An Effective Proxy Caching and Rate Control Mechanism for Transporting Multimedia Streams over the Internet

*Kun-Yuan Chang, Yi-Hsuan Lee, Tzu-Han Tsao, and Cheng Chen**
Department of Computer Science and Information Engineering
National Chiao Tung University, Hsinchu, Taiwan, R.O.C.
Tel: (8863) 5712121 ext: 54734, Fax: (8863) 5724176
E-mail: {kychang, yslee, thtsao, cchen}@csie.nctu.edu.tw

Abstract

Recently, the demand for multimedia service is increasing dramatically, but this service may cause the bottleneck of Internet due to the bandwidth requirement and storage constraint of multimedia data. Streaming proxy server, much similar to general proxy server, is a useful architecture that can ease the load of original video server and shorten the response time while serving clients. However, existing techniques for caching documents in general proxy server is not appropriate for caching multimedia data that requires much more storage size. Hence, in this paper, we propose a streaming proxy architecture that stores portions of each video program according to its popularity. Additionally, we use Fibonacci function to divide the video data into variable-sized segments to speedup the data allocation and cache replacement. During the transmission of video data for each session, we also propose a transmission rate control mechanism based on exponential function to achieve higher bandwidth utilization and lower data loss rate. According to our preliminary analysis and simulation results, our techniques have positive results compared with other conventional methods.

Keywords: *Video proxy, Streaming proxy, Replacement, Rate control*

1. Introduction

Recently, the demand for multimedia service is increasing dramatically [4]. It not only requires much network resources, but also increases the loads of web sites. To reduce the response time and bandwidth usage due to the multimedia application, a streaming proxy is needed for forwarding and caching video data just like conventional proxies. However, existing techniques for caching documents in web is not appropriate for multimedia data with much more storage size. To improve the performance of the streaming proxy, it needs an effective cache replacement mechanism. Thus, in this paper, we propose a streaming proxy architecture that stores portions of video programs according to its popularity, and use Fibonacci function to divide the video data to speedup the data allocation and cache replacement. We also Our main goals are to increase the byte-hit ratio in the streaming system and decrease the amount of data migrations. Meanwhile, the execution time for making replacement decision should be acceptable compared with other methods.

Besides issues for cache replacement, we also consider the problem of transmission rate control during each service session. Due to the large amount of data transmitted in each session, increasing bandwidth utilization and decreasing data loss rate become much important. To improve conventional rate control methods, we propose an *Exponential rate control mechanism* to adjust the transmission rate of video data. Using this mechanism we can increase bandwidth utilization and decrease data loss rate.

The remaining part of our paper is organized as follows. In section 2, we give related work about cache replacement and rate control. In section 3, we describe our system service flow and replacement scheme in detail. In section 4, we show how to improve the transmission rate control scheme by using an exponential function. We also give preliminary analyses of our replacement mechanism. The simulation environment and evaluating results are illustrated in section 5. Finally, we give some conclusions and future work in section 6.

2. Fundamental Background and Related Work

In this section, we give a brief survey about some related works of streaming caching techniques and rate control problem during actual data transmission.

2.1 Survey of Replacement Methods

In this subsection, we first introduce FGS (*Fine-Grained Scalable Video Coding*), which is the video coding method used in the MPEG-4 standard [10]. Next, we introduce the prefix caching mechanism and some replacement schemes based on it.

Because of the wide variation of available bandwidth over Internet, there is a need for scalable video coding methods and corresponded flexible streaming approaches that are capable of adapting to changing network conditions in real time. FGS framework strikes a good balance between coding efficiency and scalability while maintaining a very flexible and simple video-coding structure. FGS encoding technique is designed to cover any desired bandwidth range while maintaining a very simple scalability structure.

Due to the video objects with much larger size, there should be come new mechanisms to increase the hit rate and reduce the start-up delay of each request. Sen et al. [5] proposed an idea that network service providers deploy proxies that cache the initial frames of popular videos. Upon receiving a request for the stream, the proxy initiates transmission to the client and simultaneously requests the remaining frames from the original video server. This approach assumes that users see the video from the beginning and few users view the media playback till the last segments.

Lim et al. [6] assumes that every video object in the proxy has a caching utility, which is derived by the popularity divided by the amount of data stored in the storage. The popularity of a stream is the total amount of data played back by clients during a time interval. Caching or replacement of each stream data is performed in granularity of fixed size segment based on the popularity. When the size of remaining storage is not enough, the video with smallest caching utility is replaced.

In order to increase the efficiency of cache replacement decision, Yu [7] introduced the exponential distribution approach for data placement. Blocks of a media object received by the proxy server are grouped into variable-sized, distance-sensitive segments. The segment size increases exponentially from the beginning segment. The purpose of this segmentation mechanism is that it can quickly discard a big chunk of a cached media object that was once hot but has since turned cold. Compared to the fix-sized allocation scheme, this method is more efficient. On the other hand, because the quickly increasing of block size for each block, it may reduce the refinement of caching replacement.

2.2 Survey of Rate Control Techniques

The adjusting schemes of transmission rate in TCP network nowadays mostly follow the *Additive Increase / Multiplicative Decrease* (AIMD) mechanism [8]. This mechanism generates great fluctuation and possible poor user perspectives when applying to transmitting multimedia data. To improve the effect of AIMD, Chung et al. [9] proposed a new mechanism to smooth the variation of transmission rate. If there is no congestion in the network, the server gradually increases the transmit rate until the packet loss rate exceed the predefined threshold. When network congestion occurs, the client goes into the congestion mode and sends feedbacks to the server to decrease the transmission rate. Although this approach can smooth the variation of rate when closing to the link bandwidth, it may still cause bandwidth usage fluctuation and lower bandwidth utilization when the loss rate exceeds the threshold.

3. Fibonacci Replacement Scheme

3.1 Basic Service Flow

The operations of our streaming proxy server are quite similar to conventional proxy servers, but with some differences that we will state here in detail. The service flow diagram is shown in Figure 1. When a client wants to view a video program *V1*, it sends a request to the proxy. If the proxy containing *V1* is stored in its storage, it can send the data to the client

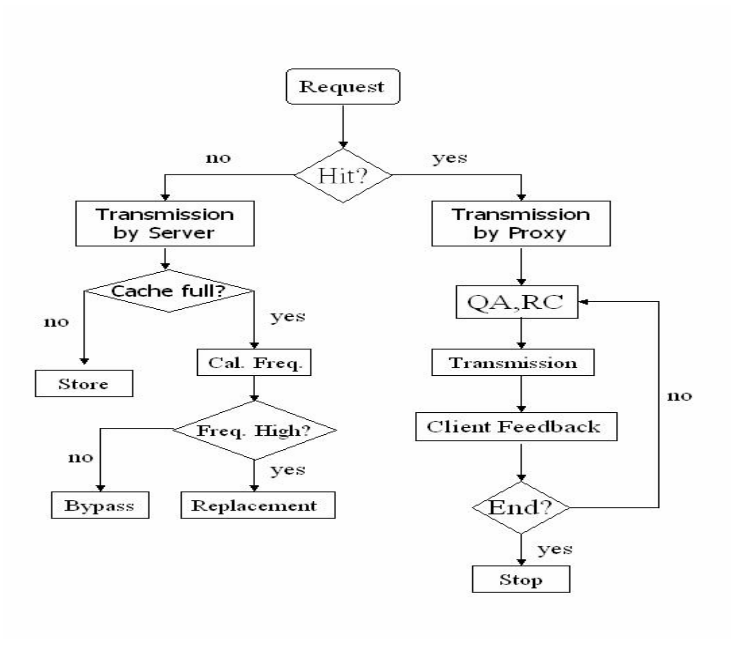

Figure 1. System flow.

directly. If the client continues to request the remaining data that is not stored in the proxy, the proxy can send the data to the client by "pre-fetch". If the prefix of video data is not stored in the storage, the proxy forwards the request to original video server. Upon redirecting the video data to the client, the proxy decides if it is worthy to evict some data of victims and allocate the space for *V1* if necessary.

3.2 Fibonacci Replacement Scheme

In this subsection we illustrate our *Fibonacci* replacement scheme in detail. It contains three steps. At first we explain why do we need a different allocation scheme and how to divide the video data into variable-sized segments. The second one shows how to calculate the popularity of each video title appropriately. The last step is to devise the replacement scheme according to above two principles.

Segment ID	1	2	3		4			5				
Block ID	1	2	3	4	5	6	7	8	9	10	11	12

Figure 2. Example of media segmentation.

3.2.1 Data Placement

In our data placement scheme we use variable-sized segmentation because it can reduce the number of segments and the time for replacement decision. On the other hand, we have explained that Yu's segmentation approach may loss the refinement. We choose Fibonacci function as the size distribution function to enhance the performance of Yu's method. The definition of Fibonacci function, Fib(), is shown as follows [11]:

$$Fib(i)=1 \quad i=1,2$$

$$Fib(i)=Fib(i-1)+Fib(i-2) \quad i>2$$

The function value of Fibonacci function of each index *i* grows as *i* grows, but is not as fast as that of Exponential Distribution used in Yu's approach. Figure 2 shows an example of segmenting the media data into Fibonacci series. While a video object will be stored, the segment with smaller ID is cached first. On the other hand, the last segment is evicted first when the video object becomes the victim of replacement. There are two advantages of this segmentation. First, this approach allows the cache to quickly discard a big trunk while performing replacement. Second, it retains the precision and refinement of calculating the popularity and replacement.

3.2.2 Popularity Function

Good estimation of popularity function makes the replacement scheme perform correctly and properly. Unlike traditional caching mechanism, we concern about not only whether the video object is requested or not at some time but also the duration of that request session. At first, we calculate the popularity according to the summation of duration of playback during a time interval. And then, we give higher qualities to the requests recently. Assume that we consider the last *n* requests. The popularity function in our scheme can be defined by formula (3-1).

$$P_{ij} = \sum_{k=Nj-(n-1)}^{Nj} \left[(playback\_time\_of\_k_{th}\_req.)*W_k \right] ....(3-1)$$

P_{ij} : *Popularity of stream i at time j*

N_j : *The number of requests at time j*

W_k: *The weighting value in the k_{th} request*

3.2.3 Replacement Scheme

After introducing two steps above, we start to illustrate the replacement method in detail. First, we define the caching utility of video *V* as the popularity of *V* divided by the amount of data stored in the cache. Figure 3 shows the cache update and replacement procedures formally. The unit of allocation and replacement is variable-sized segment. When the available size of storage is not enough, the proxy check that if there are enough *victims* with smaller caching utility. If yes, the proxy will perform cache replacement. Otherwise, the allocation of cache storage is not changed.

4. Exponential Rate Control Scheme and Preliminary Analysis

In this section, we introduce our rate control scheme during each data transmission session. In the first subsection, we divide our rate control algorithm into four steps and

```
Procedure UpdateCache
Input:
    Video : P
    Segment_ID : iSegID
    The free space of cache storage : iRem_Size
Program:
    Set iAmount equal to the size of the segment
    if(iAmount<=iRem_Size) store the video directly
    else {
        if (object P is referenced for the first time) return
        else ReplaceDecision(P,iSegId)
    }

Procedure ReplaceDecision
Input :
    Video : P
    Segment_ID : iSegID
    The free space of cache storage : iRem_Size
Program:
    Set iFree to be iRem_Size
    while(iFree<Seg_Table[iSegId]){
        find object Q with minimum caching utility
        if(the caching utility of P is larger than Q)
            put the last segment of Q into victim list and add free space
        else break
    }
    if(free space is enough)Perform actual replacement
    else  recover the replacement
```

Figure 3. UpdateCache and Replacement Decision procedures.

explain them clearly. In the second section, we give a preliminary analysis of our replacement scheme and compare with other schemes.

4.1 Exponential Rate Control Scheme

We have introduced that a better rate control mechanism can increase the bandwidth utilization and decrease the data loss rate. There are four steps in our scheme and we illustrate them in detail as follows.

4.1.1 Available Bandwidth Prediction

In order to avoid the fluctuation of transmission rate, we predict the maximum available bandwidth at first. Many existing tools can be used to perform this prediction and we choose *Iperf* among them to achieve this goal. Iperf [12] is a tool to measure the maximum TCP or UDP bandwidth between two hosts. It can report bandwidth, delay jitter, and datagram loss. Generally, we assume that the available bandwidth will not change rapidly in a short time interval. Thus, the prediction result can be used as reliable information. We show how to adjust our transmission scheme dynamically as follows.

4.1.2 Upper Bound Selection

After predicting the available bandwidth, the proxy can select the appropriate upper bound of transmission rate according to the predicting result. If the available bandwidth does not vary rapidly, the transmission rate will not go beyond this threshold and cause severe data loss. In our method, the upper bound is chosen to be 90% of the available bandwidth. Its advantage is that while the network condition has small variation, the transmission rate will not exceed the maximum available bandwidth soon.

4.1.3 Video Quality Adaptation

In this subsection, we introduce how to choose the lower bound of transmission rate. If the available bandwidth is smaller than this lower bound, the client will receive video programs with much lower quality, even the ejection of service. AIMD and receiver-based mechanism only transmit the whole quality version of video data. The video data cannot be transmitted smoothly as long as the available bandwidth is lower than the total bit rate requirement of video data. In our design, we adopt FGS because it provides the scalability of coding bit rate. The lower bound of transmission rate will depend on which layer the proxy wants to provide. We compare the available bandwidth B with all bandwidth requirements Ri for video layer i, $1 \leq I \leq m$, and find i such that $Ri < B < Ri+1$. Thus, i is the number of layer we want to transfer.

4.1.4 Design of Our Exponential Rate Control Mechanism

We have illustrated how to predict the available bandwidth and use the predicting result to select both upper bound and lower bound. In the following, we introduce our transmission scheme that uses *exponential-like* functions to control the transmission rate. We divide the transmission into ascending mode and descending mode. The transmission rate V in each mode is obtained using formulas (4-1) and (4-2).

Ascending mode: $V = (UB - LB)(1 - 2^{-(t-t0)}) + \mathrm{LB}$ ----(4-1)

Descending mode: $V = V_{t0}(1 - 0.1 * 2^{(t-t0)})$ ----(4-2)

UB: Upper Bound

LB: Lower Bound

t: The current time

$t0$: The initial point of the function

Notice that the short-term available bandwidth variation will affect our transmission rate, and long-term variation will affect not only the transmission rate but also video quality. The change of lower bound means that we adjust the quality of the transmitted data.

4.2 Preliminary Analysis of Replacement Decision

In this subsection, we give a preliminary analysis of our replacement scheme. We consider storage and time complexities in our analysis. Once a request comes, the proxy aligns the duration of the request into segments. The number of segments and the amount of data depend on the allocation scheme used in the proxy. Assume that it requests for video title *V1* with *N* blocks. Let *Pi* denotes the probability that the client views the video program from the beginning and stops viewing at the i_{th} block, $I = 1, 2, \dots N$. By Zipf law, we can express it as formula (4-3):

$$Pi=(i^k - (i-1)^k) / N^k \qquad \text{----(4-3)}$$

where *k* indicates the parameter of Zipf law. For example, if we specify that for each video, 60% of requests access 40% of video blocks from the beginning, then *k* equals to 0.6. As soon as we get the popularity of each block, we can calculate the average amount of requested blocks from the beginning for each request on the video by formula (4-4):

$$S = p1*1+p2*2+..+PN*N \qquad \text{----(4-4)}$$

Thus, we can find the minimum number of blocks that are more than *S* and are aligned into integral segments. Assume that the summation of blocks of the *n1* segments is larger than *S* in Yu's approach. We can use the summation of exponential series to obtain *n1* by formula (4-5) directly.

$$\begin{aligned}&(2^{n1}-1) > S\\ &n1 > \log_2(S+1) \qquad \text{----(4-5)}\end{aligned}$$

The smallest integer *n1* indicates the minimum number of segments that are needed to *cover* the S blocks. That is, *n1*=$\lceil \log_2(S+1) \rceil$. After obtaining *n1*, the amount of blocks that will be aligned is $2^{n1}-1$.

In fix-sized allocation like Lim's approach, the number of segments equals to $\lceil \frac{S}{10} \rceil$. Therefore, the amount of blocks that is aligned is $\lceil \frac{S}{10} \rceil * 10$.

Finally, in our Fibonacci allocation scheme, assume that there is *n2* segments to *cover S* blocks. We have

$$\sum_{i=1}^{n2} Fib(i) > S \qquad \text{---(4-6)}$$

$$\Rightarrow \quad \frac{(3+\sqrt{5})(\frac{1+\sqrt{5}}{2})^{n2} - (3-\sqrt{5})(\frac{1-\sqrt{5}}{2})^{n2}}{2\sqrt{5}} - 1 > S$$

$$\Rightarrow \quad n2 > \log((S+1) \times \frac{2\sqrt{5}}{3+\sqrt{5}}) / \log(\frac{1+\sqrt{5}}{2})$$

The smallest integer of *n2* is $\left\lceil \log((S+1) \times \frac{2\sqrt{5}}{3+\sqrt{5}}) / \log(\frac{1+\sqrt{5}}{2}) \right\rceil$. After obtaining *n2*, the amount of blocks is regarded as $\sum_{i=1}^{n2} Fib(i)$.

From the viewpoint of increasing storage utilization, the amount of blocks that the proxy aligns should be close to the actual amount of blocks that the client requests. Assume that the amount of blocks regarded as in Yu's method, our method, and Lim's method is *s1*, *s2*, and *s3*, respectively. We can derive that $s1 > s2 > s3 > S$ when $S > 20$.

As to analyze the time complexity of each method, we find the more the number of segments is, the more time the cache will take to perform allocation and replacement. From formulas above we can see that the number of segments divided for *N* blocks in these three

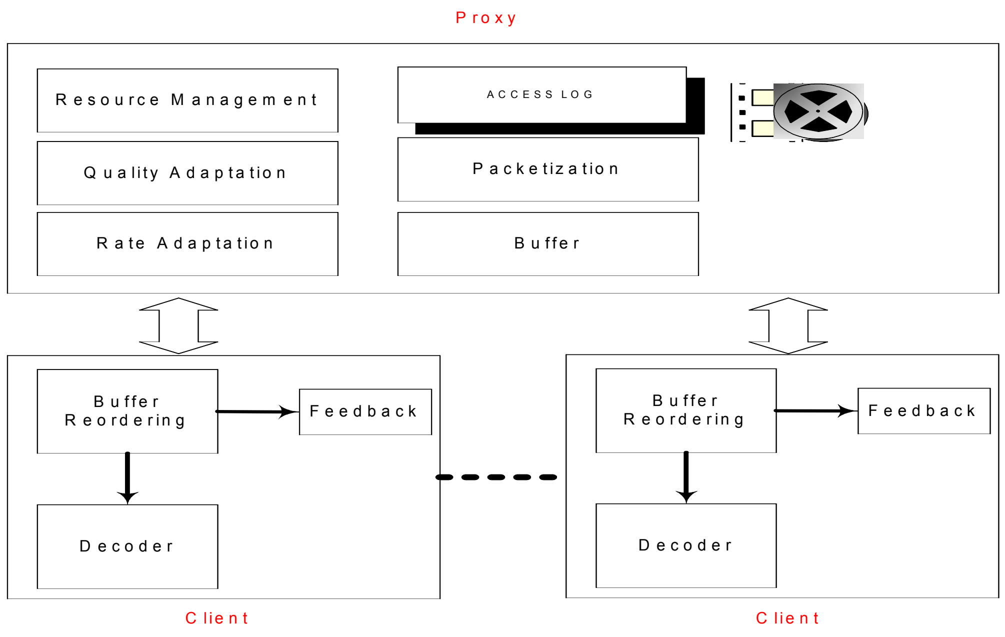

Figure 4. The structure of our simulator.

allocation schemes is *n1*,*n2*, and $\lceil \frac{S}{10} \rceil$, respectively. Thus, their time complexities are corresponded to O($\lceil \log_2(S+1) \rceil$), O($\lceil \log_2(S+1) \rceil$), and O($\lceil \frac{S}{10} \rceil$), respectively. Furthermore, we can derive that $n3 > n2 > n1$ when $S > 100$.

5. Simulation Environment and Performance Evaluations

In this section, we introduce our simulation environment and evaluating results of our schemes in some detail.

5.1 Simulation Environment

Figure 4 shows the overall structure of our simulation environment. Video objects in our simulation follows MPEG-4 standard. The sizes of video object and the cache storage are expressed in terms of number of media blocks. We assume that the popularity of each video title follows Zipf distribution. The simulation parameters are shown in Table 1. We set the

Table 1. Simulation parameters.

Simulation Parameter	Default Value
Zipf Distribution for Each Video Title	0.6
Total Cache Capacity	300
Video Size	200
Number of Video Titles	200
Zipf Parameter for Video Popularities	0.8

Zipf distribution parameter as 0.8, which means 80% of all requests reference 20% of video duration from the beginning for some video data.

5.2 Performance Evaluations

After illustrating the architecture of our simulation environment and parameters, we evaluate the results of our replacement and rate control schemes separately.

5.2.1 Evaluation of Replacement Scheme

In this subsection we compare our Fibonacci replacement scheme with methods proposed by Lim and Yu. When evaluating the performance of replacement scheme, we focus on the following metric: byte hit ratio, execution time, and the amount of data replaced.

We first compare results of these methods under different number of client requests. Figure 5, 6, and 7 contain their results. From Figure 5, we can find that when the number of requests increases, the execution time to perform replacement increases. Furthermore, when the number of requests is more than 8000, the execution time of our scheme lies between other two schemes. In average, the execution time of our method is faster than Lim's about 88%, and the execution time of Yu's method is faster than ours about 54%.

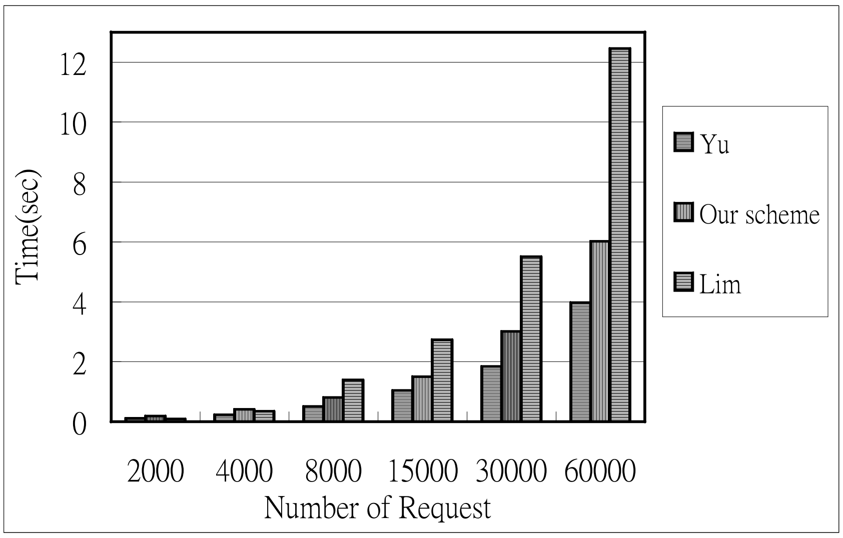

Figure 5. Number of requests vs. execution time.

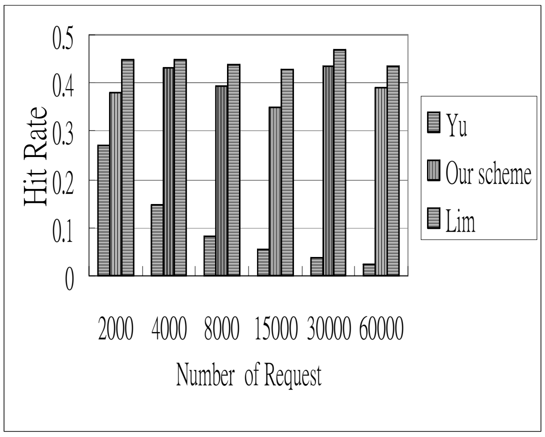

Figure 6. Number of requests vs. hit ratio.

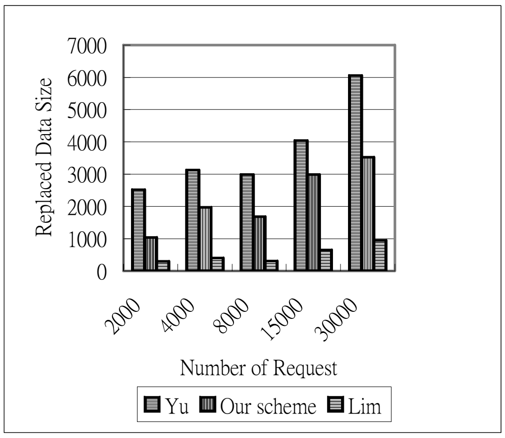

Figure 7. Number of requests vs. replace data size.

Figure 6 shows the byte-hit ratio of these methods by varying the number of requests. We can see that the fix-sized allocation scheme like Lim's provides higher precision when performing data replacement, so it can achieve higher byte-hit ratio. The byte-hit ratio of our method is a little lower than Lim's method and is much higher than Yu's method.

In the design of a video proxy system, it is important that the system cannot cause lots of data migration and replacement. Otherwise, it will increase the disk loading of the proxy

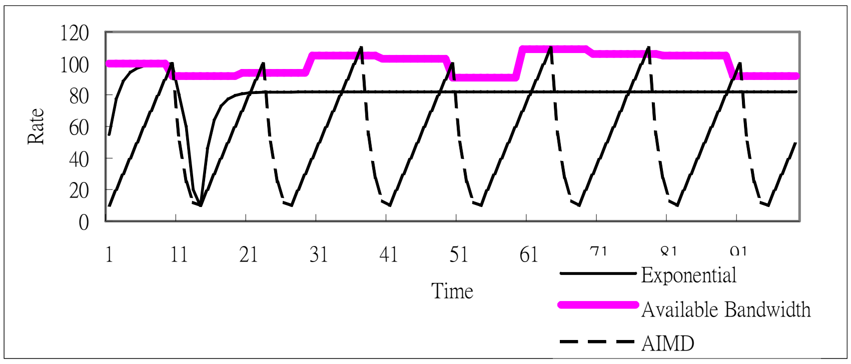

Figure 8. Bandwidth utilization comparison between Exponential Method and AIMD.

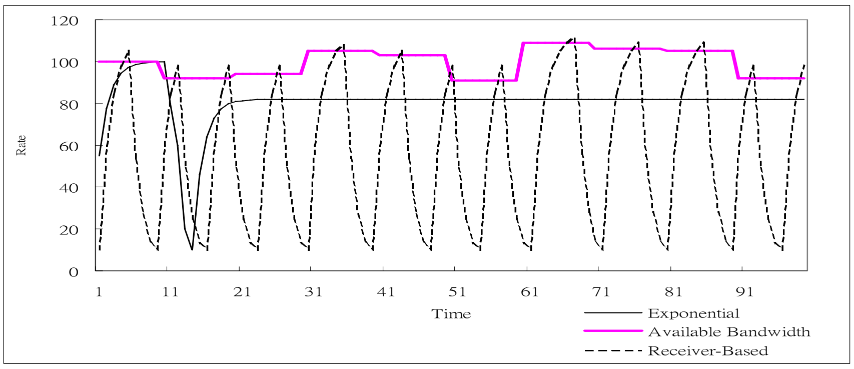

Figure 9. Bandwidth utilization comparison between Exponential and Receiver-Based Methods.

server. The amount of data replaced for different number of requests is shown in Figure 7. In Yu's method, the overestimation of request duration causes frequent data migration and increases the disk loading. Although this approach can reduce the replacement decision time, it causes large amount of data migration while the number of requests increases. Thus, it is not appropriate to be used for a large multimedia proxy system.

5.2.2 Evaluation of Rata Control Scheme

In this subsection we show the comparison of our exponential rate control scheme with other methods. We focus on evaluating bandwidth utilization and loss rate during transmission of some video layers. Figure 8 and 9 show the bandwidth usage compared with AIMD and receiver-based method when the variance of available bandwidth is 10%. We find that the transmission rate in our scheme increases rapidly and slows down when it approximates the upper bound we select. It can achieve high bandwidth utilization in the ascending mode. In the descending mode we uses an exponential-like function to control the decreasing of transmission rate. Therefore, the bandwidth utilization in the descending mode is still higher than AIMD. According to our simulation results, we outperform AIMD and receiver-based method about 85% and 43%, respectively. Besides, the loss rate of our method is much smaller than that of AIMD.

6. Conclusion and Future Work

In this paper, we use the Fibonacci function to our allocation scheme, and calculate the popularity function according to LFU and LRU principles. Our effective replacement scheme with Fibonacci data allocation scheme achieves higher byte-hit rate compared with conventional replacement mechanism. Besides, it can reduce the amount of data migration caused by cache replacement. During the actual data transmission in each session, we also propose the exponential transmission rate control mechanism to adjust the transmission rate. This mechanism achieves not only higher bandwidth utilization but also lower data loss rate.

In addition to previous features, there are still several promising issues in future work. First, if we increase the number of proxies, how to share the load to every video proxy server in balance under storage and timing constraints becomes an important issue. Second, we may integrate multicast technique to serve clients when they want to view the same video title.

Last, we assume that there is no limitation of the buffer size at the client side. Thus we can transmit the video data as much as possible. To extend our mechanism to more realistic solution, we should take the buffer size at the client side into consideration.

Reference

[1] W. Meira, Jr. Fonseca, E. Murta, and V. Almeida, “Analyzing Performance of Cache Server Hierarchies”, *Proc. of XVIII International Conference of the Chilean Society on Computer Science,* pp. 113 –121, 1998.

[2] Y. B. Lee, “Parallel Video Servers: A Tutorial”, *IEEE Multimedia*, No. 5, Issue 2, pp. 20 –28, April-June 1998.

[3] A. Das, Sun-Euy Kim and C. R., “Analyzing Cache Performance for Video Servers”, *Proc. of ICPP Workshops on Sivasubramaniam, Architectural and OS Support for Multimedia Applications/Flexible Communication Systems/Wireless Networks and Mobile Computing*, pp.38 –47, 1998.

[4] Y. M. Chiu and K. H. Yeung, “Partial Video Sequence Caching Scheme for VOD Systems with Heterogeneous Clients”, *Proc. of 13th International Conference on Data Engineering*, pp.323 –332, 1997.

[5] S. Sen, J. Rexford, and D. Towsley, “Proxy Prefix Caching for Multimedia Streams”, *Proc. of the Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, Vol. 3, pp.1310-1319, 1999.

[6] Eun-Ji Lim, Seong-Ho Park, Hyeon-Ok Hong, and Ki-Dong Chung, ”A Proxy Caching Scheme for Continuous Media Streams on the Internet”, *Proc. of the 15th International Conference on Information Networking*, pp. 720–725,2001.

[7] Kun-Lung Wu, Philip S. Yu, and Joel L. Wolf, “Segment-based Proxy Caching of Multimedia Streams”, *Proc. of the Tenth International World Wide Web Conference*, pp. 36-44, 2001.

[8] Y. R. Yang and S. S. Lam, “General AIMD Congestion Control“, *Proc. of International Conference on Network Protocols*, pp. 187–198, 2000.

[9] Yon Jun Chung, Young-Gook Kim, JongWon Kim, C.-C. Jay Kuo, “Receiver-Based Congestion Control Mechanism for Internet Video Transmission”, *IEEE International Symposium on Circuits and Systems*, Vol. 4, pp. 239-242, 1999.

[10] H. M. Radha, van der Schaar, M. Yingwei Chen, “The MPEG-4 Fine-grained Scalable Video Coding Method for Multimedia Streaming over IP”, *IEEE Transactions on Multimedia,* Vol. 3, Issue: 1, pp.53-68, March 2001.

[11] **Fibonacci Numbers, the Golden Csection and the Golden String**, http://www.mcs.surrey.ac.uk/Personal/R.Knott/Fibonacci/fib.html.

[12] **Iperf 1.6 - The TCP/UDP Bandwidth Measurement Tool**, http://dast.nlanr.net/Projects/Iperf/.