On the Design of Scalable Peer-to-Peer Video Caching

Alex Bikfalvi
IMDEA Networks
alex.bikfalvi@imdea.org

Nazanin Magharei
University of Oregon
nazanin@cs.uoregon.edu

Reza Rejaie
University of Oregon
reza@cs.uoregon.edu

Jaime García-Reinoso
University Carlos III of Madrid
jgr@it.uc3m.es

ABSTRACT

Peer-to-Peer (P2P) video caching is a promising approach to accommodate asynchronous requests from cached content at individual peers. However, coherently managing a distributed, heterogeneous, dynamic and potentially large scale cache space is a challenging task. In particular, a key challenge is to effectively control the number of cached copies for popular streams in order to accommodate their concurrent requests with minimum thrashing in the cached content. A few prior studies on P2P video caching rely on the global cache state in order to achieve this goal and, therefore, exhibit limited scalability.

This paper examines key issues in the design of a scalable P2P video caching mechanism that can effectively control the number of cached copies for popular streams by only leveraging the local information at each peer. We argue that the local notion of popularity can serve as an effective measure to perform cache replacement at individual peers. We sketch two straw-man P2P video caching techniques that rely on the trends in the popularity of individual streams to control the required number of copies in a reactive or proactive fashion. Using simulation, we examine the performance of the proposed mechanisms along with the distributed (and uncoordinated) version of LRU and LFU mechanisms that only use local workload at each peer. Our results show that distributed and uncoordinated P2P video caching generally exhibit good performance across a wide range of scenarios.

1. INTRODUCTION

Peer-to-Peer (P2P) overlays have become increasingly popular for one-to-many delivery of live video streams [4] that we refer to as P2P streaming. P2P streaming accommodates scalability by enabling individual peers to contribute their resources, namely outgoing bandwidth. A promising approach to seamlessly accommodate asynchronous sessions (*i.e.,* playback) into P2P streaming is to leverage storage across participating peers as cache space. This enables the peers who view a live session to cache (all or part of) the associated video stream, and later serve the asynchronous requests for that stream. For example, users who watch a live concert through a P2P streaming session can cache the video and playback the stream for other users who may decide to watch the video at a later time. We refer to this approach as P2P video caching (or PVC).

While a P2P cache architecture ensures scalability by enabling individual peers to contribute their storage into the overall cache space but, it introduces new challenges that does not exist in traditional proxy caching of video [5]. In general, coherently managing a P2P cache space is challenging due to the distributed, heterogeneous, dynamic and potentially large scale nature of contributed storage by individual peers. More specifically, using common caching techniques (LRU and LFU) for PVC implies that a copy of a popular stream is cached at any peer who viewed the stream. The skewed nature of stream popularity [2] results in too many cached copies of popular streams, which in turn leads to eviction of less popular stream and thrashing of cached content. The ability of PVC to serve concurrent requests for a cached stream is limited to the aggregate outgoing bandwidth across peers who have a copy of the stream in their local cache. A key trade-off in the design of a PVC mechanism is to minimize thrashing of cached content by adaptively controlling the number of cached copies for each popular stream up to the level that is required to accommodate the concurrent demand from asynchronous users. To manage a P2P cache while scaling to a large population of users, the solution neither should use a central approach nor should it require tight coordination among cache replacement mechanisms at individual peers.

To our knowledge, a few previously proposed P2P video caching mechanisms [1, 9, 10] treat the P2P cache space as a single proxy cache. In these mechanisms, individual peers require global information about cache space in order to perform local cache replacement which in turn limits their scalability.

This paper examines key issues in designing a scalable PVC mechanism that can effectively manage a P2P cache space in a distributed and uncoordinated fashion. Toward this end, first we briefly describe some of the key differences between commonly used proxy caching techniques for video streams and their differences with PVC mechanism in Section 2. We identify the management of the number of cached copies as an important and new design issue for the PVC mechanism, especially in the presence of workload with skewed popularity distribution among video streams. We clarify the opposite effects of the number of cached copies on the cache performance and the ability of the cache to serve concurrent requests. We then sketch two straw-man PVC mechanisms in Section 3, focusing on the content discovery and request management. In particular, we argue how the local notion of popularity (based on requests that are observed by individual peers) can be effectively used as an implicit signal among peers to determine caching decision without any coordination with other peers. Furthermore, we present a simple technique to control the excess number of cached copies of popular streams by leveraging the smoothed trend in the evolution of their popularity for cache replacement. The key idea is to select a victim stream for replacement from those streams whose smoothed popularity is not growing. We discuss several issues in implementing such a technique, and illustrate how it can be used in a reactive and proactive fashion.

Section 4 presents the preliminary evaluation based on simulations of the proposed straw-man mechanisms along with LRU and LFU schemes for comparative purposes. Our main findings can be summarized as follows:

(i) Our proposed PVC mechanisms and distributed LRU exhibit good performance over the wide range of scenarios. PVC mechanisms slightly outperform distributed LRU while caching a smaller number of copies of popular streams. Distributed LFU's performance is significantly lower than other techniques.

(ii) Using the local notion of popularity offers a reliable signal to perform cache replacement and managing the number of copies of popular stream in an uncoordinated fashion without capturing global state of cached content.

We briefly review the prior work on P2P video caching in Section 5, and finally, Section 6 concludes the paper and sketches our agenda for future work on this topic.

2. PROXY VS P2P VIDEO CACHING

Similar to Web caching, video caching has traditionally been used at a proxy server. In such a setting, an adequately provisioned proxy server is associated with a group of closely-located clients. The proxy intercepts the requests for the associated clients and provides the requested stream from the cache when it is available (*i.e.,* cache hit). Otherwise, the request is relayed to the original server, the provided stream is cached and it is sent to the client. In proxy caching, at most one copy of popular streams is needed in the cache, and the outgoing bandwidth is provisioned to serve the likely number of concurrent requests from the associated clients.

In the context of PVC, generally there is no previously specified mapping between requesting (or client) peers and caching (or serving) peers. Each peer may serve a stream from its local cache to potentially any peer that may request the stream. The distributed nature of cache space in PVC results in two important differences compared to proxy caching:

- *Multiple Cached Copies*: a popular stream could be cached at different (and potentially many) peers after they view the stream,
- *Limited Serving Bandwidth*: The ability of the system to serve multiple copies of the stream s depends on the aggregate outgoing bandwidth of all peers that have a copy of that stream. We call this "serving bandwidth for the stream s" and it depends on the number of cached copies.

Caching too few copies limits the serving bandwidth for a stream, while caching too many copies results in unnecessary thrashing of cache content by evicting the least popular streams from the cache. The reported skewed distribution of stream popularity [2] coupled with the commonly used strategy of caching, any newly popular received stream, could easily lead to many cached copies of that popular stream while the resulting serving bandwidth may significantly surpass the demand.

Therefore, the key trade-off in the design of a PVC mechanism is to limit the number of cached copies of an individual stream to the level that can support the likely number of concurrent requests. In turn, this enables the cache to effectively serve popular streams while minimizing the level of thrashing in the aggregated cache content and the resulting performance hit. Specifically, by limiting the number of copies for popular streams, we can maintain and serve less popular streams from the cache and improve cache hit rate without affecting serving bandwidth for popular streams.

We envision PVC mechanism to be deployed in two scenarios:

(i) Infrastructure-based scenario, where individual peers are dedicated nodes such as set-top boxes that are owned by a company. In this case, participating peers are likely to have homogeneous capabilities and be more stable.

(ii) Pure P2P scenario, where participating peers are computers owned by individual users. This scenario is more heterogeneous, dynamic and thus more challenging. We note that the problem of managing the number of copies in both scenarios is equally important.

3. SCALABLE P2P VIDEO CACHING

This section describes different issues in design of PVC mechanisms and sketches two straw-man mechanisms.

Content Discovery: In order to locate a particular stream in the distributed P2P cache, there must be a content discovery mechanism to determine whether a particular stream s is in the cache, and which peer(s) can provide a copy of stream s. We assume that a boot-strap node keeps track of cached streams at individual peers, using a soft-state update mechanism. Supposing that each stream is divided into segments, each peer periodically sends an update to the boot-strap nodes and reports the bitmap of available segments for each cached stream. The boot-strap node incorporates each report and maintains a relatively accurate view of the available segments for each stream among participating peers. Any change in the cached content is reported in the next update and is reflected in the information at the boot-strap node. The information associated with a departing peer is removed in the absence of a periodic update.

An alternative approach for content discovery is to use a DHT among participating peers [1], where a stream ID and a segment ID are used as a key to obtain contact information of peers which can provide the requested segment. Another distributed approach is discovery the content through gossiping. In gossiping, each peer periodically informs a set of randomly selected of other peers about its available content. In general, these different content discovery techniques are equally effective and lead to the desired performance. The choice of content discovery mechanism should not significantly affect the performance of a P2P video caching mechanism, as long as the requests for each stream are divided among corresponding peers in a uniform random fashion. In the rest of this paper and for simplicity reasons, we assume that all PVC mechanisms work in conjunction with a boot-strapping mechanism.

Request Management: To describe how user-requests are managed, we assume that there is a server, called original server, that can provide any video stream that is not available in the P2P cache space. The user-requests are managed as follows. First, a user at a receiver peer identifies a particular video stream s to view through some means, *e.g.,* browsing through an online portal. Then, the receiver peer sends a request to a boot-strap node to locate peers that may have copies of stream s in their local cache. The boot-strap node randomly selects p peers that have at least f percent of all segments of stream s, and report their contact information to peer p. Both p and f are configuration parameters. We refer to the peers reported by the boot-strap node as *sender* or *provider* peers for the stream s. The receiver peer contacts the provider peer with the highest fraction of available content to determine whether that peer is currently online and it has sufficient outgoing available bandwidth to serve the requested stream. If the provider peer is not available or does not have sufficient outgoing bandwidth, the next provider peer is examined until a proper provider is identified or the list is exhausted. When no proper provider is identified, the receiving peer contacts the original server to obtain stream s. Once a provider peer is identified, the receiver requests a new segment from the provider. Any segment that is not available at the provider peer is simply pulled from the original server.

Global vs. Local Popularity: Caching mechanisms often use as input the global popularity of an individual stream. In the context of PVC, capturing the global popularity of individual streams could be prohibitively expensive. We note that in our proposed architecture, the boot-strap node observes all the requests for all video streams and, thus, can potentially capture their global popularity. However, we intentionally, do not rely on the information from boot-strap node so as to be able to deploy our mechanism with other content discovery mechanisms without global popularity knowledge (*i.e.,* DHT or gossiping). In essence, communicating the evolving popularity information with individual peers from boot-strap node could lead to a significant overhead and may not be necessary.

Our goal is to design a PVC mechanism that enables each peer to make segment-based cache replacement decisions without the need to know the global popularity of their cached video streams, in order to accommodate scalability. Indeed, each peer can only rely on its locally observed popularity of the stream to make the cache replacement decisions. More specifically, when the boot-strap node selects provider peers uniformly at random, each provider peer observes a roughly equal fraction of the workload for a stream, within a given time window. Clearly, the longer a provider peer keeps a stream in its cache, the larger the history it observes. Therefore, the local popularity for each stream at individual peers provides a representative (*i.e.,* unbiased) but scaled-down sample of the aggregate requests. Such a local popularity can serve as a reliable indicator of the stream popularity, without requiring to capture the global one.

Controlling the Number of Cached Stream Copies: We propose a relatively simple idea to control the number of copies of each stream to the level that the available bandwidth can cope with the observed demand. Typically, a new stream becomes popular (*i.e.,* hot stream) over a period of time and then its popularity gradually drops (*i.e.,* warm stream) until it becomes very unpopular (*i.e.,* cold stream). To limit the number of replicas for a popular stream, we only increase the number of cached copies while the stream is hot, *i.e.,* its local popularity increases. Note that as the stream s is delivered to more peers, a larger number of its copies becomes available ($n(s)$). Therefore, the observed trend in local popularity of the stream s at each peer depends on the relative growth of its aggregate popularity divided by the number of copies. The intuition for the above simple strategy can be justified as follows: the growth in the popularity of a stream suggests that the increasing number of copies is not adequate to provide sufficient supply bandwidth and therefore, more copies are required.

One important issue is how fast $n(s)$ is updated as more copies are cached. Considering a peer as a provider only when it receives the entire stream, leads to guaranteed availability at the cost of over-reaction (*i.e.,* excess copies) among participating peers. In contrast, considering a peer as a provider as soon as it has received a segment, quickly reflects the effect of the number of copies on the workload. This in turn leads to a faster reaction, and thus smaller number of copies. We refer to this later approach as a "segment relaying" technique and we assume that such a technique is incorporated.

We propose the following method to determine whether the popularity of a cached stream is rising or falling. Each peer keeps track of a short term ($s\_ewma\_irt(s)$) and a long term ($l\_ewma\_irt(s)$) exponentially weighted moving average of the inter-request time for stream s, where the popularity of s is $pop(s) = \frac{1}{s\_ewma\_irt(s)}$. Then, the ratio between the long-term and the short-term moving average indicates whether the popularity of a stream is rising or falling. More specifically, when $m(s) = \frac{l\_ewma\_irt(s)}{s\_ewma\_irt(s)} \geq 1$, the stream is considered hot. Otherwise, the stream popularity has passed its rising phase and becomes cold. We recognize that the stream popularity may oscillate over time, and for this reason the moving average filters out some of the minor variations while the ratio of short to long-term popularity captures significant short-term trends.

Cache Replacement Algorithm: Our proposed PVC mechanism controls the number of copies without any explicit coordination among peers. Toward this end, the replacement algorithm selects b victim segments to be evicted from the cache as follows: first, it considers all the cached streams whose popularity is not growing (*i.e.,* cold stream where $m(s) < 1$). Then, it selects the stream with the lowest popularity ($pop(s)$) and removes b segments of that stream. If the victim stream does not have b segments in the cache, all of its segments are evicted and another victim stream is selected to evict up to the required number. In summary, the replacement algorithm "sequentially thins out" cold and unpopular streams until the required number of segments are evicted.

Note that we discuss the victim selection based on "streams" rather than "segments". The above approach can be extended to segments by considering each segment as an independent object, and similarly selecting victim segments based on segment hotness level and popularity. While this approach tends to keep popular segments of different video streams in the cache, these segments are unlikely to be sufficient to provide a good representation of a video. For example, it is not useful to only cache a few segments of a movie that are very popular. One needs to view at least a "thin out" version of another part of the movie in order to capture the story and interpret the popular part[1]. In this case, segment-based cache replacement should ensure that an adequate number of unpopular segments with proper spacing exist in the cache, and that the content can be meaningfully presented. This requires a more elaborate segment selection strategy and remains as a future work item for us.

Proactive vs Reactive Mode: The number of copies for popular stream can be controlled in a proactive or reactive fashion as follows:

- *Reactive Mode (PVC-R)*: In this approach, which is similar to other caching mechanisms, each peer caches any requested stream. When the cache is full, the above cache replacement algorithm is invoked to free up sufficient space for the new stream. In this approach, once the number of copies of a stream exceeds the demand, some providing peers evict the stream from their cache. The thrashing effect that is caused by new copies at any receiving peer could affect the performance of aggregate cache space.

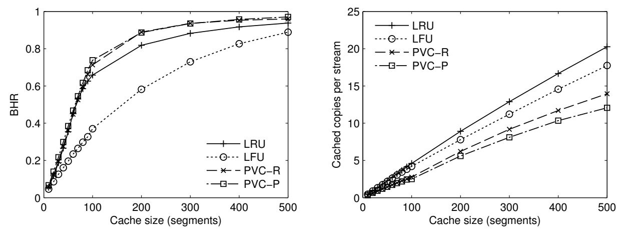
- *Pro-active Mode (PVC-P)*: In this approach, a stream is only cached by the receiving peer if it is still hot. Towards this end, the providing peer informs the receiver whether the stream is hot, based on the available history at the provider. This approach is more conservative in adding more copies and is likely to experience a lower level of thrashing in cached content. The side effect of this approach is that, the request load for a popular stream is mostly observed by the peers that initially have cached the stream.

4. PRELIMINARY EVALUATION

In this section, we present the preliminary evaluation of the proposed PVC techniques, *i.e.,* PVC-R and PVC-P, along with classic cache replacement techniques, *i.e.,* LFU and LRU, that only use the observed request sequence by individual peers, (LRU and LFU in our simulations only rely on local information as opposed to global information that is used in prior studies). Towards this end, we implemented these caching technique in an event-based P2P simulator, that emulates network delay between two hosts (*e.g.,* delay in exchanged messages between peers and/or boot-strap node) using the "King" dataset, as well as outgoing bandwidth of individual peers, and dynamics of peer participations. Furthermore, we use a parametric media workload generator called MediSyn [7] in our simulation. However, we extended MediSyn's features to generate aggregate workload over a wide range of parameters to stress test caching mechanisms in our simulations. The aggregate generated workload is mapped to participating peers uniformly at random.

Evaluation Settings: In MediSyn, the stream popularity is modeled by a generalized Zipf-like distribution [7, 10], which is specified by the number of sessions for the most popular stream n_{max}, and its exponent α (*i.e.,* the slope of

[1]Clearly, such a requirement depends on the type of video content and may not be always needed.



(a) (b)

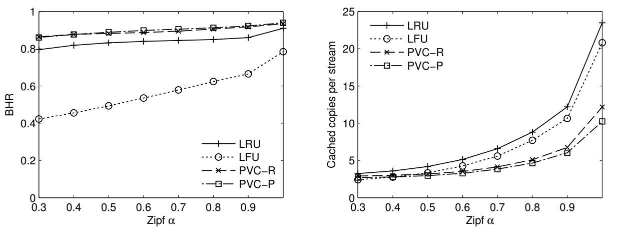
Figure 1: The effect of the cache size: (a) byte-hit ratio, (b) stream replicas

Zipf distribution). MediSyn allows us to control the level of burstiness of all sessions for each stream. Specifically, the inter-request time for each stream is modeled with a log-normal distribution that has two parameters: a mean μ and variance σ. We can increase the burstiness of requests for each stream by setting μ to larger values.

Each scenario is simulated 10 times with different random seeds and the presented results are averaged over them. Each simulation starts with a warm-up phase that lasts until the cache of all peers is full. Then, we start our measurement phase that lasts at least a few times the duration of the warm-up phase. In our simulations, we make a few simplifying assumptions: we do not consider churn among participating peers, we focus on homogeneous scenarios where all peers have the same bandwidth and same cache size, and we assume that all streams are 100 segments, stream rate is 1 Mbps, and all segments of a requested stream are viewed by the receiver peer. This implies that all segments of any stream have the same popularity. These assumptions allow us to examine basic characteristics of cache replacement mechanisms rather than any potential side-effects of complex scenarios. In all caching mechanisms, the boot-strap node provides contact information for five provider peers in response to each request for a stream. We use the following default parameters in our simulations: 1000 streams, 1000 peers, peer outgoing bandwidth of 5 Mbps.

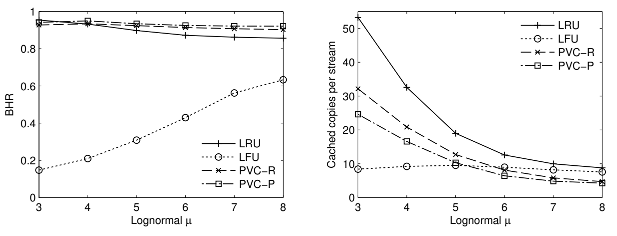
4.1 Effect of Cache Size

Fig. 1(a) illustrates the effect of the cache size at individual peers on the performance of caching mechanisms using Byte-Hit-Ratio (BHR). Fig. 1(b) shows the average number of copies across cached stream as function of cache size to demonstrate the ability of each technique to control the number of replicas. The workload parameters in these simulations are $\alpha = 0.8$ and $\mu = 8$. Fig. 1(a) shows two regions: when the cache size is smaller than the stream size, all caching mechanisms (except LFU) exhibit nearly identical performance that rapidly increases with cache size. In this region, all caching techniques are thrashing due to the small size of individual caches at each peer. When the cache size is larger than the stream size (especially for moderate cache size), both PVCs exhibit 8-10% better performance than LRU. In this latter case, both PVCs are able to cache a couple of streams and maintain some history to behave properly. In particular, Fig. 1(b) clearly demonstrates that both PVCs manage to outperform LRU, while consistently maintaining a smaller average number of replicas for cached streams (especially for PVC-P). In short, both PVCs appear to be able to achieve their design goal by effectively controlling the number of cached copies without causing any



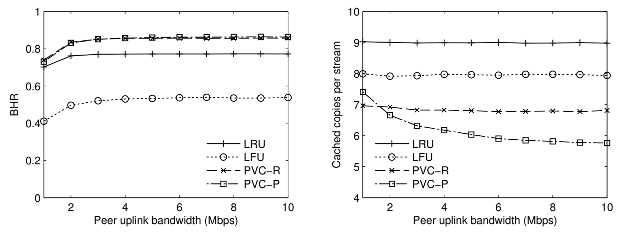
(a) (b)

Figure 2: The effect of stream popularity distribution: (a) byte-hit ratio, (b) stream replicas



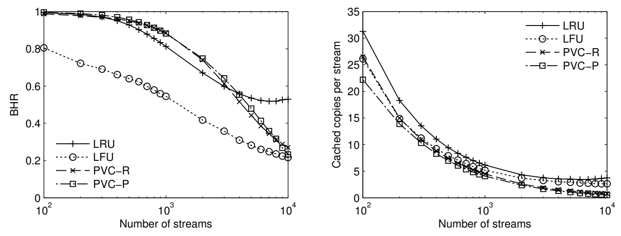
(a) (b)

Figure 3: The effect of session burstiness: (a) byte-hit ratio, (b) stream replicas



(a) (b)

Figure 4: The effect of peer uplink bandwidth: (a) byte-hit ratio, (b) stream replicas



(a) (b)

Figure 5: The effect of the number of streams: (a) byte-hit ratio, (b) stream replicas

performance degradation. Our results also show that LFU exhibits poor performance despite maintaining a relatively larger number of replicas for cached stream compared to other three caching mechanisms.

4.2 Effect of the Workload

We examine the effect of two basic characteristics of workload on the performance of caching mechanisms independently.

Popularity Distribution:: Fig. 2(a) depicts the BHR for all caching mechanisms as a function of parameter α that determines the slope of Zipf popularity distribution *i.e.,* the level of skewness in popularity of streams. In these simulations, the peer cache size is 200 segments, and level of burstiness parameter μ is 8. Fig. 2(b) depicts the average number of replicas across cached copies for the same scenarios. These results show that both PVC mechanisms as well as LRU consistently exhibit high performance for a rather wide range of skewness in popularity distribution even when popularity is mildly skewed (*i.e.,* α is only 0.3). Again, the PVCs slightly outperform LRU while maintaining a smaller number of replicas for cached stream. In particular, for very skewed distribution of popularity, PVC mechanisms (in particular PVC-P) cache half the number of replicas that LRU needs for comparable performance.

Session Burstiness: We decrease the level of burstiness for the requests associated with each stream by increasing the μ. Figures 3(a) and 3(b) show the BHR and number of replicas for each caching mechanism as a function of μ. These figures reveal that both PVC mechanisms and LRU achieve very high performance independent of burstiness of the workload while the number of replicas for PVC mechanisms is much lower than LRU. PVC mechanisms generally outperform LRU (by around 5%) except for a very bursty workload, *i.e.,* flash-crowd. This is due to the fact that LRU offers the most aggressive approach to increase the number of replicas for a popular stream in order to cope with the rapid surge with its demand. However, as we hypothesized, the performance gain of such aggressive increase in the number of replicas is small even for a very bursty workload. In essence, these results suggest that our proposed technique in PVC mechanisms for capturing the changing trend in the popularity of individual stream (*i.e.,* $m(s)$ or the ratio of two exponentially weighted moving average of popularity) can effectively achieve its goal for different level of bursty workloads.

4.3 Effect of Peer Uplink Bandwidth

Fig. 4(a) presents the BHR for all caching mechanisms as a function of peer uplink bandwidth, where the cache size at each peer is 200 segments. This result reveals that the performance is insensitive to the changes in uplink bandwidth as long as it is high enough to server the average demand, which is workload dependent. Again, both PVC mechanisms consistently outperform LRU for all outgoing bandwidth values. Fig. 4(b) depicts the average number of cached streams for all caching mechanisms in these scenarios and reveals an interesting behavior for the PVC-P mechanism. This behavior can be explained as follows. The higher outgoing bandwidth of individual peers enables them to serve a larger number of concurrent requests and observe a larger fraction of requests for each cached stream and thus capture a more comprehensive view of their popularity. If the observed popularity does not exhibit a sudden surge, the minimum number of cached replicas can serve several concurrent requests without prompting receiving peers to cache a new copy. In a nutshell, this result exhibits how PVC-P can effectively utilize serving bandwidth across the minimum number of replicas to avoid generating more copies and prevent thrashing of cached content.

4.4 Effect of the Number of Streams

Fig. 5(a) shows the performance of all caching mechanisms as a function of the number of streams in our dataset. In these simulations we used a cache size of 200 segments, the exponent of the stream popularity $\alpha = 0.67$ and the burstiness parameter $\mu = 8$. To make these scenarios comparable, we keep the popularity distribution and the number of request for the most popular stream fixed across these simulations. This implies that the total number of requests increases with the number of streams and a larger fraction of streams have very low popularity.

Fig. 5(a) indicates that the performance of both PVCs and LRU drops as the number of streams increases beyond 1000. As the number of streams in our dataset increases, the ratio of aggregate cache size to dataset decreases and the diversity among requests grows. These factors can magnify the level of thrashing in cached content and adversely affect the performance. Fig. 5(b) depicts the average number of replicas and shows that PVCs maintain lower number of cached copies for different number of streams. PVC mechanisms generally perform better than LRU except for very large datasets. While we are investigating the changing trend in the behavior of LRU, our hypothesis is that as the number of replicas reduces, the local information in peers with a copy of the stream would be very close to the global popularity of the stream. Thus, LRU accurately identifies the least used stream and replaces it.

5. RELATED WORK

Proxy caching of video has been extensively studied during the past decade as an effective technique to reduce associated network load, decrease startup delay [6], or improve delivered quality [5]. The growing popularity of P2P streaming applications have motivated researchers to explore the idea of P2P video caching in recent years. A few prior studies that examined the idea of PVC [1, 3, 8] have leveraged global cache state in order to make cache replacement decision at individual peers. In these approaches, participating peers should closely coordinate (through direct signaling or frequent updating of state in DHT) in order to coherently manage the P2P cache space to behave as a single large proxy cache. In such proposed cache replacement techniques [1, 3, 8], individual peers require accurate information about the global popularity of each cached stream (based on the observed requests by all peers) as well as the number of cached copies across all peers in order to identify the victim stream to evict from the cache. This clearly has obvious implications on the scalability of these techniques. Furthermore, the potential performance gain of the proposed coordination techniques compared to a distributed and uncoordinated PVC mechanism have not been quantified. To the best of our knowledge, the idea of leveraging the local popularity of each stream at individual peer as a reliable indicator for cache replacement without any global information is proposed in this paper for the first time. Compared to prior studies on PVC, our general goals are: *(i)* to design a PVC mechanism that only relies on local observation by individual peers to perform cache replacement in order to accommodate scalability, and; *(ii)* to quantify any performance penalty of such an uncoordinated PVC mechanism compare to those approaches that require tight coordination for maintaining global state.

6. CONCLUSIONS & FUTURE WORK

This paper presented the examination of key issues in the design of a P2P video caching mechanism. We argued that managing the number of replicas for cached stream is a key design issue that determines the trade-off between the serving bandwidth of individual streams and the level of thrashing in cached content. We discussed that achieving this design goal should be largely feasible with the local notion of popularity and thus may not require global information about cached content across all peers. We devised two straw-man P2P video caching mechanisms, *i.e.,* PVC-R and PVC-P, that try to achieve this goal by leveraging only local peer information. Through simulations, we examined the performance of these two mechanisms with the distributed LRU and LFU across a range of scenarios and workloads. Our simulation results indicate that unlike LFU, both PVC mechanisms and distributed LRU often exhibit a very good performance. PVC mechanisms slightly outperform LRU in most cases while maintaining a smaller average number of replicas for cached streams. Overall, our results suggest that the PVC mechanisms, as well as distributed LRU, seem to offer promising scalable alternatives to existing P2P video caching techniques that require global knowledge.

We are still at an early stage in this project and plan to explore an array of issues including the following: We plan to extend the PVC mechanisms to perform segment-based replacement and incorporate segment-based popularity to keep popular segments in the cache along with evenly spaced unpopular segments among them to ensure proper viewing of popular segments. We plan to conduct sensitivity analysis on our techniques to identify trends in popularity of each stream and possibly use other approaches for this purpose. We also extend the design to incorporate streaming from multiple provider peers, and parallel thinning of multiple cached streams in order to improve overall performance.

7. REFERENCES

[1] L. Guo, S. Chen, S. Ren, X. Chen, and S. Jiang. PROP: a scalable and reliable P2P assisted proxy streaming system. In *IEEE ICDCS*, 2004.

[2] L. Guo, E. Tan, S. Chen, Z. Xiao, and X. Zhang. The Stretched Exponential Distribution of Internet Media Access Patterns. In *ACM PODC*, 2008.

[3] Y. Huang, T. Fu, D. Chiu, J. Lui, and C. Huang. Challenges, Design and Analysis of a Large-Scale P2P-VoD System. *ACM SIGCOMM CCR*, 2008.

[4] N. Magharei and R. Rejaie. Prime: Peer-to-peer receiver-driven mesh-based streaming. *IEEE/ACM TON*, 2009.

[5] R. Rejaie, H. Yu, M. Handley, and D. Estrin. Multimedia proxy caching mechanism for quality adaptive streaming applications in the internet. In *INFOCOM*, 2000.

[6] S. Sen, J. Rexford, and D. Towsley. Proxy Prefix Caching for Multimedia Streams. In *INFOCOM*, 1999.

[7] W. Tang, Y. Fu, L. Cherkasova, and A. Vahdat. Medisyn: A synthetic streaming media service workload generator. In *NOSSDAV*, 2003.

[8] J. Wu and B. Li. Keep Cache Replacement Simple in Peer-Assisted VoD Systems. In *INFOCOM Mini-Conference*, 2009.

[9] K. Wu, P. Yu, and J. Wolf. Segment-based proxy caching of multimedia streams. In *WWW*, 2001.

[10] H. Yu, D. Zheng, B. Zhao, and W. Zheng. Understanding user behavior in large-scale video-on-demand systems. *ACM SIGOPS OSR*, 2006.