Adaptive Proxy-Assisted Cache Scheme for Multiple-version Video Transmission

Chi-Feng Kao^ and Chung-Nan Lee*
Department of Computer Science and Engineering, National Sun Yat-Sen University, Kaohsiung, Taiwan, Republic of China
*E-mail: m9034617@student.nsysu.edu.tw^ and cnlee@mail.cse.nsysu.edu.tw**

ABSTRACT

This work describes the process of efficiently streaming a set of multiple-version videos from a remote server via proxy to a variety of heterogeneous and asynchronous clients that request different quality of the video according to their profiles. The process focuses on reducing transmission cost by caching the optimal version of the video. A set of proxy-based delivery schemes are proposed by integrating the proxy caching with the reactive transmission schemes such as batching or patching. The optimal proxy prefix cache allocation is calculated for each transmission scheme to identify cache length of all versions of each video to minimize the aggregate transmission cost. As a result, it enhances the quality of video transmission. Experimental results demonstrate that for either the batching or patching transmission scheme, the proposed cache scheme can lead to significant transmission cost reduction.

1: INTRODUCTIONS

Streaming media services, including on-line conferences, distance education and movie broadcasting, have recently become popular on the Internet. However, bridging the gap between rich multimedia contents and diverse devices remains a challenge for researchers. Rate-adaptive video should be provided in efficient approaches to satisfy diverse client profiles and network characteristics. Additionally, the high bandwidth requirements and long-lived nature of digital video complicate adaptation tasks. Scalable video technology and video transcoding are both attractive and practical approaches of adhering to such demand. To ensure efficient delivery, proxy cache is generally deployed to reduce the traffic between the content origin and the proxies, and to decrease the transmission cost by caching popular videos at a proxy. In addition, the reactive transmission scheme is another approach to reduce transmission cost. How to integrate these approaches to provide efficient video transmission motivates us to develop this work, which considers the problem of efficiently streaming a set of rate-adaptive videos from a remote server via a proxy by caching the optimal version of each video and using reactive transmission schemes to multiple asynchronous and diverse clients, enabling them to experience playback with a minimum transmission cost.

For reactive transmission schemes, some previous works [2]-[6][18], have focused on employing multicast or broadcast connections to transmit a popular video to multiple asynchronous clients, thus reducing the server load and required network bandwidth. Each scheme has a server only to transmit the on-demand video, when client requests arriving, e.g., batching, patching and stream merging. For the batching transmission scheme [2], those requests arriving close together are batched in time, and then the server multicasts the stream to clients who make the requests. For the patching or stream tapping scheme [3]-[5], the server initializes the entire video transmission sequentially for the first client, while later clients that obtain their future playback data by joining an ongoing multicast stream of the same video, with the clients only receiving the missing prefix via a separate unicast stream. For stream merging [6], all streams (prefix and complete) are delivered via multicast, and clients can join an earlier multicast stream. Eager et al. [18] analyzed the minimum required server bandwidth for any delivery technique that provides immediate real-time delivery to clients increases logarithmically as a function of the client request arrival rate, and proposed a delivery technique, called hierarchical multicast stream merging (HMSM), which has a required server bandwidth that is reasonably close to the minimum achievable required server bandwidth over a wide range of client request rates.

To enable efficient delivery, a proxy cache is typically deployed to lower the traffic between the content origin and the proxies, and to decrease the latency perceived by a user through caching popular objects at a proxy. Caching streaming media is more of a challenge than caching simple web objects, since streaming media has a larger object size. When considering video caching rather than web caching, the video objects may be partially cached onto the proxies, such as video staging [7], [8], prefix caching [9] and selective frame caching [10]. Verscheure et al. [11] studied partial caching based on server scheduling. Zhang et al. [12] considered some video prefetching algorithms in which popular video objects are prefetched onto the proxies to reduce the WAN traffic.

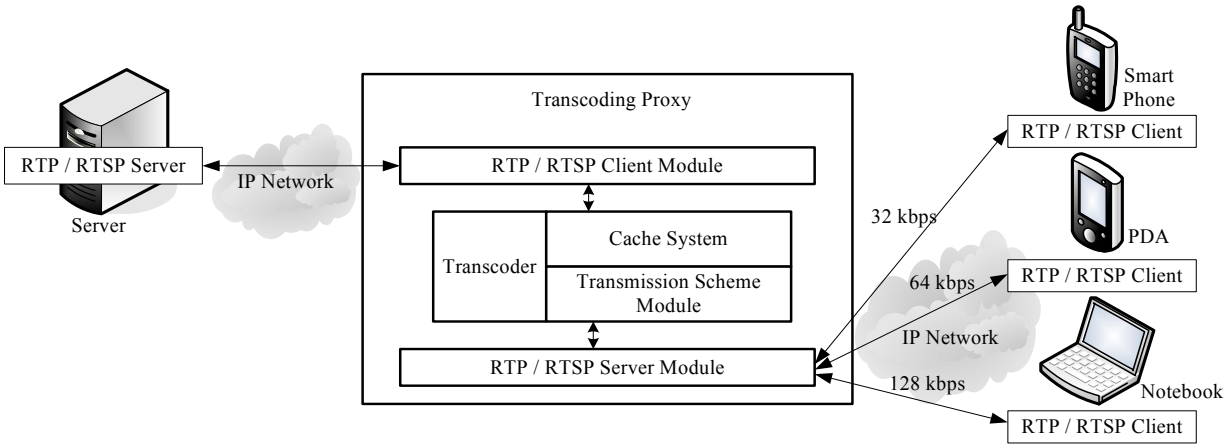

Fig. 1. Streaming multiple-version videos on the Internet: multiple-version video streams from a remote server are transmitted to a variety of clients. The proxies located close to the clients perform prefix caching based on the client profile.

Lee et al. [13] adopted video summaries to cache the media stream. Rejaie et al. [14] developed a layered video caching mechanism to maximize the quality of popular streams delivered to interested clients. Ma et al. [15] designed a progressive video caching policy, in which each video is cached at several levels according to cached data sizes and required WAN bandwidths. Kangasharju et al. [22] proposed some heuristics to determine which videos and which layers in the videos should be cached in order to maximize the revenue from the streaming service. However, most existing investigations have concentrated on unicast delivery of an individual stream to each client without considering the impact of the transmission scheme on the cache efficiency.

Tang et al. [20] combined caching with the transcoding of streaming objects. Algorithm FVO (Full Version Only) caches only the full, original version and serves requests to lower versions with transcoding. However, the transcoded objects are not cached. TVO (Transcoded Version Only) always caches the transcoded objects, and if a request does not yield an exact hit, the full version is fetched from the origin to generate a transcoded version. Shen et al. [19], developed a transcoding-enabled caching (TeC) system that performs transcoding and caching for multimedia efficient delivery to heterogeneous network users. However, the TeC strategy fails to make optimal cache replacement decisions, and uses only the least recently used (LRU), as the replacement policy.

Although Eager et al. [16], and Sen et al. [17] combined caching with transmission schemes, they focused on utilizing nonreactive schemes such as periodic broadcast. Ramesh et al. [21] combines caching with video transmission on networks with end-to-end multicast/broadcast capability. Wang et al. [1] explored the combination of proxy prefix caching and reactive transmission to reduce the transmission cost of multiple heterogeneous videos, which is most relevant to this work. However, in [1], only the transmission scheme and proxy caching for a single version of the video was considered, the multiple-version videos as well as the transcoding-enabled proxy are yet to be explored.

This work to our knowledge is the first work to combine proxy prefix caching and reactive transmission schemes for multiple-version videos to determine the optimal version to be cached, the most appropriate prefix cache length and the best batch/patch threshold for a given transmission scheme.

The remainder of this paper is organized as follows. Section 2 introduces the proposed system environment and parameter definitions. Section 3 then presents the proposed optimal proxy prefix caching. Next, Section 4 introduces a set of proxy-assisted reactive transmission schemes for multiple-version videos. Performance analysis is presented in Section 5 and, finally, conclusions are drawn in Section 6.

2: SYSTEM ENVIRONMENT AND PARAMETER DEFINITION

Figure 1 shows the system environment consisting of a variety of clients receiving videos streamed across the Internet from a server via a proxy. Clients receive different versions of the video based on their capabilities, such as storage capacity, rendering power and network bandwidth. The proxy determines which version and prefix length of each video should be cached to achieve the maximum benefit. The proxy intercepts the client request. If a prefix of version of the requested videos is already stored at the proxy, then the proxy streams the prefix directly to the client. If the complete video is not stored at the proxy, then the proxy informs the server to transmit the suffix of the video, and relays the incoming data to the client. The server has the video with multiple versions, and higher versions can be transcoded into lower versions. The proxy can transcode a higher version into a lower version to meet the clients' requirements. Thus, the proxy only cache one version of a video to achieve better cache efficiency.

This work employs a single server and a single proxy to verify the proposed schemes, which can be directly applied to multiple-proxy content distribution networks where the server adopts unicast connections to the proxies. Each proxy serves a group of clients which do not overlap, and the proxies do not interact with each other.

Table 1 lists the key parameters used in analysis and simulation.

Table 1
Analysis and Simulation Parameters

Parameters	Definition
N	Number of videos
L_i	Length of video i (sec)
l_i	Cached length of video i (sec)
$b_{i,j}$	Bit rate of version j of video i (kbps)
u	Caching grain (kbits)
$n_{i,j}$	Size of version j of video i (units)
f_i	Access probability of video i
p_j	Access probability of version j
$\lambda_{i,j}$	Access rate of version j of video i
λ	Aggregate request arrival rate
S	Proxy cache size (units)
v_i	Cached version of video i
c_s	Transmission cost on server-proxy path (per bit)
c_p	Transmission cost on proxy-client path (per bit)
$C_i(v_i, l_i)$	Transmission cost per unit time for all versions of video i when a length l_i seconds prefix of version v_i of video i is already cached at the proxy

Assume that a server has a repository of N constant-bit-rate (CBR) videos as in [1]. Each video has R versions. The higher versions can be transcoded into lower versions. Additionally, suppose that the access probabilities of each version of all videos and the aggregate access rate to the video repository are known as a priori. These parameters can be obtained by monitoring a real system. Let f_i denote the access probability of video i, $\sum_{i=1}^{N} f_i = 1$, which is a measure of the relative popularity of a video. Let p_j denote the access probability of version j, $\sum_{j=1}^{R} p_j = 1$, which is dependent on the client device distribution (e.g. notebooks tend to request higher versions with better quality, while PDA tends to request lower versions). The access probability of version j of video i is $f_i \times p_j$. Let $\lambda_{i,j}$ denote the access rate of version j of video i, and λ represent the aggregate access rate to the video repository. Thus, $\lambda_{i,j} = \lambda \times f_i \times p_j, 1 \le i \le N, 1 \le j \le R.$

The smallest unit of cache allocation is a caching grain of size u kbits, and all allocations are set to multiples of this unit. The size of video i and the proxy cache size is a multiple of a caching grain. Assume that version j of video i has playback bandwidth $b_{i,j}$ kbps. Version j of video i has size $n_{i,j}$ units and length L_i seconds, and $n_{i,j}u = b_{i,j}L_i$. Assume that the proxy can store S units, also is $S \times u$ kbits. The storage vector $Sv = (l_1, l_2, \ldots, l_N)$ specifies that a length l_i seconds prefix of video i is cached at the proxy, $i=1,2,\ldots,N$. The version vector $Vv = (v_1, v_2, \ldots, v_N)$ specifies the cached version v_i of video i, $i=1,2,\ldots,N$. Notably, the videos cached at the proxy cannot exceed the storage capacity of the proxy, that is, $\sum_{i=1}^{N} l_i b_{i,v_i} \le Su$, where b_{i,v_i} denotes the bit rate of the cached version v_i of video i. Let c_s and c_p respectively represent the transmission costs of one bit of video data on the server-proxy path and the proxy-client path. The objective is to develop appropriate transmission and caching schemes that minimize the mean transmission cost per unit time aggregated over all versions of the videos in the repository, that is $\sum_{i=1}^{N} C_i(v_i, l_i)$, where $C_i(v_i, l_i)$ denotes the transmission cost per unit time for all versions of video i when a length l_i seconds prefix of version v_i of video i is already cached at the proxy.

On receiving a client request for a version of a video, the proxy determines a transmission schedule based on the transmission scheme utilized. The transmission scheme determines when and on what transmission channel the proxy transmits the video. The proxy also requests the suffix from the server, and transmits a reception schedule from the proxy to the client, when and from which transmission channel the client should receive the data. Significantly, a client may have to receive data from multiple transmission channels simultaneously. Frames received ahead of their playback times are stored in a client buffer. Finally, the server only needs to transmit a suffix of the video requested by the proxy via a unicast connection. Hence, the proposed delivery techniques can be applied to unicast-based media servers.

3: OPTIMAL PROXY PREFIX CACHE ALLOCATION

This section presents a general approach to calculate the optimal proxy prefix cache allocation for a proxy-assisted multiple-version video transmission scheme. For a given transmission scheme, the average transmission cost per unit time for all versions of video i, $C_i(v_i, l_i)$, is a function of the length l_i seconds prefix of version v_i of video i already cached at the proxy, where $0 \le l_i \le L_i$. The total length of video i is L_i seconds, and the highest version R of video i has size $n_{i,R}$ units. Let $A_i = \{m_i \mid 0 \le m_i \le n_{i,R}\}$ denote the set of possible prefixes of video i, where m_i is the size, and $m_i u/b_{i,k}$ is the length in seconds of a possible prefix of version k of video i, $1 \le k \le R, m_i u/b_{i,k} \le L_i$. Let $saving(m_i)$ denote the maximal saving in transmission cost from caching an m_i-unit prefix of video i than caching no prefix of the video at the proxy. The transmission cost of video i when

caching no prefix of video i is $C_i(0,0)$. Given m_i units to cache the video i, the optimal version of video i must be determined to be cached to maximize the transmission cost saving ($saving(m_i)$) over all versions of video i.

$$saving(m_i) = \max_{1 \le k \le R} \{C_i(0,0) - C_i(k, \frac{m_i \times u}{b_{i,k}})\},$$

$$subject\ to\ \frac{m_i \times u}{b_{i,k}} \le L_i.$$

Based on the above equations, the optimal version v_i of video i can be found to be cached to maximize the saving, given m_i units to cache the video i. Next, the purpose is to maximize the aggregate savings, i.e., minimize the aggregate transmission cost over all versions of videos. The optimization problem can therefore be formulated as

$$maximize: \sum_{i=1}^{N} saving(m_i),\ subject\ to\ \sum_{i=1}^{N} m_i \le S, m_i \in A_i,$$

where the size of the proxy is S units. The dynamic programming algorithm is then adopted to find the optimal allocation. Let M=($I_{x,y}$) be a two-dimensional matrix, where entry $I_{x,y}$ represents the maximum saving in transmission cost for the first x videos in a proxy cache of size y, and has the following form:

$$I_{x,y} = \begin{cases} \max_{\forall m_i \in A_i} \{I_{x\text{-}1,y\text{-}m_i} + saving(m_i)\}, & x > 0 \\ 0, & x = 0 \end{cases}$$

This matrix is filled in row order from $I_{0,y}$, $y = 0,\ldots,S$. The value $I_{N,S}$ denotes the maximum saving in transmission cost when all N videos have been considered. The minimum transmission cost is given by $\sum_{i=1}^{N} C_i(0,0) - I_{N,S}$, since the saving is relative to caching nothing at the proxy. The optimal cache allocation can be determined as follows. For each entry, store a pointer to an entry from which this current entry is computed. The optimal allocation is obtained by tracking the pointers back from the entry $I_{N,S}$.

4: PROXY-ASSISTED MULTIPLE-VERSION VIDEO TRANSMISSION SCHEMES

This section presents the optimal proxy prefix caching for a given reactive multiple-version video transmission scheme. For each scheme, a closed-form expression for the transmission cost $C_i(v_i,l_i)$ associated with video i is derived, when a length l_i seconds prefix of version v_i of video i is already cached, $1 \le i \le N$.

The transmission cost $C_i(v_i,l_i)$ is employed to determine the proxy prefix cache allocation for each video that minimizes the aggregate transmission cost. These transmission schemes are general and applicable to any client arrival process. A Poisson arrival process, which is a conservative assumption for reactive schemes [18], is adopted to estimate the transmission costs. Wang et al. [1] proposed three schemes:

(1)SBatch utilizes the video prefix cached at the proxy, so that multiple requests share a single server-to-proxy transmission to save transmission costs.
(2)UPatch employs patching for the suffix.
(3)MPatch exploits prefix caching at the proxy under a multicast-enabled proxy-to-client environment.

Though these schemes perform well for the single version video, the multiple-version videos and heterogeneous clients have not been considered. Therefore, new schemes are proposed for multiple-version videos transmission. In this work, the server-to-proxy path and proxy-to-client path are only unicast-enabled, the multicast-enabled proxy-to-client path is not considered since the transmission bottleneck is mainly from the WAN (Server-to-proxy path). However, it is expected that if proxy-to-client is multicast-enabled, the transmission cost can be reduced. In addition, we assume that clients always request playback from the beginning of a video, as in [1].

4.1 Version-dominated Batching (VBatch)

VBatch is proposed for multiple-version video transmission. Suppose that the first request for version k ($k \le v_i$) of video i arrives at time 0. The length l_i seconds prefix of version v_i of video i is already cached at the proxy. For these requests arrive in time $(0,l_i]$ and requested versions are equal to or lower than the cached version v_i, the proxy immediately start transcoding the cached version into these requested versions and transmitting the prefix of requested versions of video i to the clients as illustrated in Fig. 2. VBatch schedules the transmission of the suffix of version v_i from the server to the proxy as late as possible, just in time to guarantee continuous playback for the clients. That is, the first frame of the suffix of version v_i is scheduled to reach the proxy a little earlier than time l_i, which is the length of the prefix of version v_i. Then, the proxy transcoding the version v_i suffix into these requested versions and simply forwards these suffixes of requested versions (of length L_i-l_i) to these requested clients, and no new suffix transmission is required from the server. Multiple requests for the suffix of video i are thus batched together. Note that though the first requested version is k ($k \le v_i$), the version v_i suffix, rather than version k, is transmitted from server to proxy, since version v_i can be transcoded into version k and can be shared by later requests for equal to or lower than version v_i. Thus, the average transmission cost per unit time for delivering version k ($k \le v_i$) of video i, based on a Poisson arrival process, is given by

$$\cos t1_i(v_i,l_i) = \frac{c_s(L_i - l_i)b_{i,v_i}}{1 + l_i \sum_{k=1}^{v_i} \lambda_{i,k}} \times \sum_{k=1}^{v_i} \lambda_{i,k} + \sum_{k=1}^{v_i} c_p L_i b_{i,k} \lambda_{i,k}$$

If requested version k is higher than the cached version v_i, the proxy must immediately request an entire version k of video i from the server to proxy, and the transmission cannot be shared. In this case, the

transmission cost per unit time for delivering version $k(k>v_i)$ of video i is formulated as follows.

$$\cos t2_i(v_i,l_i) = \sum_{k=v_i+1}^{R}(c_s + c_p)L_i b_{i,k}\lambda_{i,k}$$

In summary, when the length l_i seconds prefix of version v_i of video i is already cached at the proxy, the transmission cost per unit time for all versions of video i is formulated as follows.

$C_i(v_i,l_i) = \cos t1_i(v_i,l_i) + \cos t2_i(v_i,l_i)$.

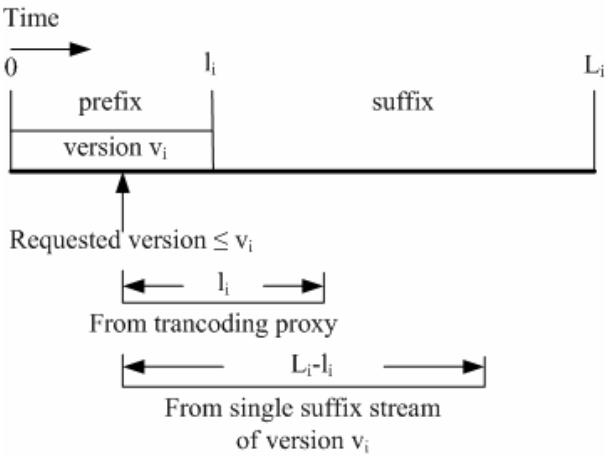

Fig. 2. Version-dominated batching with prefix caching (VBatch)

4.2 Version-dominated Patching(VPatch)

This section presents a VPatch scheme that utilizes patching for multiple-version video transmission. Suppose that the first request for version k ($k \le v_i$) of video i arrives at time 0, the length l_i seconds prefix of version v_i of video i is already cached at the proxy, and the version v_i suffix reaches the proxy from the server a little earlier than time l_i,as illustrated in Fig. 3. For these requests arrive in time $(0, l_i]$ and requested versions are equal to or lower than the cached version v_i, the transmission scheme is the same as VBatch scheme. In another case, suppose that later request for version k ($k \le v_i$) of video i occurs at time t_2, $l_i<t_2<L_i$. The proxy can schedule a transmission of the complete suffix of version v_i at time t_2+l_i from the server. Alternatively, a patch of $[l_i, t_2)$ of the version k suffix can be scheduled from the server, since segment $[t_2, L_i]$ of version v_i has already been scheduled to be transmitted and can be transcoded to meet the version k ($k \le v_i$) request. The decision to transmit a complete suffix or a patch depends on a suffix threshold G_i, measured from the beginning of the suffix. If one request for version k ($k \le v_i$) arrives within G_i from when the nearest complete transmission of the version v_i suffix was started, then the proxy schedules a version k patch from the server for the version k request. Otherwise, a complete transmission of the version v_i suffix is started. The suffix threshold G_i is chosen to minimize the transmission cost. Assuming a Poisson arrival process, the average transmission cost per unit time for delivering version k ($k \le v_i$) of video i is given by

$$\cos t3_i(v_i,l_i) = (c_s \frac{\sum_{k=1}^{v_i}\frac{\lambda_{i,k}G_i^2 b_{i,k}}{2} + (L_i - l_i)b_{i,v_i}}{1+\sum_{k=1}^{v_i}\lambda_{i,k}(l_i+G_i)}) \times \sum_{k=1}^{v_i}\lambda_{i,k} + c_p\sum_{k=1}^{v_i}\lambda_{i,k}L_i b_{i,k}$$

,

where the average patch length for the request arriving in time $(l_i, l_i+G_i]$ is $G_i/2$ based on a Poisson arrival process and the number of requests for version k within G_i is $\lambda_{i,k}G_i$. Thus, the total patch bits for version k per unit time is $\frac{\lambda_{i,k}G_i^2 b_{i,k}}{2}$.

If requested version k is higher than the cached version v_i, the proxy must immediately request an entire version k of video i from the server to proxy, and the transmission cannot be shared. In this case, the transmission cost per unit time for delivering version $k(k>v_i)$ of video i is the same as $\cos t2_i(v_i,l_i)$.

In summary, when the length l_i seconds prefix of version v_i of video i is already cached at the proxy, the transmission cost per unit time for all versions of video i is formulated as follows.

$C_i(v_i,l_i) = \cos t3_i(v_i,l_i) + \cos t2_i(v_i,l_i)$.

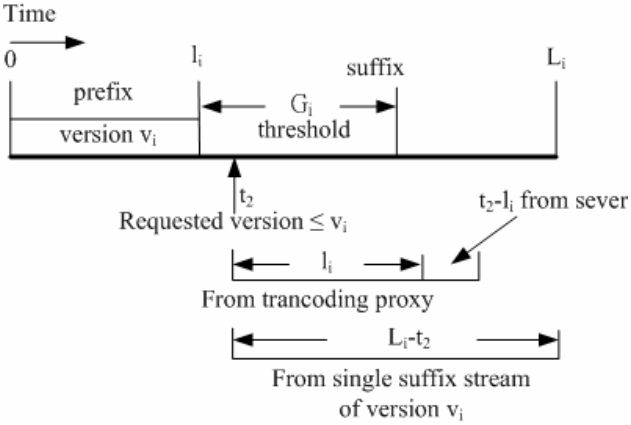

Fig. 3. Version-dominated patching with prefix caching (VPatch)

5: PERFORMANCE EVALUATION

This section presents the performance evaluation for the above caching and transmission schemes, which was based on a repository of 100 CBR video clips whose popularity followed a Zipf distribution with a skew factor α of 0.271 [2]. In the multiple-version case, all videos were assumed to be two hours long with five versions. The bit rate of version 1 was 32kbps, version 2 was 64kbps, version 3 was 128kbps, version 4 was 256kbps and version 5 was 512kbps. Let c_s=1 and c_p=0, the transmission cost is the required server bandwidth per second. The default cache capacity of the proxy was assumed to be 20% of total sizes of all 32kbps versions of all videos. The caching grain was set to three minutes of 32kbps video version.

5.1 The Impact of Access Patterns

This set of experiments examined the performance of transmission schemes in various access patterns. The access to different versions is modeled as follows. As in Shen et al. [19], assume that the accesses to different versions of the videos follow a normal distribution with mean m, where version m denotes the dominant version. The access probability of a version x is

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(x-m)^2/2\sigma^2},$$

where m and σ^2 represent the mean and variance, respectively, of how the versions are accessed. When σ is small, most of the accesses tend to be one version (m). When σ is large, the accesses are evenly distributed among the different versions. The access model examines the concentration level of access versions for the impact of transmission performance. In this experiment, m = 2 (i.e. the version 3(128kbps) is the dominant version when σ is small).

Figure 4 depicts the required server bandwidth of transmission schemes under different access patterns when the arrival rate λ is 100 requests/min. Clearly, the patch-based schemes have lower required bandwidth than the batch-based ones. For instance, when the value of σ = 1.4, the required bandwidth of VPatch, and VBatch are 19.3%, and 65.2%, respectively, of the original bandwidth that, in the following, called as "original bandwidth". When the value of σ = 1.4, the VBatch and VPatch schemes require 167.32MB/s and 49.62 MB/s bandwidth. Notably, as the value of σ rises, the accesses become more evenly distributed among different versions, and required server bandwidth increases, since the limited cache space can not accommodate all popular objects when the accesses are dispersed. As the value of σ rises, the required bandwidth of batch-based schemes increases more than that of patch-based schemes. This is because that the patch-based schemes can dynamically tune up the threshold G according to access patterns to minimize the required server bandwidth.

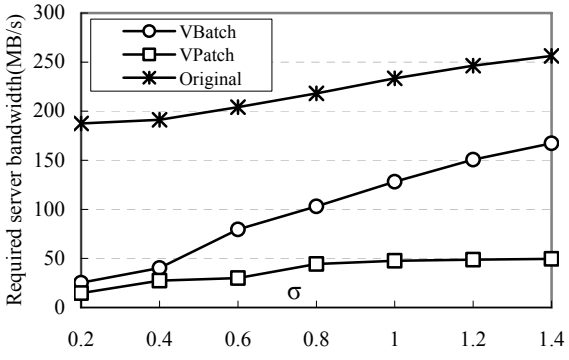

Fig. 4. Required server bandwith versus access pattern when λ=100/min.

5.2 The Impact of Proxy Cache Size

Figure 5 depicts the required server bandwidth of transmission schemes under various cache sizes when λ = 100 requests/min, and the value of σ is set to be 1.4. The required bandwidth of patch-based schemes is clearly lower than that of batch-based schemes over all the range of proxy cache sizes. The patch-based schemes can save much bandwidth even if the cache size is very small. For example, when cache size = 10% of total sizes of all minimal versions, the VPatch scheme only needs 20% of the original bandwidth. The required bandwidth reduction is significantly less for the patch-based schemes than that of the batch-based schemes as the cache size increases. Thus, increasing cache size is more beneficial when adopting the batch-based transmission schemes. For instance, when cache size = 10% of total sizes of all minimal versions, the VBatch scheme needs 77.1% of original bandwidth. However, when cache size = 90% of total sizes of all minimal versions, it only needs 33.1% of original bandwidth.

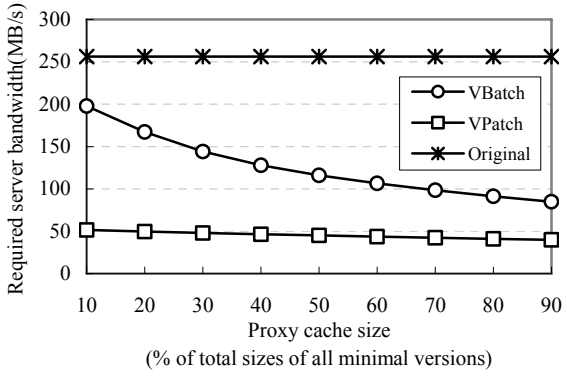

Fig. 5. Required server bandwidth versus proxy cache size when λ=100/min.

5.3 The Impact of Arrival Rate

Figure 6 illustrates the required server bandwidth of transmission schemes as the arrival rate increases from 10 to 100 requests/min where the cache size is 20% of total size of all minimal versions and the value of σ is set to be 1.4. Comparing patch-based schemes with batch-based schemes, the gap significantly increases as the arrival rate rises. When the arrival rate is 10 request/min, the VPatch scheme needs 65.1% of required bandwidth of VBatch. When the arrival rate is 100 requests/min, the required bandwidth of the VPatch scheme becomes 29.6% of that of VBatch, respectively. This finding clearly demonstrates that using patch-based schemes can produce more bandwidth savings than batch-based schemes as the arrival rate increases.

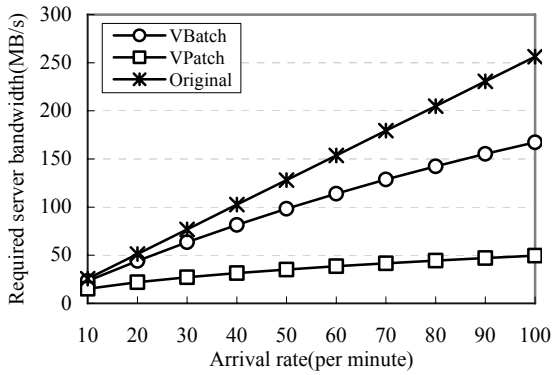

Fig. 6. Required sever bandwidth versus arrival rate.

6: CONCLUSIONS

This work has proposed an approach for determining an optimal proxy prefix cache allocation for multiple-version videos that minimizes the required server bandwidth for a given transmission scheme. Two transmission schemes: the version-dominated batching scheme, and the version-dominated patching scheme are proposed. Experimental results demonstrate that, an adaptive proxy-assisted transmission scheme can result in significant required server bandwidth reduction. We expect such schemes to provide higher reception quality (more detailed versions) at the client side than without schemes, due to lower server bandwidth requirements. In other words, given the same bandwidth constraints, the proposed schemes can provide better viewing quality. In the future we will study the cache scheme for layered video.

REFERENCES

[1] B. Wang, S. Sen, M. Adler, and D. Towsley, "Optimal proxy cache allocation for efficient streaming media distribution," *IEEE Trans. Multimedia*, vol. 6, no. 2, pp. 366-374, Apr. 2004.

[2] C. Aggarwal, J. Wolf, and P. Yu, "On optimal batching policies for video-on-demand storage servers," in *Proc. IEEE Int. Conf. Multimedia Computing and Systems*, pp. 253–258, June 1996.

[3] S. Carter and D. Long, "Improving video-on-demand server efficiency through stream tapping," in *Proc. Int. Conf. Computer Communications and Networks*, Las Vegas, NV, 1997.

[4] L. Gao and D. Towsley, "Threshold-based multicast for continuous media delivery," *IEEE Trans. Multimedia*, vol. 3, no. 4, pp. 405–414, Dec. 2001.

[5] K. Hua, Y. Cai, and S. Sheu, "Patching: a multicast technique for true video-on-demand services," in *Proc. ACM Multimedia*, Sept. 1998, pp. 191–200.

[6] D. Eager, M. Vernon, and J. Zahorjan, "Optimal and efficient merging schedules for video-on-demand servers," in *Proc. ACM Multimedia*, Nov. 1999, pp. 199–202.

[7] W. Ma and D. H. C. Du, "Reducing bandwidth requirement for delivering video over wide area networks with proxy server," *IEEE Trans. Multimedi*a, vol. 4, no. 4, pp. 539–550, Dec. 2002.

[8] Y. Wang, Z.-L. Zhang, D. H. C. Du, and D. Su, "A network-conscious approach to end-to-end video delivery over wide area networks using proxy servers," in *Proc. IEEE INFOCOM*, San Francisco, CA, vol. 2, pp. 660-667, Apr. 1998.

[9] S. Sen, J. Rexford, and D. Towsley, "Proxy prefix caching for multimedia streams," in *Proc. IEEE INFOCOM*, New York, Mar. 1999, pp. 1310–1319.

[10] Z. Miao and A. Ortega, "Scalable proxy caching of video under storage constraints," *IEEE J. Select. Areas Commun.*, vol. 20, no. 7, pp. 1315–1327, Sept. 2002.

[11] O. Verscheure, C. Venkatramani, P. Frossard, and L. Amini, "Joint server scheduling and proxy caching for video delivery," *Computer Communications*, vol. 25, no. 4, pp. 413–423, Mar. 2002.

[12] Z.-L. Zhang, Y. Wang, D. H. C. Du, and D. Su, "Video staging: A proxyserver-based approach to end-to-end video delivery over wide-area networks," *IEEE/ACM Trans. Networking*, vol. 8, no. 4, pp. 429–442, Aug. 2000.

[13] S.-J. Lee, W.-Y. Ma, and B. Shen, "An interactive video delivery and caching system using video summarization," *Computer Communications*, vol. 25, no. 4, pp. 424–435, Mar. 2002.

[14] R. Rejaie, H. Yu, M. Handely, and D. Estrin, "Multimedia proxy caching mechanism for quality adaptive streaming applications in the internet," in *Proc. IEEE INFOCOM*, Tel Aviv, Vol 2, pp. 980-989, Mar. 2000.

[15] W. Ma, and D. H. C. Du, **"**Design a progressive video caching policy for video proxy servers**,**" *IEEE Trans. Multimedia*, vol. 6, no. 4, pp. 599–610, Aug. 2004.

[16] D. Eager, M. Ferris, and M. Vernon, "Optimized regional caching for on-demand data delivery," in *Proc. Multimedia Computing and Networking (MMCN '99)*, San Jose, CA, Jan. 1999.

[17] S. Sen, L. Gao, and D. Towsley, "Frame-based periodic broadcast and fundamental resource tradeoffs," in *Proc. IEEE Int. Performance Computing and Communications Conf.*, Phoenix, AZ, Apr. 2001.

[18] D. Eager, M. Vernon, and J. Zahorjan, "Minimizing bandwidth requirements for on-demand data delivery," *IEEE Trans. Knowledge and Data Engineering*, vol. 13, no. 5, pp. 742–757, Sept./Oct. 2001.

[19] B. Shen, S. J. Lee, and S. Basu, "Caching strategies in transcoding-enabled proxy systems for streaming media distribution networks**,"** *IEEE Trans. Multimedia*, vol. 6, no. 2, pp. 375–386, Apr. 2004.

[20] X. Tang, F. Zhang, and S.T. Chanson, "Streaming media caching algorithms for transcoding proxies," in Proc. *Intl Conf. on Parallel Processing*, pp. 287 – 295, 2002.

[21] S. Ramesh, I. Rhee, and K. Guo, "Multicast with cache (Mcache): an adaptive zero-delay video-on-demand service," *IEEE Trans. Circuits and Systems for Video Technology*, vol.11, Issue 3, pp. 440 – 456, March 2001.

[22] J. Kangasharju, F. Hartanto, M. Reisslein, and K. W. Ross, "Distributing Layered Encoded Video through Caches," *IEEE Transactions on computer*, vol. 51, no. 6, pp. 622-636, June 2002.