicast all what it missed in the target stream, the unicast stream is terminated and the client listens only to the target stream, and the process repeats.

On the other hand, prefix caching assisted periodic broadcast schemes [7, 8] combine open-loop and closed-loop systems. They divide the video into two parts, the prefix and the suffix. The prefix is delivered via a closed-loop scheme while the suffix is multicast via an open-loop scheme. This combination makes these schemes suitable for both popular and non-popular videos.

In P2P environments, most of the existing approaches relie on constructing and maintaining a multicast tree rooted at the server [11, 12, 13, 14, 9]. P^2cast [9] belongs to this family of approaches. It applies multicast controlled threshold [5] to P2P systems. It divides each video into two parts, the patch and the suffix. The patch is delivered via unicast while the suffix is delivered through a multicast tree rooted at the server. P^2cast will be more detailed in section 4.

3 PBA: A Pull-Based Approach for a VoD Service

PBA's main advantage is to keep the server as simple as possible. When a new client wants to receive a video, it first contacts the server. In case there is enough available bandwidth along the path between the server and the new client, the server unicasts the video directly to the new client. In contrast, in case there is not enough available bandwidth, the server responds to the new client with the IP addresses of N_s servants chosen at random. The new client then tries to find an available servant to download the video from it via unicast.

Actually, estimating the available bandwidth between two end hosts in the Internet is not an easy task. Moreover, the available bandwidth of a connection might fluctuate over time. In this paper we do not address these issues. Our goal here is to show that as compared to a multicast tree-based approach, a pull-based approach simplifies the server and saves bandwidth resources. In addition, assuming a static network allows us to better understand the basic behavior of PBA (and P^2cast later). Hence, we assume that (i) Each connection has a transfer rate equal to the playback rate b of the video and (ii) A connection is set up (and can not be interrupted) in case the left over bandwidth along the path of that connection is larger than or equal to b.

3.1 The Server Task

When the server receives a request for a video from a new client, it estimates the available bandwidth along the path to that new client. In case the estimated available bandwidth is larger than or equal to the playback rate b of the video, the server unicasts the video to the new client. Otherwise, the server responds to the new client with the IP addresses of N_s servants. The server chooses at random N_s servants amongst active clients (peers) in the network.

3.2 The New Client Task

A new client first sends a request to the server to retrieve a video. If the available bandwidth along the path between the server and the new client is less than b, the server provides the new client with the IP addresses of N_s servants chosen at random. The servants are clients that have downloaded or are currently downloading the video. The new client then tries to improve this initial list of servants by accounting for local servants, i.e. servants that are located in its local area network. The motivation is that, retrieving the video from local clients saves bandwidth in the core of the network, which allows more clients to access the service. To discover local servants, the new client broadcasts locally a message. Each client that receives this message responds to the new client indicating its arrival time. We denote by N_{ls} the number of local servants found. If N_{ls} is larger than N_s, the new client replaces its initial list of servants by the N_s most recent local servants. In contrast, if N_{ls} is less than N_s, the new client replaces at random N_{ls} servants in its initial list by the N_{ls} local ones. Afterwards, the new client searches for an available servant to download the video. The new client contacts its servants starting with the local ones. If none of them is available, it contacts at random one of the non local ones. The new client keeps on contacting its servants until (i) A servant accepts to serve it. In this case, the new client joins the network and sends a message to notify the server. (ii) None of the candidate servants accepts to serve the new client. In this case, the new client is rejected.

3.3 The Servant Task

When a servant receives a request for a video from a new client, the servant estimates the available bandwidth to the new client. If the available bandwidth is larger than or equal to b, the servant unicasts immediately the video to the new client. Otherwise, the servant rejects the request.

In case the servant is behind a firewall, the servant can push the video towards the new client. However, the client can not get the video in case both, the servant and the new client are behind firewalls. To avoid such a scenario, we can require the server to choose an appropriate list of servants for the new client. Also, one might think of having intermediate proxies that allow two peers to communicate. This will be investigated more in future work.

4 Performance Evaluation

Constructing a multicast tree to distribute the video to clients has been advocated in previous work. Such approaches usually require the server to run complex algorithms to construct the multicast tree and, most often, do not account for the physical distance between the client and its servant. Our goal in this paper is to show that we can simplify the algorithm performed at the server and, at the same time, use less system resources. To this purpose, we will compare our pull-based approach PBA to P^2cast, a multicast-tree based approach.

4.1 Overview of P^{2}cast

P^2cast [9] applies multicast controlled threshold [5] to P2P systems. The first client that requests the video receives a complete stream for the whole video from the server. A new client that arrives within a threshold T_{cast} after the first client connects to that complete stream that changes from unicast to multicast. In addition, the new client receives immediately from the server a unicast patch for the part it missed in the complete stream due to its late arrival. On the other hand, the next client that arrives later than T_{cast} after the first client, it receives a new complete stream and new clients are patched again within the same new complete stream instead of the old one until the threshold time passes again and the same process is repeated.

Hence, P^2cast divides each video into a patch and a suffix. The patch is delivered via unicast while the suffix is delivered on a multicast tree rooted at the server[1]. When a new client nc requests a video, nc contacts a client P starting from the server. P estimates its bandwidth $BW(P, nc)$ to nc. Meanwhile, P asks its child to estimate their bandwidth to nc. The child that has the largest bandwidth $BW(cmax, nc)$ to nc amongst all child is denoted by $cmax$. In case of tie, the closest child is chosen. P compares $BW(P, nc)$ to $BW(cmax, nc)$ and three scenarios are then possible:

– $BW(P, nc) > BW(cmax, nc)$ and $BW(P, nc) \geq 2b$. Then, nc receives the patch directly from P and joins the multicast tree for the suffix as a child of P.
– $BW(P, nc) > BW(cmax, nc)$ and $2b > BW(P, nc) \geq b$. In this case, P delivers the patch to nc and forwards the request for the suffix stream to the child $cmax$. $cmax$ then estimates its bandwidth to nc and asks its child to estimate their bandwidth to nc and the same process is repeated.
– $[BW(P, nc) > BW(cmax, nc)$ and $BW(P, nc) < b]$ or $[BW(P, nc) < BW(cmax, nc)]$. Then, P forwards the request to $cmax$ and the process above is repeated.

4.2 Simulation Settings

In this study, we consider the case of a single video of length L. However, the study can be easily generalized to the multiple videos case. In P^2cast, clients that share the same multicast stream form a session. Sessions are completely independents and there are no interactions between clients from different sessions. In contrast, PBA has no such constraint and a new client can retrieve the video from any other client. However, to make a fair comparison between PBA and P^2cast, we should take into account the session limitation of P^2cast. Therefore, we introduce for PBA a threshold, T_{pba}, to limit the interaction between clients. Suppose that client C_1 has joined the network at time t_0. C_1 can download the video only from clients that arrived between time $t_0 - T_{pba}$ and t_0.

In the following, we will assume that clients request the video from the beginning to the end (no early departure) and that there are no service disruptions. In fact, service disruptions represent a main challenge for P2P systems. in section 6, we discuss how to extend PBA to deal with service disruptions. However, in the performance evaluation, we do not concern with these issues. The reason is that we are interested in evaluating the bandwidth consumption and the percentage of rejected clients for a multicast tree-based approach and a pull-based approach. Hence, in this comparison, we assume that a client leaves the system only if (i) It has completely received the video and (ii) It is not a servant for any other client. Thereby, the maximum amount of time spent by a client

[1] As we will see later, a P^2cast client helps delivering both, the patch and the suffix.

in the network is given by:

$$T_s = \begin{cases} max(L, 2 \cdot T_{cast}) & \text{for } P^2cast \\ L + T_{pba} & \text{for } PBA \end{cases} \quad (1)$$

We choose the values of T_{pba} and T_{cast} in such a way that T_s is the same for both approaches. In this comparison, we will consider the same environment as in the performance evaluation of P^2cast in [9]. Thus, we consider the network of 100 nodes[2] (figure 1). The network has been produced by GT-ITM [15]. It consists of two levels,

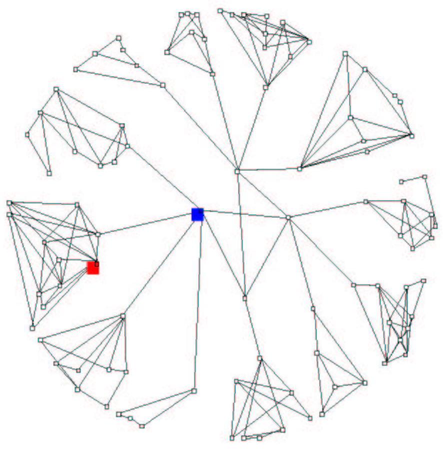


Fig. 1. A 100 nodes network topology

the transit-domain level and the stub-domain level. We assume that each stub-domain node is an abstraction of a local area network. However, we limit the number of streams inside each local network to L_{bw}. For instance, consider a local network A for a stub-domain node. At the same time, the maximum number of connections that have at least one of its two ends located in A can not exceed L_{bw}. Each connection has a transfer rate of b. We set the capacity of each link between two transit-domain nodes (BW_{tt}) or between a transit-domain node and a stub-domain node (BW_{ts}) to $BW_{tt} = BW_{ts} = 20b$. This means that, at the same time, such a link can support up to 20 streams each of rate b. For a link between two stub-domain nodes, we set its capacity to $BW_{ss} = 5b$.

We assume that clients arrive to the network according to a Poisson process with a total aggregate rate λ. Clients connect only to stub-domain nodes, which all have the same probability to receive a new client. We will consider the following scenario where: (i) The network includes four transit-domain nodes (the four nodes in the middle) and 96 stub-domain nodes. (ii) The server is placed in the transit domain, i.e. the shaded square in the middle ($S_p =$ *transit-domain*). (iii) We set T_{pba} to 10 min for PBA. From

[2] In this paper, a node refers only to a router and a peer refers only to a client.

equation (1) we obtain $T_s = L + T_{pba} = 110$ min. Given that value of T_s, the value of T_{cast} for P^2cast is $T_{cast} = 55$ min[3]. (iv) The maximum number of servants a client can contact is $N_s = 10$. (v) The bandwidth capacity of each local area network is $L_{bw} = 50b$.

In both approaches, PBA and P^2cast, we account for the output capacity C_{out} of clients. We consider two values of $C_{out} = \{2, 10\}$. The value $C_{out} = 2$ (respectively 10) means that an existing client can deliver at most two (respectively 10) streams at the same time each at rate b. In the performance evaluation, we are interested in (i) The percentage of rejected clients. (ii) The average number of hops the data traverse in the network. The term hop accounts for a link between any two nodes in the network. (iii) The system cost expressed in units of b. The system cost is computed as the sum of the amount of bandwidth expended over all links in the network during an interval of time of length L. We evaluate these metrics for a wide range of clients arrival rate λ (clients/min). We define the system workload W as the average number of arriving clients per L ($W = \lambda L$).

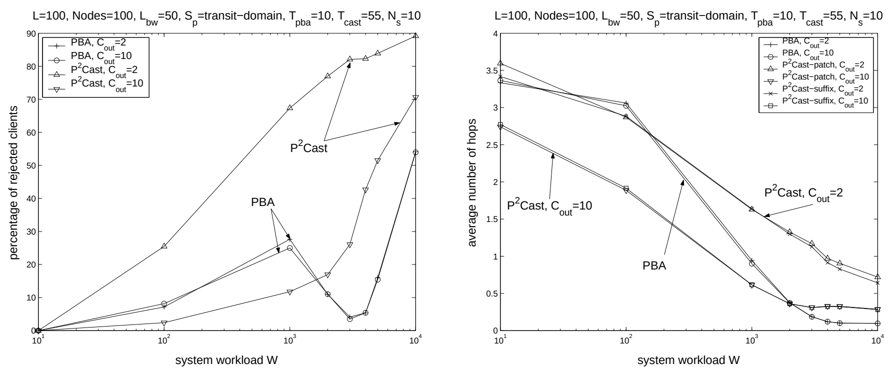
5 Simulation Results

In the following, we present results for the case of a single video of length $L = 100$ min. We run the simulation for a time of 1000 min = $10 \cdot L$, where the first 100 min are to warm up the system.

In figure 2(a), we plot the percentage of rejected clients for PBA and P^2cast vs the system workload W. Figure 2(a) shows that, as compared to P^2cast, PBA reduces significantly the percentage of rejected clients when the system workload W is high. such a reduction can reach 80%. Under a low workload ($W = 10$), the two approaches perform well and almost no clients are rejected. For P^2cast, as W increases, the percentage of rejected clients increases monotonically. In addition, to achieve a reasonable performance, P^2cast requires clients to have a large output capacity (i.e. $C_{out} = 10$). On the other hand, for PBA, the percentage of rejected clients does not increase monotonically with W. This is due to the threshold T_{pba} that we have introduced. T_{pba} limits the number of candidate servants for new clients. For instance, for $W = 100$ (i.e. 100 arriving clients per L), on average, 10 clients arrive within an interval of time $T_{pba} = 10$ min. Thus, few clients can collaborate with each other and candidate servants will be probably located far from the new client. Therefore, clients need to go far in the network to reach their servants. As a result, more bandwidth is consumed in the core of the network, which leads to congestion and clients rejection. When W exceeds 1000, the probability of finding a local servant increases and more clients are served locally. This saves bandwidth in the core of the network and allows to serve more clients. Consequently, the percentage of rejected clients starts decreasing for PBA. For $W > 5000$, the performance of both, PBA and P^2cast deteriorates. This is mainly due to limiting the number of streams in a local network ($L_{bw} = 50$ streams per local network).

Figure 2(b) supports our intuition. Figure 2(b) depicts the average distance (in terms of hops) between clients and their servants. Recall that, the term "hop" accounts for a

[3] We outline that this larger value of T_{cast} gives more advantages to P^2cast as compared to PBA.



(a) percentage of rejected clients (b) average number of hops

Fig. 2. The percentage of rejected clients and the average number of hops as a function of the system workload W, for a network of 100 nodes, $L_{bw} = 50$, the server is placed in the transit-domain, the number of servants is $N_s = 10$, $T_{pba} = 10$ min, and $T_{cast} = 55$ min.

link between any two nodes (routers). Thereby, we do not account for the distance traveled by the data inside the local networks. So, the value $zero$ for the number of hops between a servant and a client means that both, the servant and the client are located in the same local network. As we can observe from figure 2(b), as W increases, the average number of hops decreases. For $W > 1000$, the average number of hops becomes smaller than 1, which means that the majority of clients are served by local servants.

So far, we have seen how, at a high system load, PBA allows more clients to access the service as compared to P^2cast. Let us now consider the total system cost for the two approaches. The system cost is computed as the sum of bandwidth consumed over all links in the network during an interval of time of length L. In figure 5 we draw the system cost for both approaches as a function of the effective system workload W_e. W_e represents the number of admitted clients who get access to the service. We denote by C_{cost}^{PBA} and $C_{cost}^{P^2cast}$ the system costs of PBA and P^2cast respectively. As we can observe from figure 5, PBA consumes less bandwidth than P^2cast for large values of W_e. For moderate values of W_e (i.e. $W_e = 100$), P^2cast slightly outperforms PBA. As mentioned above, this is mainly due to the threshold T_{pba} that we introduced for PBA to limit the interaction between clients. When W_e reaches 1000, C_{cost}^{PBA} decreases, which means that new clients always find local servants to download the video and less bandwidth is consumed in the core of the network. If we look at C_{cost}^{PBA} for $W_e > 5000$, we can notice that it increases again. The reason is that, limiting the bandwidth capacity of local networks to $L_{bw} = 50$ streams forces clients to connect to servants further away.

Note that we also investigated the efficiency of both approaches for other scenarios. The results that we obtained are similar to what we have showed.

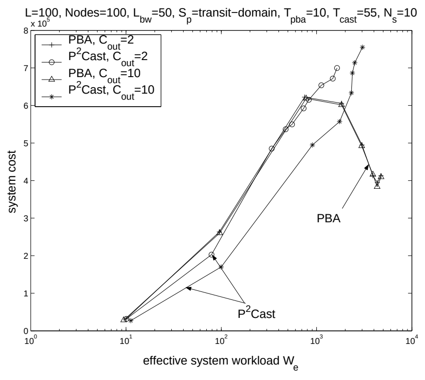
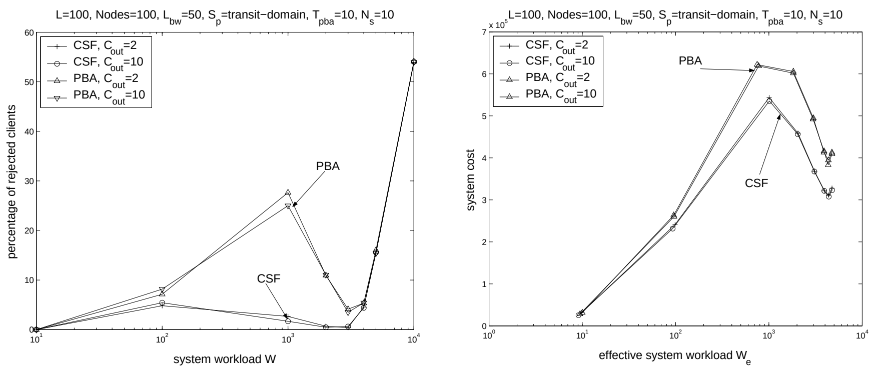


Fig. 3. The system cost for PBA and P^2cast, for a network of 100 nodes, $L_{bw} = 50$, the server is placed in the transit-domain, the number of servants is $N_s = 10$, $T_{pba} = 10$ min, and $T_{cast} = 55$ min.

5.1 Impact of the choice of Servants

In the initial version of PBA, the server chooses the N_s servants at random amongst active clients. In this section, we evaluate an improved version of PBA in case the server is able to account for the physical distance between servants and clients. This means that, instead of choosing N_s servants at random, the server chooses the N_s closest ones to the new client. We refer to this approach as CSF (Closest Servant First). In case of tie, the most recent one is chosen first. Finding nearby clients in the Internet has been addressed recently and an interesting work has been done for this purpose [16, 17].

We compare PBA to CSF to figure out the gain that can be achieved if (exact) locality information are available. The implementation issues are not treated here. In figure 4(a) we draw the percentage of rejected clients for both approaches vs the system workload W. As we can observe from figure 4(a), CSF outperforms significantly PBA for large values of W (i.e. $W = 1000$). Indeed, in CSF, a new client always contacts a near by servant. In contrast, in PBA, if the new client finds no local servant, it picks one candidate at random. For $W = 1000$ arrivals per the video length L, there are 100 arriving clients within a T_{pba} of 10 min. These 100 clients will be distributed over 96 stub-domain nodes. This gives on average 1 client per stub-domain node. Thus, a new client finds with a large probability a near by servant but not a local one. This explains why CSF outperforms PBA for $W = 1000$. In contrast, for $W = 100$, this locality property has no main impact as candidate servants are few (i.e. $\simeq 10$ servants distributed over 96 nodes) and most probably located far away from the new client. Hence, choosing the closest servant or one at random gives the same result. When W becomes high ($W > 1000$), the probability of having a local servant increases and the performance of PBA improves. For high values of W ($W \geq 3000$), a new client finds always a local servant and as a consequence, both PBA and CSF have the same performance. For $W \geq 5000$, the performance of both approaches deteriorates due to limiting the bandwidth capacity of local area networks to $L_{bw} = 50$ streams.



(a) percentage of rejected clients (b) system cost

Fig. 4. The percentage of rejected clients and system cost for both, PBA and CSF, for a network of 100 nodes, $L_{bw} = 50$, the server is placed in the transit-domain, the number of servants is $N_s = 10$, $T_{pba} = 10$ min, $T_{cast} = 55$ min, and $W = 5000$.

Figure 4(b) depicts the system cost for each of the two approaches as a function of the effective system workload W_e. Figure 4(b) confirms our intuition, i.e. accounting for the physical distance between servants and clients allows to save bandwidth and reduces the system cost.

6 Service Disruptions

A main challenge for P2P systems is how to deal with service disruptions. When a client leaves the network, all its descendents will stop receiving the stream until they find a new servant. Also the network state changes dynamically and the available bandwidth between two end hosts might fluctuate over time. This has for sure a severe impact on the service the system offers to its clients. PBA can be extended to handle these issues. The key idea is to allow clients to contact multiple servants instead of a single one. Clients perform locally a greedy algorithm to download different portions of the video from multiple servants at the same time. We assume that the video consists of a sequence of ordered portions. When a new client arrives to the network, it first contacts the server. The server answers to the new client with the IP addresses of N_s candidate servants. The client then connects to N_{cs} out of its N_s servants. Each time a portion that has been requested from a servant is completely downloaded, the client requests from that servant the next portion that has not yet been requested from any other servant. We can pipeline requests to avoid idle times in the different connections.

When a client receives no packet from one of its servants during a given interval of time, the client considers that the servant has left the network. It then drops the connection to that servant and sets up a new connection with a new servant from its list.

Many of recent approaches such as BitTorrent [18] advise the use of parallel download to retrieve the content. Indeed, parallel download helps distributing the charge over the network. In addition, having multiple servants with a total aggregate download rate larger than b (the playback rate of the video) allows clients to quickly fill their buffer. Thereby, when a servant leaves the network, its children should have enough data in their buffer to find a new servant and consequently, experience no interruptions while playing out the video. Parallel download also makes clients less sensitives to the bandwidth fluctuation in the network since the parallel download algorithms always post more requests to the servants that offer the best response time and thus smooth out the load. We leave the quantitative evaluation of this parallel download approach (e.g. choice of N_{cs}, ...) as future work.

7 Conclusion

Previous work on VoD in P2P networks has suggested to construct multicast trees to deliver the video to clients. However, such approaches usually make the server complex. In this paper our goal was to show that we can simplify the algorithm performed at the server and, at the same time, save system resources. For this purpose, we have introduced a new pull-based approach, called PBA. PBA keeps the server as simple as possible. When a new client wishes to receive a video, the new client first contacts the server. If there is enough left over bandwidth along the path to the new client, the server unicasts the video to the new client. Otherwise, the server responds to the new client with a list of candidate servants. These servants are clients that have received or are currently receiving the video. The new client searches then for an available servant to download the video from.

We compared PBA to P^2cast, a multicast-tree based approach proposed previously. Our results showed that PBA consumes significantly less bandwidth and allows more clients to access the service. We then investigated the impact of the choice of servants on the performance of PBA. We showed that allowing the server to choose the N_s closest servants to the new client reduces the bandwidth consumed in the core of the network and allows to serve more clients.

We also discussed how to extend PBA to deal with service disruptions and dynamic changes in the network conditions. The key idea is that, instead of contacting a single servant, a PBA client can perform a parallel download of the video. This makes clients less sensitives to bandwidth variations in the network as well as unexpected departures of servants.

8 Acknowledgments

The authors would like to thank Marry Vernon for her helpful discussions and comments.

References

[1] Almeroth, K.C., Ammar, M.H.: On the use of multicast delivery to provide a scalable and interactive video-on-demand service. In: IEEE JSAC. Volume 14. (1996)

[2] Viswanathan, S., Imielinski, T.: Pyramid broadcasting for video on demand service. In: Proc. of Multimedia Conference, San Jose, CA (1995)

[3] Birk, Y., Mondri, R.: Tailored transmissions for efficient near-video-on-demand service. In: Proc. of ICMCS. (1999)

[4] You, P.F., Pâris, J.F.: A better dynamic broadcasting protocol for video on demand. In: Proc. of IPCCC, Phoenix, AZ (2001)

[5] Gao, L., Towsley, D.: Supplying instantaneous video-on-demand services using controlled multicast. In: Proc. of IEEE Multimedia Computing Systems. (1999)

[6] Eager, D., Vernon, M., Zahorjan, J.: Optimal and efficient merging schedules for video-on-demand servers. In: Proc. of ACM Multimedia. (1999)

[7] Guo, Y., Sen, S., Towsley, D.: Prefix caching assisted periodic broadcast: Framework and techniques for streaming popular video. In: Proc. of IEEE ICC. (2002)

[8] Biersack, E.W., Hamra, A.A., Urvoy-Keller, G., Choi, D.: Cost-optimal dimensionig of a large scale video on demand server. In: Quality of Future Internet Services COST Action 263 Final Report. Volume 2856 of Lecture Notes in Computer Science., Springer-Verlag (2003)

[9] Guo, Y., Suh, K., Kurose, J., Towsley, D.: P2cast: Peer-to-peer patching scheme for vod service. In: Proc. of the 12th World Wide Web Conference (WWW), Budapest, Hungary (2003)

[10] Hu, A.: Video-on-Demand broadcasting protocols: A comprehensive study. In: Proc. of Infocom. Volume 1., Anchorage, Alaska, USA (2001)

[11] Deshpande, H., Bawa, M., Garcia-Molina, H.: Streaming live media over a peer-to-peer network. CSD, Stanford University (2001)

[12] Xu, D., Hefeeda, M., Hambrusch, S., Bhargava, B.: On peer-to-peer media streaming. In: Proc. of 22nd International Conference on Distributed Computing Systems, Washington - Brussels - Tokyo (2002)

[13] Padmanabhan, V.N., Wang, H.J., Chou, P.A., Sripanidkulchai, K.: Distributing streaming media content using cooperative networking. In: Proc. of ACM/IEEE NOSSDAV. (2002)

[14] Tran, D.A., Hua, K.A., Do, T.T.: A peer-to-peer architecture for media streaming. To appear in IEEE JSAC Special Issue on Advances in Overlay Networks (2003)

[15] Zegura, E.W., Calvert, K., S.Bhattacharjee: How to model an internetwork. In: Proc. of Infocom. (1996)

[16] ylvia Ratnasamy: A Scalable Content-Addressable Network. PhD thesis, University of California (2002)

[17] Krisnamurthy, B., Wang, J.: On network-aware clustering of web sites. In: Proc. of SIG-COMM. (2000)

[18] Cohen, B.: Incentives to build robustness in bittorrent. Technical report, http://bitconjurer.org/BitTorrent/bittorrentecon.pdf (2003)