Chasing: An Efficient Streaming Mechanism for Scalable and Resilient Video-on-Demand Service over Peer-to-Peer Networks[⋆]

Jian-Guang Luo, Yun Tang, and Shi-Qiang Yang

Tsinghua University, Beijing 100084, P.R. China
{luojg03, tangyun98}@mails.tsinghua.edu.cn, yangshq@mail.tsinghua.edu.cn

Abstract. Provisioning scalable and resilient Video-on-Demand (VoD) service is both challenging and interesting. Recently, peer-to-peer (P2P) networks are introduced to address the scalability of VoD service over Internet. Most of existing work follows the line of cache-and-relay (CR) scheme to accommodate the asynchronous characteristic of requests from a community of end users. Aiming to take full advantages of bandwidth capacities at each node and pre-recorded feature of requested video files at streaming server, we improve traditional CR approach by efficiently exploiting surplus bandwidth and proactively prefetching media contents from either the server or other peers. Our proposed basic chasing and advanced chasing mechanism not only achieve significant reduction of workload on streaming server, which could translate into better scalability, but also help the streaming session to adapt to volatile network fluctuation. Our extensive experiments have demonstrated encouraging results with respect to increased system performance.

1 Introduction

With the growth of Internet, many researchers have recognized the potential of providing video-on-demand (VoD) service to a large population of network users. In traditional client/server approach, each client subscribes streaming service directly from the server through unicast connections. Due to high bandwidth consumption and long-playing features of streaming sessions, the server soon becomes the bottleneck of the system as the popularity increases. Therefore, a number of IP multicast based solutions, comprising batching [8], patching [9], periodical broadcasting [10], stream merging [11], and etc., have been proposed. However, they are infeasible to deploy in the absence of network infrastructure support. Other efforts advocate proxy-based approaches [12] or CDN [13] to balance server workload by deploying a number of secondary servers at the edge of network. However, it is difficult to place the servers efficiently over Internet due to the high dynamic distribution of clients. Furthermore, the exorbitant cost also affects the wide deployment of proxy-based solutions or CDN.

[⋆] This work is supported by the National Natural Science Foundation of China under Grant No.60432030 and No.60503063.

Fortunately, peer-to-peer (P2P) architecture is introduced into the arena of large scale data dissemination in recent years. In a typical P2P environment, each node plays the role of both server and client, contributing its available computation, storage and/or bandwidth resources into the collective resource pool. As the benefits accrue with mutual cooperation among peers, P2P community offers a scalable approach to support a large number of users without either costly servers or network infrastructure support. Therefore, P2P networks are promising to solve the scalability problem in large scale VoD applications.

Despite the success of P2P live video streaming [5–7], P2P VoD services are of significant differences. Most importantly, intrinsic to the notion of on demand, the video viewed by different users may vary in file position. This asynchronous nature of VoD might be regarded as running counter to the fundamental design philosophy of P2P networks. Besides, compared to live video streaming, the requested video are always pre-recorded in typical VoD systems, and this "***forward-availability***" enables appealing features of content prefetching. Most of existing work follows the line of cache-and-relay (CR) scheme [1, 2] to tackle the asynchronous problem, in which end nodes do not prefetch any contents in advance, and thus the visual quality at users might suffer severely from network fluctuations. Another latest proposed protocol dPAM [3] utilizes the surplus downloading bandwidth of each peer to prefetch some portions of video content, so it really alleviates the disruption of video playback under dynamic network fluctuation and provides each node capabilities to recover from upstream nodes failure or departure. Nevertheless, as soon as the incoming bandwidth of peers is lower than twice the video playback rate, which is fairly common in realistic networks, dPAM will cause much more users to directly stream from source server, which in turn imposes higher workload than CR scheme.

In this paper, we try to take full advantages of bandwidth capacities at each node and pre-recorded feature of requested files at streaming server to achieve better system scalability and resilient visual QoS. Our main contributions lie in twofold. First, by efficiently exploring surplus bandwidth, we propose basic chasing to greatly reduce the server bandwidth cost. Second, we enforce proactive prefetching from either the server or other peers, namely advanced chasing to ensure that all peers are resilient to volatile network fluctuation.

The remainder of the paper is organized as follows: Section 2 gives a brief introduction to CR and dPAM and further derive our motivations. Section 3 presents the proposed chasing mechanism in detail and we intentionally separate it into basic and advanced chasing with respect to different motifs. Simulation results of experiments and comparisons are provided in Section 4. At last, we conclude the paper and discuss future work in Section 5.

2 Cache-and-Relay and Prefetching

In this section, the key principles and drawbacks of CR and prefetching schemes are briefly analyzed to further concentrate our motivations. In the context of this paper, we only consider the representative case of CBR media distribution with single source server. For sake of exposition, the playback rate of video stream is assumed to be r_p, while the downloading rate of client is denoted as r_d. The maximum downloading bandwidth capacity of each peer is assumed to be r_c identically. Thus, we have the following relations: $r_c \geq r_p$, $r_d \leq r_c$. In the rest of the paper, we use R_1, R_2 and R_3 to denote the sequential requests or clients which arrive at time t_1, t_2 and t_3. And the buffer size of R_1, R_2 and R_3 is w_1, w_2 and w_3 time units respectively. Each peer in P2P networks is able to buffer the streamed content for a certain amount of time and overwrite its buffer window in a circular manner.

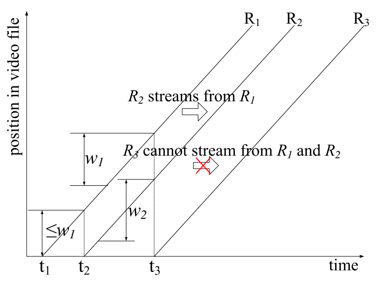


Fig. 1. Traditional cache-and-relay scheme: illustrative example

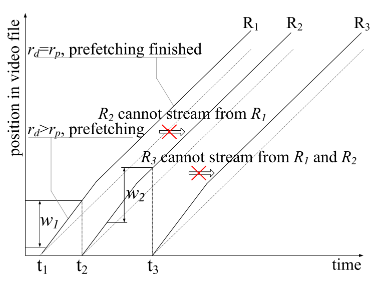


Fig. 2. Prefetching scheme in dPAM: illustrative example($\alpha = 1.33$)

The CR scheme is illustrated schematically in Fig.1. As shown, when R_2 demands the service at t_2, the contents desired by R_2 are available within the buffer of R_1 because the gap between R_1 and R_2 falls into the size of R_1's buffer, that is, $t_2 - t_1 \leq w_1$. Hence, R_2 starts streaming directly from R_1 instead of going to the server. However, when R_3 joins the service community at t_3, it could not find appropriated upstream clients since the requisite contents has already been drained out from the buffer of both R_1 and R_2. Here R_3 will seek to the support of the source server. Although CR has been demonstrated to scale well in [1], the downloading rate of peers is always equal to the playback rate, which means all the peers attempt to play the newest content in their buffers. Therefore, the visual quality of playback is highly sensitive to network fluctuations and the departure or failure of upstream peers.

Prefetching scheme is proposed in dPAM [3] to alleviate QoS degradation with the consideration of "forward-availability". Fig.2 exhibits the examples of prefetching, in which each client tries to fill its own buffer at the maximum downloading rate. The solid line depicts the position of the downloading content, while the dotted line shows the position of content being playback at the end node. The fundamental advantage of prefetching is to store more contents by exploiting surplus bandwidth of clients and hence offering an increased service robustness to network dynamics. However, it could also be observed that R_2 is not able to stream from R_1 anymore under the same arrival pattern in Fig.1. This is because the content cached in the buffer of R_1 is refreshed at a faster rate than in CR scheme. In such a case, R_2 has to establish a new connection to the server if the maximum downloading rate r_c is smaller than twice the playback rate R_p. To make the question clear, we define $\alpha = r_c/r_p$, which could be utilized as the indication of prefetching capabilities. We could deduce that dPAM will impose more bandwidth consumption upon server even than CR scheme when $\alpha < 2$. In Fig.2, $\alpha = 1.33$.

Essentially motivated by above analysis, in next section we mainly consider how to effectively utilize surplus bandwidth capacity of peers to improve system properties in terms of reduced server workload and higher service resilience. Our objective comes in twofold. One is to further reduce bandwidth cost of server to keep the system scalable. The other is to eliminate QoS degradation of video playback due to network fluctuations and parent switch operations by prefetching portions of contents.

3 Chasing Mechanism

In this section, we depict chasing mechanism in two steps. Firstly, in subsection 3.1, by efficiently exploiting surplus bandwidth capacity at each node, we improve traditional CR scheme by basic chasing mechanism, which significantly reduce server bandwidth cost. Secondly, in subsection 3.2, by proactively considering the "forward-availability" of requested files, we enable more end clients to prefetch contents by advanced chasing mechanism, and therefore alleviating visual quality degradation. It should be stressed that, in this paper we mainly concentrate our work in the context of $1 < \alpha < 2$. Otherwise each node could directly apply patching techniques by opening two streams simultaneously.

3.1 Basic Chasing Mechanism

The substantial drawback of traditional CR scheme lies in the striking workload imposed by "***lonely***" peers, who are either newcomers or foundling peers. Aiming to further reduce the consumption of server bandwidth, we intuitively resort to encourage more peers to take charge of the fostering procedure. In basic chasing mechanism, each peer attempts to catch up with the earlier ones which are already in the community. Compared to Fig.1, R_3 with basic chasing mechanism in Fig.3 is now able to stream from R_2 with a transient favor of server under the same arrival pattern. Hence the bandwidth consumption of server is diminished. The detailed discussions proceed along the dimensions of new arrival and parent failure or departure in following schematic analysis.

New Arrival: When the new coming peer R_3 enters the service community at time t_3, the oldest content in the buffer of R_2 is $w_d = (t_3 - t_2 - w_2) > 0$, leaving a gap of w_d time units to what R_3 demands. Thus R_3 establishes a streaming connection rated at r_p from server for these $[0, w_d)$ packets. At the same time, R_3 is still able to fetch portions of intending video contents in advance from R_2 with the aid of the surplus bandwidth $(\alpha - 1) \times r_p$, different from CR. This parallel downloading procedure essentially builds the most important component

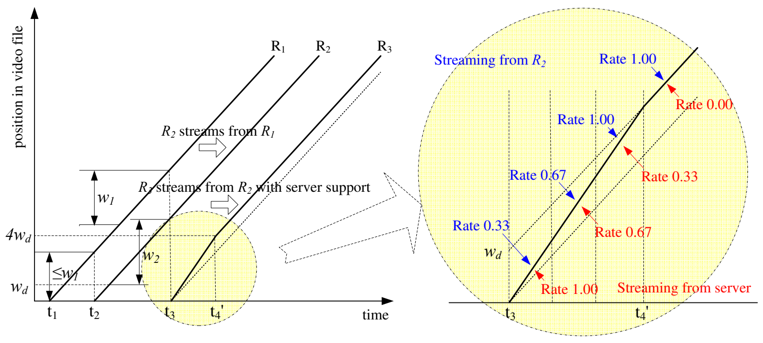


Fig. 3. Basic chasing: illustrative example($\alpha = 1.33$)

of basic chasing mechanism. Till $t_3 + w_d$, R_3 starts to playback the video stream from position w_d. Since R_3 has already cached parts of the stream from R_2 ranged in $[w_d, 2w_d)$, R_3 now only need to download the remaining stream rated at $r_p - (r_c - r_p) = 2r_p - r_c$ from server. Therefore, the streaming bandwidth from R_2 to R_3 currently increases to $r_c - (2r_p - r_c) = 2(r_c - r_p)$. If $2(r_c - r_p) \geq r_p$, i.e., $\alpha \geq 1.5$, R_3 is now able to download the entire stream from R_2. Consequently, at time $t_3 + 2w_d$, basic chasing mechanism terminates the parallel streaming process and from then on R_3 can be fully served by R_2. If $\alpha < 1.5$, the process will continue for a longer duration, and in each interval w_d, the bandwidth of R_3 spent to stream from R_2 increases by $(r_c - r_p)$. Since the bandwidth used by R_3 to prefetch content increases step by step, we call the chasing process "***step-like***". Basically, R_3 totally spends $ceil(1/(\alpha - 1)) \times w_d$ time units to catch up with R_2 altogether, resulting in a lower server workload after an initial acceleration.

Departure or Failure of Parent: When one node leaves the P2P community or experiences a transient failure, the streams to all of its children nodes halt. Then those downstream foundlings have to search new suppliers or alternatively ask the server for help. We keep in mind that if some peers have employed basic chasing at previous rounds, they should have some prefetched content in their buffers. So we depict an general non-overlapped buffer scenario of R_1 and R_2 in Fig.4, in which R_1 and R_2 has w_{p1} and w_{p2} time units of content prefetched in their buffers respectively, and the "missing" content between the buffer of R_1 and R_2 is w_d time units. Since the prefetched content and "missing" content (if downloaded) will be exhausted to playback in $(w_{p2} + w_d)$, the bandwidth used by R_2 to download the missing content from server should be at least $w_d \times r_p/(w_{p2} + w_d)$ to ensure the smooth playback. Then the chasing scheme can be understood in the following two cases:

1) If $w_d \times r_p/(w_{p2} + w_d) \leq (r_c - r_p)$, R_2 uses bandwidth of $w_d \times r_p/(w_{p2} + w_d)$ to fetch the "missing" content from server, and meanwhile the remainder bandwidth is still enough to prefetch the entire stream from the buffer of R_1.

2) Otherwise, since R_2 has to reserve the bandwidth of $w_d \times r_p/(w_{p2} + w_d)$ to ensure the smooth playback, R_2 only has bandwidth rated at $r_c - w_d \times r_p/(w_{p2} +$

w_d) to prefetch portions of stream from R_1, and is forced to start a "step-like" chasing process to catch up with R_1.

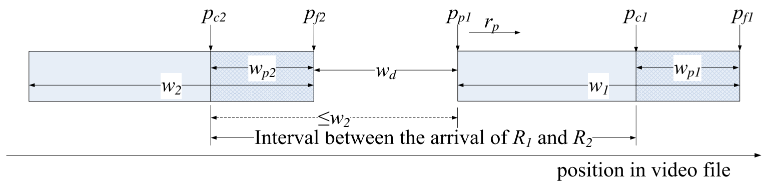


Fig. 4. Non-overlapped buffer scenario of R_1 and R_2 in basic chasing mechanism

Whether a peer just joins the service community or it experiences parents' failure or departure, the basic chasing mechanism described above could achieve lower server workload because the surplus bandwidth at each peer indeed helps server to share the responsibility to foster. We emphasize this salient feature of basic chasing as one of main contributions in this paper. Afterwards, several key components of proposed mechanism will be examined summarily.

Buffer Size: In basic chasing mechanism, the buffer size and the interval between the arrivals of peers definitely plays a key role in the decision to whether the former peers can be chased. We can image in Fig.4, if $w_2 < w_{p2} + w_d$, the buffer of chasing client R_2 could not cache the contents going to prefetch during the process of chasing R_1. Since we have assumed a sequential access model for client requests, given the buffer size w_1, w_2 of R_1 and R_2, the maximum interval between their arrivals is $(w_1 + w_2 - w_{p1})$ so that R_2 can stream from R_1. Recall that the maximum interval allowed in cache-and-relay scheme is w_1, only depending on the buffer size of R_1. We claim that basic chasing explores the utility of the buffer of the chasing clients, thus provides more clients with opportunities to share a single stream from server, and in turn effectively reduce the server bandwidth cost.

Downloading Bandwidth: We have assumed the maximum downloading bandwidth r_c of peer is ranged between the playback rate and twice of this rate, i.e. $1 < \alpha < 2$. Although it is clear that whether a client could chase an earlier node depends on the buffer states of clients, rather than the maximum downloading bandwidth, the downloading bandwidth capacity will definitely affect the chasing process due to the "step-like" characteristic. Clearly, the larger α is, the less steps chasing process needs and in turn less contents would be stream from server. Therefore, we claim that larger downloading bandwidth of peer will lead to lower server workload. This result will be clearly demonstrated in simulation experiments in Section 4.

Our proposed basic chasing mechanism is similar to the partition scheme of bandwidth skimming [4]. However, there are two distinguished differences between them. One is that proposed chasing applies to P2P environment while the partition scheme of bandwidth skimming is deployed in IP multicast. The other is that chasing streaming in general is deemed to catch up with the buffer of parents and it goals to achieve asynchronous multicast in overlay network. On the contrary, clients with partition scheme tries to follow the step of IP multicast stream, composing unique synchronous multicast eventually.

In this subsection, we mainly illustrate the basic chasing mechanism for newcomers and ones that suffer from parents' failure or departure. Additionally, some root parameters of this mechanism, e.g. buffer size and downloading capacity are also analyzed qualitatively. In next subsection, we further extend basic chasing mechanism to offer a more resilient streaming mechanism.

3.2 Advanced Chasing Mechanism: Proactive Prefetching

In basic chasing mechanism, the parallel downloading procedure aids a continuous playback for those who are often regarded as "***lonely***" peers. However, for other normal subscribers, if they could find appropriate upstream peers to obtain cooperative services when joining the system, all of them will receive the stream at the playback rate, so that they will suffer severe visual quality degradation from network fluctuation. Inspired by the prefetching in basic chasing mechanism, we import proactive prefetching into other peers and improve basic chasing scheme by enforcing them to cache some contents in advance from server or other peers. This so called advanced chasing mechanism ensures considerable resilience to network dynamics.

Fig.5 exhibits the advanced chasing mechanism explanatorily. Different from traditional CR scheme and basic chasing mechanism, advanced chasing tries to download the contents from either server or other peers at the maximum downloading bandwidth rate r_c once R_1 arrives in the system. As soon as the prefetched contents reach a predefined threshold, R_1 slows its downloading rate to the playback rate r_p and the primordial prefetching ceases. For the sake of explanation, here we define β as ratio of the contents that one is willing to prefetch to its own buffer size. As shown in Fig.5, R_1 decreases its downloading rate r_d to r_p once its prefetched content achieves $\beta \times w_1$ at time t'_1. Clearly, the quick buffer refreshment due to proactive prefetching in advanced chasing will impose an obstacle to the stream sharing between clients as in dPAM. At time t_2 the contents required by new participant R_2 have already been overwritten by the prefetched data in the buffer of R_1. Then R_2 cannot help but to attain what it needs from the server with dPAM scheme. However, R_2 in Fig.5 is still able to stream from R_1 with proposed basic chasing mechanism. Nevertheless, compared to basic chasing, if the forthgoer R_1 deploys advanced chasing mechanism to proactively prefetch, it is apparent that the chance for R_2 to chase R_1 is diminished.

As mentioned above, with advanced chasing mechanism R_1 may be downloading contents at its maximum downloading rate r_c and refreshing its buffer equally when R_2 attempts to chase R_1, which is illustrated in Fig.6. Observe that R_1 and R_2 already have w_{p1} and w_{p2} time units of prefetched contents in their buffers respectively. Because the pointer p_{p1} is moving more quickly than the playback rate r_p, it is infeasible to adopt the basic chasing mechanism directly to obtain portions of the stream contents from p_{p1} at the buffer of R_1. Recall that

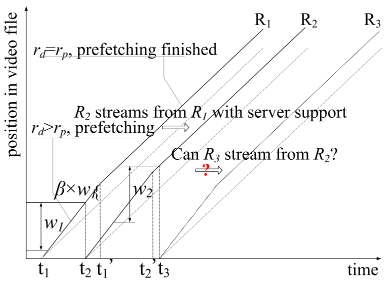


Fig. 5. Advanced chasing mechanism: illustrative example($\alpha = 1.33$)

R_1 will decrease its downloading rate to r_p afterwards as soon as its prefetched contents achieve $\beta \times w_1$, so henceforward R_2 is able to gain cooperative service from the buffer of R_1 at position p'_{c1} shown in Fig.6. Therefore, the interval between the arrivals of R_1 and R_2 should be smaller than $(1 - \beta) \times w_1 + w_2$ to ensure that R_2 can be served by R_1. Evidently, the benefits among peers to share burden of server decrease slightly when compared to basic chasing, but we still argue that if the buffers are of same size, i.e. $w_1 = w_2$, the maximum arrival interval in advanced chasing is larger than that in cache-and-relay scheme if $\beta < 1$, which still provides increased performance in terms of reducing the server bandwidth in comparison with CR scheme.

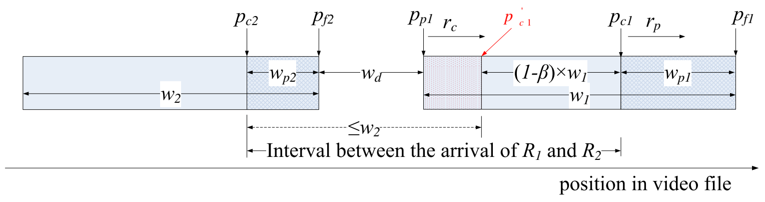


Fig. 6. Non-overlapped buffer scenario of R_1 and R_2 in advanced chasing mechanism

Due to space limitation, we do not intend to present every detail within the advanced chasing mechanism. By similar reasoning, we could understand it with buffer size, downloading capacity and other complexities. Intuitively, in advanced chasing mechanism, all the clients will prefetch some contents due to the proactive prefetching process, so that they can conceal a temporary degradation or jitter of downloading bandwidth without affecting the playback quality. From the viewpoint of system performance, advanced chasing mechanism attemps to balance the reduction to server workload by basic chasing with the resilience level that prefetching provides. In the face of extreme scenario, basic chasing could be thought as the case with $\beta = 0$ of advanced chasing mechanism.

4 Performance Evaluation

In this section, we evaluate the performance of proposed chasing mechanism by extensive simulations. We first explain the setup of our simulation in subsection 4.1, and then quantify the comparison between basic chasing and CR in subsection 4.2, and in subsection 4.3 we compare the advanced chasing with both dPAM and basic chasing with respect to the amount of prefetched contents.

4.1 Simulation Setup

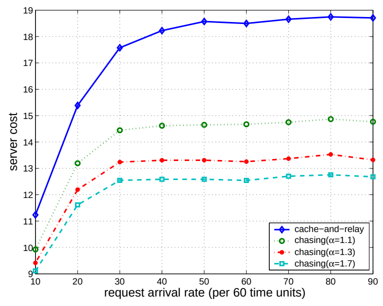
We implemented a discrete time driven tool to simulate basic chasing, advanced chasing, as well as cache-and-relay and dPAM mechanisms. To make the question clear, we reasonably simplify the heterogeneity of end hosts by assuming they are with identical buffer size w, maximum bandwidth capacity ratio α and desired ratio of prefetched contents β. Since most of existing P2P applications always provision substitutes foreseeingly, it is also assumed that each peer could determine where it should obtain service without any delay when it suffers the departure or failure of parents.

In simulation, request arrival pattern is produced according to a standard Poisson process with rate λ and the duration of clients staying in the community follows an exponential distribution with rate μ. It should be mentioned that the online user number will increase continuously at the very beginning of the simulation and then come into a steady state after a while. Here we utilize the average ratio of server's total outflow traffic to one single flow at playback rate to evaluate the server bandwidth consumption. This metric indicates how many streams flow out from the server and, more weightily, also take the duration of stream session into consideration. Each point on the following figures represents an average over 10 independent runs.

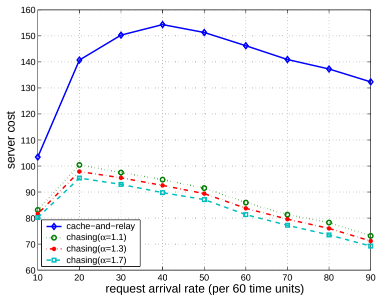
4.2 Performance of Basic Chasing Mechanism

We design experiments to investigate the performance comparison between basic chasing mechanism and cache-and-relay in terms of server bandwidth consumption and the simulation result are illustrated in Fig.7(a)-(d) with different average online duration $1/\mu$ and buffer size w, respectively. Observe that whatever these parameters are, the server bandwidth consumption in base chasing mechanism is always lower than CR scheme. Also it declines with the increase of the downloading bandwidth capacity α. Towards this end, we claim that all of those results validate previous analysis in subsection 3.1 well. More interestingly, when $\alpha < 2$, a larger α in dPAM translates into aggravation to the bandwidth consumption, but descendent trends in figures demonstrate the basic chasing mechanism can exploit cooperation among peers more efficiently when they are equipped more resources.

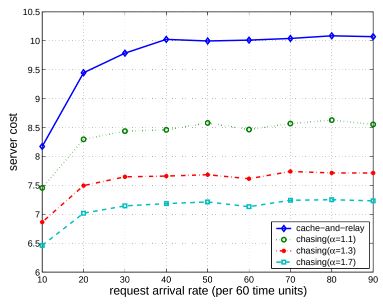
Impact of online duration: Aiming to get a thorough understanding to P2P on demand system, we conduct experiments for different average online durations. Obviously we see a remarkable descending trend with higher $1/\mu$ in



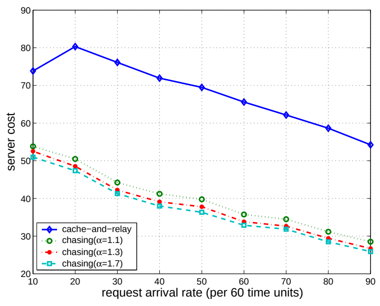
(a)$1/\mu = 100$, $w = 5$



(b)$1/\mu = 1000$, $w = 5$



(c)$1/\mu = 100$, $w = 10$



(d)$1/\mu = 1000$, $w = 10$

Fig. 7. Basic chasing vs. cache-and-relay

Fig.7(b) compared to that in Fig.7(a), as well as an analogous result in Fig.7(c) and Fig.7(d). The apparently depressed server workload consumption essentially comes from a smaller churn rate of nodes in P2P networks, and in turn lower probabilities for peer to become ”lonely” foundlings due to the departure or failure of parents.

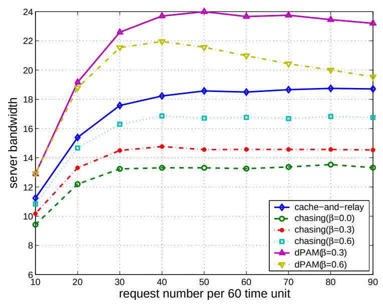
Impact of buffer size: Besides the online duration time, we also pay much attention on another important factor, i.e. the buffer size which is known to basically determine the collaboration among peers. Fig.7(a)-(d) shows that server workload decreases tremendously as the buffer size increases from 5 time units to 10 time units. Note that the inflexion at 20 requests per 60 time units in Fig.7(b) disappears in Fig.7(d), which means no ”fall-after-initial-rise” exists when buffer size $w = 10$. Actually it is not surprising if we notice that the average interval between peer arrivals is 6 or 3 time units when the arrival rate is 10 or 20 requests, and accordingly a 10-unit buffer achiever much higher possibility to share the fostering burden of servers than a 5-unit buffer.

With the basic chasing mechanism, roughly only 50% server bandwidth consumption is necessary to achieve the same goal to deal with peer requests in comparison with CR scheme. These experiments have demonstrated that by efficiently taking advantages of bandwidth capacity at end users, basic chasing mechanism outperforms traditional CR scheme to provide a more scalable approach.

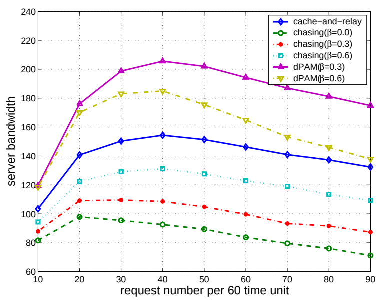
4.3 Performance of Advanced Chasing Mechanism

As mentioned previously, we proactively enforce more peers to prefetch contents in advanced chasing mechanism. Since the principal idea derives from the tradeoff between the server workload and transmission resilience, we attempt to concentrate on the interplay among multiple factors under various desired ratio of prefetched contents β.

Fig.8(a) and Fig.8(b) respectively depicts the server bandwidth consumption of CR, dPAM and advanced chasing mechanism over different online durations. Clearly, dPAM imposes striking workload upon server bandwidth, as we argued above, since it demands more service directly from sever when $\alpha < 2$. Although the lines describing advanced chasing mechanism always locate on top of those denoting basic chasing mechanism ($\beta = 0.0$), as a matter of factor, we conclude that the advanced chasing outperforms CR scheme and achieves higher resilience in terms of playback continuity. Fig.9 further provides an insightful comparison to the statistical ratio of prefetched peers to total population with various β. It shows that even if $\beta = 0.4$, there are more than 90% peers which have prefetched more than 2-unit contents. As a result, we improve basic chasing mechanism with the consideration that prefetching will help more peers to accommodate network fluctuation.



(a)$1/\mu = 100$, $w = 5$, $\alpha = 1.3$



(b)$1/\mu = 1000$, $w = 5$, $\alpha = 1.3$

Fig. 8. Advanced chasing vs. cache-and-relay

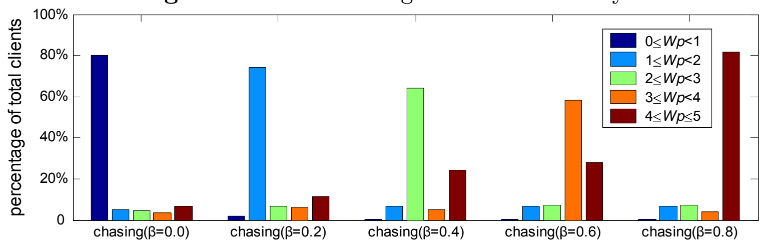


Fig. 9. Statistics of clients in prefetching content ($1/\mu = 1000$, $\lambda = 20$ per 60 time units, $w = 5$, $\alpha = 1.3$)

5 Conclusions and Future Work

In this paper, we propose an efficient streaming mechanism for scalable and resilient P2P VoD systems. We focus on how to effectively reduce the server workload in the process of fostering "lonely" peers and improve playback continuity with respect to network fluctuation. Our approach, namely basic chasing and advanced chasing mechanism, respectively helps the reduction to server bandwidth consumption by efficiently exploring surplus bandwidth for more collaboration, and offers higher resilience by proactively prefetching contents without loss of the gain to workload reduction.

The positive feedback from simulation experiments encourages us to continue our work in detailed analysis to both basic and advanced chasing mechanism, comprising the quantitative impact of buffer size, maximum downloading bandwidth, as well as prefetching amount. Our goal is to develop a practical on demand video streaming system and investigate the fundamental performance properties.

References

1. Y. Cui, B.C. Li, and K. Nahrstedt. oStream: asynchronous streaming multicast in application-layer overlay networks. *IEEE JSAC*, Vol.22, No.1, January 2004.
2. S. Jin, and A. Bestavros. OSMOSIS: scalable delivery of real-time streaming media in ad-hoc overlay networks. In *Proceedings of IEEE ICDCS'03 Workshop on Data Distribution in Real-Time Systems*, May 2003.
3. A. Sharma, A. Bestavros, and I. Matta. dPAM: a distributed prefetching protocol for scalable asynchronous multicast in p2p systems. In *Proceedings of the IEEE INFOCOM*, March 2005.
4. D.L. Eager, M.K. Vernon, and J. Zahorjan. Bandwidth skimming: a technique for cost-effective video-on-demand. In *Proceedings of SPIE MMCN*, January 2000.
5. Yang-Hua Chu, Sanjay G. Rao, and Hui Zhang. A case for end system multicast. In *Proceedings of ACM SIGMETRICS*, June 2000.
6. X. Zhang, J. Liu, B. Li, and T.-SP Yum. CoolStreaming/DONet: a data- driven overlay network for live media streaming. In *Proceedings of IEEE INFOCOM*, March 2005.
7. M. Zhang, J.-G. Luo, L. Zhao, and S.-Q. Yang. A peer-to-peer network for live media streaming - using a push-pull approach. In *Proceedings of ACM Multimedia*, November 2005.
8. A. Dan, D. Sitaram, and P. Shahabuddin. Scheduling policies for an on-demand video server with batching. In *Proceedings of ACM Multimedia*, October 1994.
9. K.A. Hua, Y. Cai, and S. Sheu. Patching: A multicast technique for true video-on-demand services. In *Proceedings of ACM Multimedia*, September 1998.
10. T. Chiueh, and C. Lu. A periodic broadcasting approach to video-on-demand service. In *Proceedings of SPIE MMCN*, January 1995.
11. D.L. Eager, M.K. Vernon, and J. Zahorjan. Optimal and efficient merging schedules for video-on-demand servers. In *Proceedings of ACM Multimedia*, November 1999.
12. Y. Wang, Z.-L. Zhang, D. Du, and D. Su. A network conscious approach to end-to-end video delivery over wide area networks using proxy servers. In *Proceedings of IEEE INFOCOM*, April 1998.
13. Akamai, http://www.akamai.com/.