OSC: Optimal Selective Caching for Video Transmission with Proxy Servers

Ray-I Chang

Shin-Hung Chang[1]

Jan-Ming Ho

Institute of Information Science

Academia Sinica

Taipei, Taiwan, ROC

[1] *He is also with Dept of Computer Science and Information Engineering, National Taiwan University, Taiwan.*

Abstract – Proxy servers have been successfully used in reducing playback latency and network bandwidth required for delivering VBR videos. By storing portions of video programs in local video proxies and the rests in remote servers, the peak and variability of network bandwidth requirements can be reduced. Regarding to which portion of video frames should be cached at higher priority, let's consider an MPEG video in which an I-frame is referenced by all the other frames in the same GOP. Obviously, losing a packet belonging to an I-frame makes it difficult to recover all the frames of the same GOP. In this paper, based on the video proxy problem introduced by [1], we present the *Optimal Caching* (*OC*) algorithm to minimize cache size required for delivering a specific video program subject to network bandwidth and client buffer size constraints, and the *Optimal Selective Caching* (*OSC*) algorithm to further select the most appropriate portions of a video program for caching in the video proxy. In our experiments, it is shown that our algorithm is more effective than the previous algorithm. Our *OC* and *OSC* algorithms reduce cache size by 50% as compared with the algorithm presented in [1]. Furthermore, the ratio of I-frames cached in the video proxy is 5 times larger than the result of applying the algorithm presented by [1].

Index Terms – stored video transmission, proxy pre-cache server, optimal selective caching, minimal cache size, error control and recovery.

1. INTRODUCTION

Due to advances in broadband network and media compression technologies, it becomes a common practice to transmit various kinds of media such as video, audio, and traditional image/text over the Internet. Among these media, video streaming is the most challenging due to its large size and critical timing. Many applications including digital library, video on demand, distant learning, and various interactive video services are in urgent need of a video transmission scheme to achieve quality-guaranteed video presentation [1][2]. Internet is a compound of a spectrum of network technologies. There are high-speed backbone routers and WDM transmission lines at several hundred Gbps in its cores and high-speed last-mile access lines connecting the end customers, e.g., cable modem, wireless cable, and xDSL. But, there are also slow links, e.g., low bandwidth lease lines and traditional PC modems, connecting high-bandwidth sub-networks or local area networks. In addition, Internet traffic varies from time to time (see Fig. 1(a) and Fig. 1(b)) and becomes congested some time in a day though it is quite smooth, i.e., low network latency and low packet loss rate, etc., most of the time.

In the past, proxy servers were widely used to cache Internet data, prevalently, hypertext and image, in local stores. Thus, through the aide of a local proxy server that can be reached via high bandwidth connections, it reduces the probability for a client to directly retrieve external contents through congested or low-speed links or routers and thus to reduce average network latency, and to offload server and network traffic. Although the entire contents of data and image objects are cached for traditional internet proxies, however, the size of video programs make it impractical to cache an entire video program especially when we consider the rapidly growing number of user requests for interactive video programs. As introduced by [1], a scalable video caching solution stores a portion of each video program in local proxy servers and the rest in the originating remote servers. When a video program is requested for display, the client interacts with the local proxy server to retrieve the video program such

that the cached portion are transmitted from the local proxy and the rest from the remote video server (see Fig. 2). As the amount of data transmitted from the remote server is reduced, not only network bandwidth required but also probability of jittery playback occurred in the client can be reduced.

Wang et al. presented an algorithm, which will be referred to as CC algorithm in this paper, to decide which portion of a video program should be cached under limited network bandwidth and client buffer size [1]. However, they did not guarantee the minimization of cache size required in a video proxy server. We thus present an *Optimal Caching* (*OC*) algorithm to compute the minimal amount and portion of a video program to be pre-cached in the video proxy under given network bandwidth and client buffer size constraints. We also aim at fully utilizing other system resources in order to support as many client requests as possible. Time complexity of algorithm *OC* is $O(n)$, where n is the number of frames, which is the same as that of the previous result. We implemented both algorithms and studied their effects by using two different video clips. Experiments show that the optimal cached size obtained by our algorithm is 50% smaller than that obtained by the previous one [1]. Moreover, these experiments also show that, using algorithm *OC*, the utilization of network bandwidth allocated is closed to 100%. It is much better than that obtained by the previous algorithms. Our algorithm is shown effective and efficient.

Let's consider the popular video compression standards such as MPEG-1, MPEG-2, H.261, and H.263 etc. These technologies are based on the notion of motion compensation. Take MPEG video as an example, there are three different types of video frames, *i.e.*, I-frame, B-frame and P-frame as shown in Fig. 3. An MPEG video is divided into Group of Pictures, GOP for short. Each GOP contains an I-frame. The I-frames is compressed using DCT (discrete cosine transform) algorithm and is computed without referencing other video frames. A P-frame contains motion vector information obtained by encoding the difference between this P-frame and the I-frame belonging to the same GOP. A B-frame contains the encoding on how it could be reconstructed from adjacent I- and P-frames. Obviously, only if an entire I-frame is received, the P- and B-frames in the same GOP can be correctly decoded. To

minimize display errors, I-frames are given higher priorities than those P- and B-frames for storing in the local video proxy in order to minimize probability of transmission errors. However, in CC algorithm, different types of video frames are cached with equal priority. a large percentage of low-priority P- and B-frames may be stored into the proxy cache. In this paper, we present the *Optimal Selective Caching* (OSC) algorithm which not only computes the minimal cache size required but also maximizes the proportion of I-frame data in the minimum cache. Since the pre-cached I-frame data are not transported though congested or low-speed sub-networks, by maximizing its amount, the probability of display errors are also reduced. Experiments show that, given the same cache size, the percentage of I-frames cached by our OSC algorithm is 5 times larger than that cached by CC algorithm [1].

The remainder of this paper is organized as follows. In section 2, we present definitions of the optimal caching and the optimal selective caching problems. In Section 3, our algorithms, the optimal caching (OC) and the optimal selective caching (OSC) algorithms, are introduced. Experiment results that evaluate and compare our algorithms with CC algorithm are shown in Section 4. Finally, conclusions are given in Section 5.

2. PROBLEM DEFINITION

Due to the best-effort service model of the underlying Internet, transporting video streams across a wide area network is susceptible to network congestion due to inadequate network link capacity, high overhead and low performance of legacy network devices, or traffic variances. In [1], an algorithm was proposed to place video proxies near client devices such that part of a video program is directly accessible in a local area network where bandwidth is abundant. By storing part of a video program in a video proxy, one could allocate less bandwidth to transport the video stream between the remote video server and the proxy. By doing so, it not only increases system scalability, but also decreases the

probability of packet losses. The algorithm presented by [1] mainly focuses on hard limiting network bandwidth to locally decide the portions of video frames to cache in the proxy server. If a frame could not be transported using the specified bandwidth in one *frame-time*, the period between two contiguous frames are displayed, the excessive portion will be directly cut-off and stored in the proxy. This is referred to as the cut-off caching (CC) algorithm in this paper. A simple illustration of algorithm CC is shown in Fig. 4. As an Internet application, scalability is of primary concern. Thus, it is a rule of thumb to minimize the allocated resources such as network bandwidth, client buffer and proxy cache sizes. Specifically, to support as many users and remote video servers as possible, a good caching algorithm needs to minimize cache size subject to the given network bandwidth and client buffer size constraints. We also take into account the effect of losing a packet on display errors at the client. As explained earlier, in order to minimize display errors, the amount of I-frame data cached in the proxy must be maximized. In this section, we will capture these notions into formal definitions.

In the following, we start with the definition of the optimal caching problem. Then, an extension to consider diverse impacts of I-frames over video playback is introduced (called the optimal selective caching problem). Before formulizing these problems, definitions of parameters used in this paper are listed below.

Definition-1: Problem Parameters

1. $V = \{ f(i) \mid 0 \le i \le n-1 \}$ represents a video program where $f(i)$ represents the size of the i-th frame and n is the number of frames.
2. R (bpf, bit-per-frame-time) is the allocated constant bandwidth.
3. L (number-of-frame-time) is the startup latency. It is the period starting from the receiving of the video program to the beginning of playback at the client side.

4. $b(i)$ represents data occupancy in the client buffer at time i (for $0 \le i \le n-1$). To guarantee the jitter-free playback quality, it must satisfies $f(i) \le b(i) \le B$, where B is the capacity of client buffer.

Without loss of generality, $f(i)$ is assumed to be displayed at time i. A portion of $f(i)$ with size q is represented by $f(i)[q]$ as shown in Fig. 5(a). The portion of video frame $f(i)$ cached in proxy server is denoted by $f(i)[c(i)]$ (see Fig. 5(b)). The total size C of video cached in the proxy is $C = \sum_{i=0}^{n-1} c(i)$.

Based on the parameters described above, the optimal caching problem can be formulized as follows.

Definition-2: Optimal Caching Problem

Given a video stream V, startup latency L, client buffer size B and a network bandwidth R, we want to compute the portion $f(i)[c(i)]$, for $0 \le i \le n-1$, of video data to be cached in the proxy server to minimize cache size required $C = \sum_{i=0}^{n-1} c(i)$ for supporting jitter-free video playback.

As previously discussed in section 1, major video compression schemes, e.g., MPEG and H.26x, are based on the notion of motion compensation. Take MPEG as an example, a video program is divided into GOPs with each GOP containing an I-frame. Each video frames in a GOP then stores encoding of picture information with reference to the I-frame. So, each frame in a GOP will not be decoded correctly unless the I-frame is correctly received and decoded. We also observed that probability of transmission error is higher between the video server and the proxy than it is between the proxy and the client. Thus, by storing as much I-frame data in the proxy would decrease the probability of video decoding error and thus increase display quality. Based on these notions, the optimal selective caching problem is defined as follows.

Definition-3: Optimal Selective Caching Problem

Given a video stream V, a startup latency L, a client buffer B and a limited bandwidth R, we want to decide the portion of data content cached $f(i)[c(i)]$ to minimize the cache size required C and then maximize the percentage, (C_I/C), C_I is I-frames cached size in proxy cache, for supporting jitter-free video playback.

The total amount C_I of I-frame data cached in proxy server can be computed as $C_I = \sum_{i=0}^{n-1} u(i) \times c(i) \le C$, where $u(i) = 1$ if the frame $f(i)$ is an I-frame; otherwise, $u(i) = 0$.

3. Optimal Selective Caching Algorithm

In this section, we first present the optimal caching (OC) algorithm that considers different data frames as having the same priority. Given a video stream and pre-specified system parameters, i.e., startup latency, client buffer size, and network bandwidth, our algorithm minimizes the cache size required to support jitter-free video playback at the receiving client. The computation complexity of algorithm OC is $O(n)$, where n is the number of frames, which is as low as that of algorithm CC presented by Wang et.al. Then, based on the OC algorithm, the optimal selective caching (OSC) algorithm is introduced to maximum amount of the most I-frame data for caching in the proxy.

3.1 Optimal Caching Algorithm

At any time i, the client consumes a video frame $f(i)$ for jitter-free display and receives at most R (bits) new data from the remote video server. Without the occurrence of buffer overflow, the quantity of buffer occupancy $b(i)$ is defined by the recurrent relation $b(i) = b(i-1) + R - f(i-1)$ as shown in Fig. 6. The initial value is $b(-L) = 0$, where L is the given startup latency. To avoid buffer overflow, the value of $b(i)$ is modified into $b(i) = \mathbf{min}\{B, b(i)\}$, where B is the buffer size. As finite network bandwidth R is allocated, the aggregation of data pre-stored in client buffer and those received from the remote

server may be less than a complete frame. When such a situation occurs, the client needs to retrieve the rest of the frame from the proxy server to guarantee jitter-free playback.. The data content pre-cached in the proxy server is $f(i)$[c(i)] where $c(i) = |b(i) - f(i)|$ as shown in Fig. 5(b). Therefore, although allocated network bandwidth is insufficient, with the help of the proxy server, the client buffer always maintains sufficient video data to satisfy jitter-free playback. A detail description of algorithm OC is shown as follows.

ALGORITHM-1: *OC (optimal caching)*

1. $i = -L$; $b(i) = 0$; /* i is the current time point with startup latency L. $b(i)$ is the buffer occupancy */
2. **repeat**
3. $i = i + 1$; $b(i) = \min\{B, b(i-1) + R - f(i-1)\}$;
4. **if** ($b(i) \geq f(i)$) **then** $c(i) = 0$; /* QoS constraints */
5. **else** { /* -- buffer is underflow */
6. $c(i) = |b(i) - f(i)|$; $b(i) = f(i)$;
7. cache $f(i)$[c(i)] in the proxy;
8. }
9. **until** ($i > (n-1)$);

In algorithm OC, the proxy and the remote video server aggressively pump video data into the client buffer. A portion of video program is cached into the proxy server if and only if it can't be timely transmitted by using the network bandwidth allocated. The cache size allocated by our proposed algorithm can be proved to be minimal for the given startup latency, client buffer and network bandwidth.

Theorem-1: The cache size allocated by our OC algorithm is minimized for the given startup latency, client buffer and network bandwidth.

Proof:

Given the startup latency L, client buffer B and WAN bandwidth R, we can sequentially compute the buffer occupancy $b(i) = \min\{B, b(i-1)_{\Delta} + R\Delta - f(i-1)\}$ to decide the frame data $f(i)[c(i)]$ cached in the proxy for $i = 0$ to $n-1$. We can prove by contradiction that, at any time point k, the cumulated size of video data cached $\sum_{i=0}^{k} c(i)$ is minimized for the given startup latency L, client buffer B and WAN bandwidth R. Therefore, the total cache size required $C = \sum_{i=0}^{n-1} c(i)$ is minimal.

3.2 Optimal Selective Caching Algorithm

In algorithm OC, we directly cache the portion of frame data to avoid subsequent buffer underflow. As different video frames are dealt with the same priority, algorithm OC, as well as algorithm CC, may arbitrarily store a large number of low-priority P- and B-frames into the proxy cache. In this sub-section, we modify algorithm OC to solve the optimal selective caching (OSC) problem. We present algorithm OSC to select into the proxy cache the maximum amount of I-frame data subject to the minimum cache size computed by algorithm OC. As shown in Fig. 7, based on the cache video data computed by algorithm OC, in algorithm OSC, we try to exchange the original cached data $f(i)[x]$ with a previous I-frame. Whenever the portion $f(j)[x]$ of I-frame is selected for caching at the time point j, x amount of subsequent video data are also pre-fetched to the client buffer. These x amount of extra client buffer space are occupied from time j to i. We also note that, the time point j, the buffer space available for pre-fetching extra data is $\bigtriangledown[j] = \min\{B - b(k) \mid \text{for } k = j \text{ to } i\}$ as shown in Fig. 8. x is thus defined as $x = \min\{f(j), \bigtriangledown[j], c(i)\}$ where $f(j)$ is the available frame size for caching, $\bigtriangledown[j]$ is the available buffer space for pre-fetching extra data, and $c(i)$ is the remaining size of the original cached

data. The same idea can be iteratively applied to other I-frames until either $\nabla[j]=0$ or $c(i)=0$. As algorithm OSC is similar to the algorithm OC in computing the amount of cached video data, we can directly rewrite the step 7 of the OC algorithm as follows.

7.1. $j=i$; $\nabla[j]=B-b(j)$;

7.2. **repeat** /* select the most I-frames */

7.3. j--; $\nabla[j]=$ **min**{$\nabla[j+1]$, $B-b(j)$};

7.4. **if** ($f(j)$ is an I-frame) **then** {

7.5. $x=$ **min**{$f(j)$, $\nabla[j]$, $c(i)$}; /* available size for caching */

7.6 $c(j)=c(j)+x$; $c(i)=c(i)-x$; $\nabla[j]=\nabla[j]-x$;

7.7 *cache f(j)[x] in the proxy*;

}

7.8. **until** ($\nabla[j]=0$);

7.9 cache $f(i)$[c(i)] in the proxy;

Theorem-2: The OSC algorithm can decide the minimal cache size required and pre-stores the most I-frames into this proxy cache for the given the startup latency, client buffer and network bandwidth.

Proof:

As the OSC scheme is similar to the OC scheme in deciding the data size cached, the cache size allocated is minimized as proved in Theorem-1. At any time point i, the minimal size of frame data cached $c(i)$ can be decided. By sequentially tracking each I-frame, the largest data size of I-frames that can be pre-cached can be calculated for the given startup latency, client buffer and WAN bandwidth. By contradiction, we can prove that at any time point i the most I-frames before frame $f(i)$ are cached by the OSC scheme.

3.3 A Fast OSC Algorithm

The above algorithm has time complexity $O(n^2)$. For reducing the time complexity to linear, we use a linked list structure, say *I-list*, to keep the location of all appeared I-frames in video program (shown as Fig. 9) so the I-frame most closed to j, say w, could be easily selected. Furthermore, the un-cached size $\Delta(w)$ and the available buffer space $U_w = \mathbf{min}\{B-b(k) \mid w < k < w + (\text{I-distance})\}$ for pre-fetching of I-frame w are also kept in the w node of I-list. In addition, we also use $\bigtriangledown U$ to keep $\bigtriangledown[j]$ from the frame w to frame j instead of back tracking frequently from each time point j. The initialized value of $\bigtriangledown U$ is set to a large value. Therefore, the real size of frame data that can be cached is $x = \mathbf{min}\{\Delta(w), \bigtriangledown U, c(i)\}$ where $\Delta(w)$ is the available frame size which I-frame w has for caching, $\bigtriangledown U$ is the available buffer space for pre-fetching, and $c(i)$ is the remained size of the original cached data. The same idea can be sequentially applied to other I-frames until there is no available buffer space for pre-fetching the original cached data. As this fast OSC algorithm is similar to the OC algorithm in deciding the data size cached, we can directly rewrite the step 3 and the step 7 of the OC algorithm to select the most I-frames for caching as follows.

Rewriting step 3 of OC algorithm as following:

3.1. $i = i + 1;$

3.2. $b(i) = \mathbf{min}\{B, b(i-1) + R - f(i-1)\};$

3.3. $\bigtriangledown[i] = B - b(i);$

3.4. **if** $(\bigtriangledown[i] < \bigtriangledown U)$ **then** $\bigtriangledown U = \bigtriangledown[i];$

3.5. **if** ($f(i)$ is an I-frame) **then**

3.6. {

3.7. $\Delta(i) = f(i); U_i = \bigtriangledown[i]; U_{i-(\text{I-distance})} = \bigtriangledown U;$

3.8.　　Add the un-cached I-frame *f(i)* to *I-list;*

3.9.　}

Rewriting step 7 of OC algorithm as following:

7.1　**repeat**　/*select the most I-frame*/

7.2　　Get the next un-cached I-frame *f(w)* from *I-list;*

7.3　　$\nabla U = \min\{\nabla U, U_w\}$;

7.4　　$x = \min\{\Delta(w), \nabla U, c(i)\}$;　/*available size for caching*/

7.5　　$c(w) = c(w) + x$; $\Delta(w) = \Delta(w) - x$;

7.6　　$c(i) = c(i) - x$;

7.7　　$\nabla U = \nabla U - x$;

7.8　　cache *f(w)*[*x*] in the proxy;

7.9　**until** (*c(i)*=0||∇U=0||*I-list*=NULL);

7.10　cache *f(i)*[c*(i)*] in the proxy;

7.11　**if** ($\Delta(w)$>0) **then**

7.12　{

7.13　　$U_w = \nabla U$;

7.14　　Add the un-cached I-frame *f(w)* to *I-list;*

7.15　}

When buffer underflow occurs at the time point *i*, the OC algorithm directly caches the portion of frame *f(i)*[*c(i)*] into the proxy server. In the OSC algorithm, we back track to seek the I-frames that can be cached. As the I-frames are cached aggressively into the proxy server, we can prove that our OSC

algorithm can cache the most I-frames into the proxy server for the given startup latency, client buffer, network bandwidth and the minimal cache size obtained.

3.4 Comparing with the Previous Algorithm by a Simple Example

To illustrate why our proposed algorithm can achieve a smaller cache requirement than the previous CC algorithm, a simple example with two video frames $f(0)$ and $f(1)$ is provided. As shown in Fig. 10, while the network bandwidth R is larger than $f(0)$, the data size cached at the time point 0 is $c'(0) = c(0) = 0$. In Addition, there is $(R - f(0))$ bandwidth capacity available for transmitting other data. The previous CC algorithm selected to idle and wasted this bandwidth capacity. On the contrary, our algorithm fully uses this bandwidth capacity to pre-fetch the next frame data $f(1)[R - f(0)]$. Thus, at the next time point, the media data required for transmitting is smaller than the original one. Assume that the buffer is underflow for transmitting the next frame $f(1)$ and some video data need to cache in the proxy server. Our algorithm caches only $f(1)[c'(1)]$ where $c'(1) = f(0) + f(1) - 2*R$ into the proxy server through the pre-fetching of $f(1)[R - f(0)]$ at the time point 0. It is better than the previous CC algorithm that requires the cache size $c(1) = f(1) - R > c'(1)$.

4. EXPERIMENTAL RESULTS

In the previous section, we theoretically compare our algorithm with the algorithm CC by their processing steps. Our algorithm is shown to optimize the cache size required for a video proxy under given startup latency, client buffer and network bandwidth. We also implement these algorithms including algorithm CC and run on several benchmark video streams. The algorithms are compared based on the following performance indices.

1. Percentage of data cached:

(cache size)/(video size)*100%.

2. Percentage of I-frames cached:

(I-frames cached)/(cache size)*100%.

3. Percentage of network bandwidth utilization:

(bandwidth utilized)/(bandwidth allocated)*100%.

The same startup latency, client buffer size, and network bandwidth are used for each algorithm in running the experiments. They show that our algorithms are more effective in minimizing the cache size required, maximizing the percentage of I-frames cached, and maximizing the utilization of network bandwidth allocated.

4.1 Percentage of Data Cached

To reduce the cost and improve the system scalability in building proxy servers, the cache space allocated for serving each video request must be precisely controlled. Under the same startup latency, client buffer and network bandwidth, a good proxy caching algorithm should store as less video data into proxy servers as possible. It minimizes the percentage of data cached for the given video stream. In Fig. 11, the percentages of data cached by different algorithms are presented. Experiments show that both the OC and OSC algorithms proposed require the same optimal cache size. The percentage of data cached by our proposed algorithms is better than that by the previous CC algorithm. When the transmission rate applied is increasing, the percentages of data cached by different algorithms are decreasing. However, the decreasing slope obtained by our proposed algorithms is sharper then that obtained by the CC algorithm. When the transmission rate is increased from 200 Kbps to 400 Kbps, our improvement in the percentages of data cached has increased 4 times. Without loss of generality, we can let the transmission bandwidth R be the mean rate of the video stream $\sum_{i=0}^{n-1} f(i)/n$. The cache size

achieved by our algorithms is averagely 50% smaller than that required by the previous CC algorithm [1]. The improvement is significant.

4.2 Percentage of I-frames Cached

I-frames play the most important role in video decoding. The more complete the I-frames are received, the more smoothness the playback quality can be guaranteed. As the network channel communicated by proxy servers are more robust than that by remote servers, we want to cache the most I-frames in proxy servers to take the best advantage on error control and recovery. Given the same cache size in the proxy server, the percentage of I-frames cached can be applied to measure the smoothness of video playback. In Fig. 12, we show the percentages of I-frames cached by different algorithms. Experiments show that our proposed OSC algorithm can minimize the cache size required and store the most I-frames in the proxy server at the same time. In the previous CC algorithm, as different frame data are cached with the same priority, the percentages of I-frames cached for different transmission rates are almost the same. On the contrary, our proposed OSC algorithm can select the most I-frames for caching. When the transmission rate applied is linearly increasing, the percentage of I-frames cached is exponentially increasing. Averagely, our approach is 5 times better than the CC algorithm in the percentage of I-frames cached.

4.3 Utilization of Network Bandwidth Allocated

As the available network bandwidth is limited, we must utilize it sufficiently and avoid wasting it at any time. In a distributed multimedia system, high bandwidth utilization implies that lots of video requests can be served at the same time. In Fig. 13, we show the utilization of network bandwidth achieved by different algorithms. As our proposed algorithms utilize the network bandwidth allocated by an aggressive pre-fetching algorithm, the bandwidth utilization obtained is closed to 100%. On the

contrary, the bandwidth utilization obtained by the previous CC algorithm is decreasing when the transmission rate allocated is increasing. The network bandwidth allocated is wasted.

5. CONCLUSIONS

In this paper, the problem of video cache design is studied. We present algorithm OC to minimize the size of proxy cache required for supporting jitter-free video playback subject to bandwidth and buffer size constraints. The time complexity of algorithm OC is same as that of the CC algorithm. We also present the *Optimal Selective Caching* (*OSC*) algorithm to pre-cache the maximum amount of I-frame data into the proxy server based on the computation result of algorithm OC. Experiments show that the cache size required by OSC algorithm is in average 50% smaller than that by algorithm CC. Besides, the percentage of I-frame cached is 5 times that of the CC algorithm. We are also studying the problem of optimization of other system resources, e.g., minimization of client buffer size, minimization of network bandwidth, etc.

6. REFERENCES

[1] Yuewei Wang, Zhi-Li Zhang, David H.C. Du, and Dongli Su, "A Network-Conscious Approach to End-to-End Video Delivery over Wide Area Networks Using Proxy Servers," in proc. of IEEE INFOCOM 1998.

[2] Wu-Chi Feng and Jennifer Rexford, "A Comparison of Bandwidth Smoothing Techniques for the Transmission of Prerecorded Compressed Video," in proc. of IEEE INFOCOM 1997.

[3] Danzig, P. B., Hall, R. S., and Schwartz, M. F., "A case for caching file objects inside inter-networks," in proc. of ACM SIGCOMM. 1993.

[4] Feng, W., and Sechrest, "Critical bandwidth allocation for delivery of compressed video," in proc. of Computer Communication (Special issue on Systems Support for Multimedia), 1995.

[5] Lam, S. S., Chow, S., and Yau, "An algorithm for loss-less smoothing of MPEG video," in proc. of ACM SIGCOMM, 1994.

[6] Dan, A., Sitaram, and shahabuddin, "Scheduling Polices for an On-demand Video Server with Batching," in proc. of ACM Multimedia, 1994.

[7] ftp://thumper.bellcore.com/pub/vbr.video.trace

[8] ftp://ftp-info3.informatik.uni wuerzburg.de/pub/MPEG/traces/

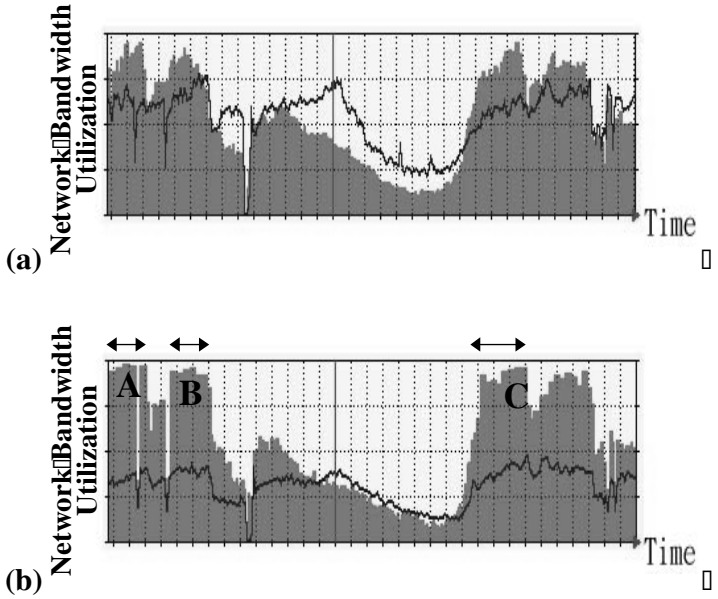


Fig. 1. (a) Internet traffic saturation never occurs in a day. (b) Internet traffic becomes congested (as shown in duration A, B, and C) in a day.

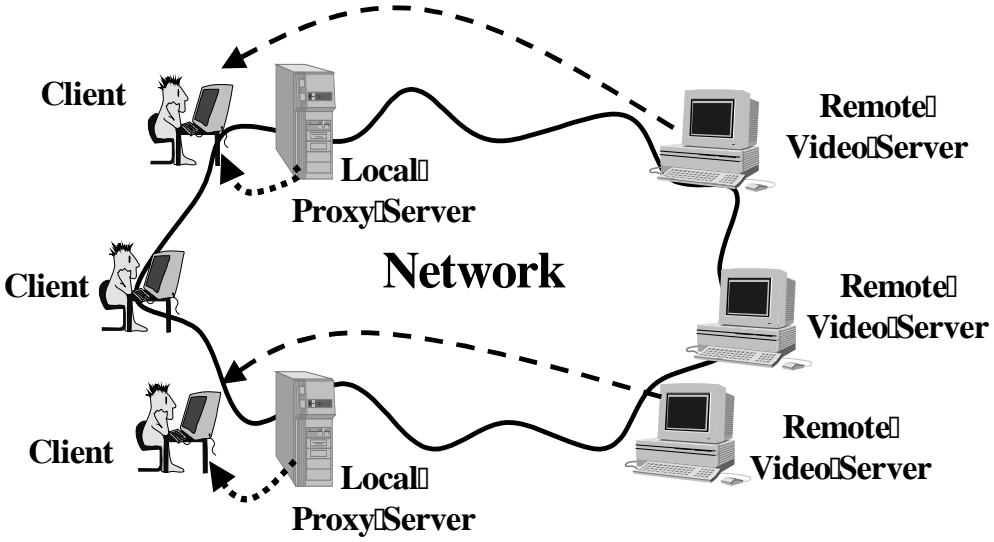


Fig. 2. A video transmission system with proxy servers between users and video servers.

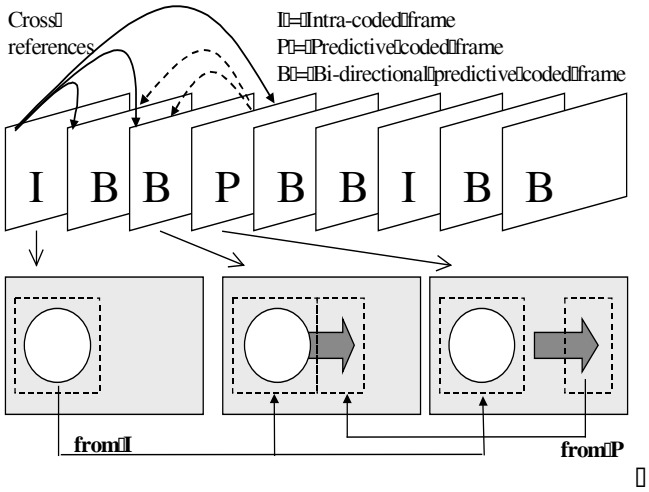


Fig. 3. A demonstration of "Motion compensation" and "Crosses reference" in a compressed video program.

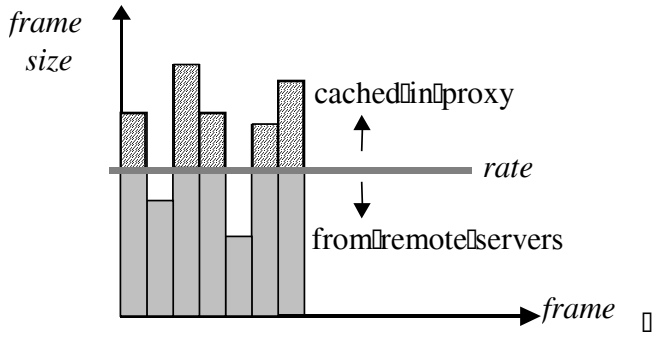


Fig. 4. The CC algorithm uses a heuristic rule to decide the cache size by the network transmission rate (network bandwidth) allocated.

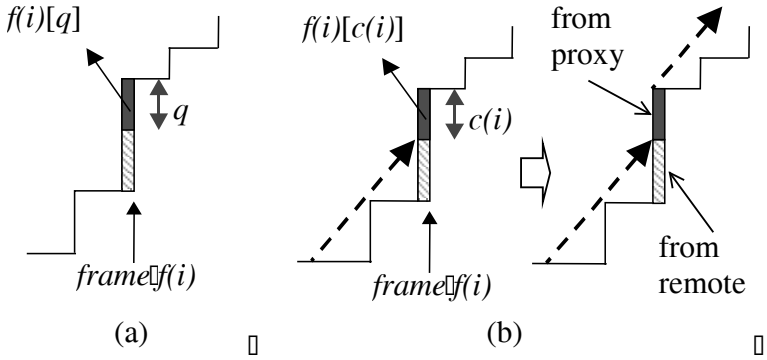


Fig. 5. (a) A portion of frame *f(i)* with size *q* can be represented by *f(i)*[*q*]. (b) In this paper, the portion of data content cached in proxy is *f(i)*[*c*(*i*)].

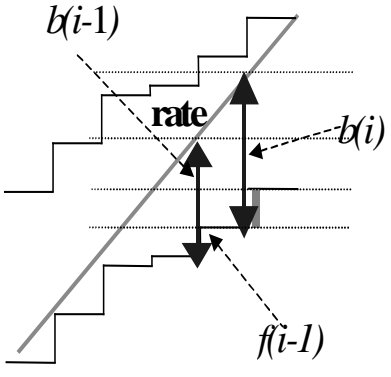


Fig. 6. The buffer occupancy can be computed by

***b(i)* = *b(i-1)* + *R* - *f(i-1)*.**

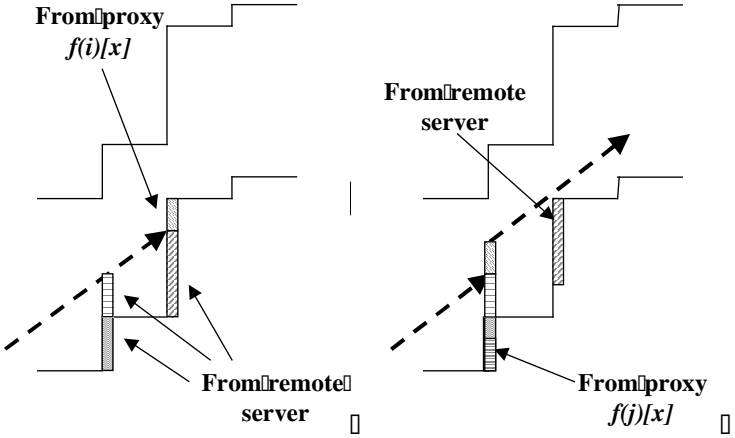


Fig. 7. Our selective caching algorithm tries to exchange the original cached data *f(i)*[*x*] with the I-frame data *f(j)*[*x*] pre-fetched in the client buffer.

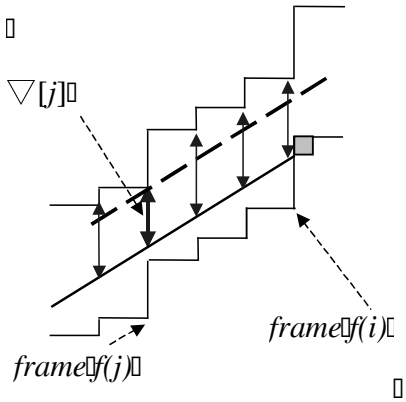


Fig. 8. Assume that the current time point is *i*. ▽[*j*] = min{ *B-b(k)* | for *k* = *j* to *i* } records the available buffer space at the time point *j* for pre-fetching the original cached data.

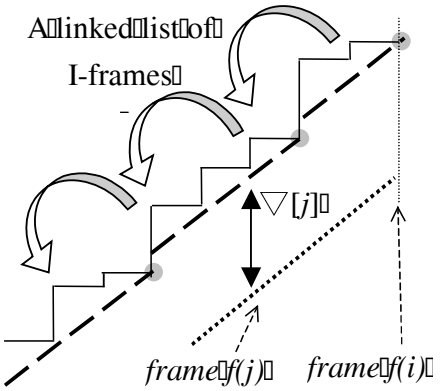


Fig. 9. Linked list is used to record the location of each I-frame.

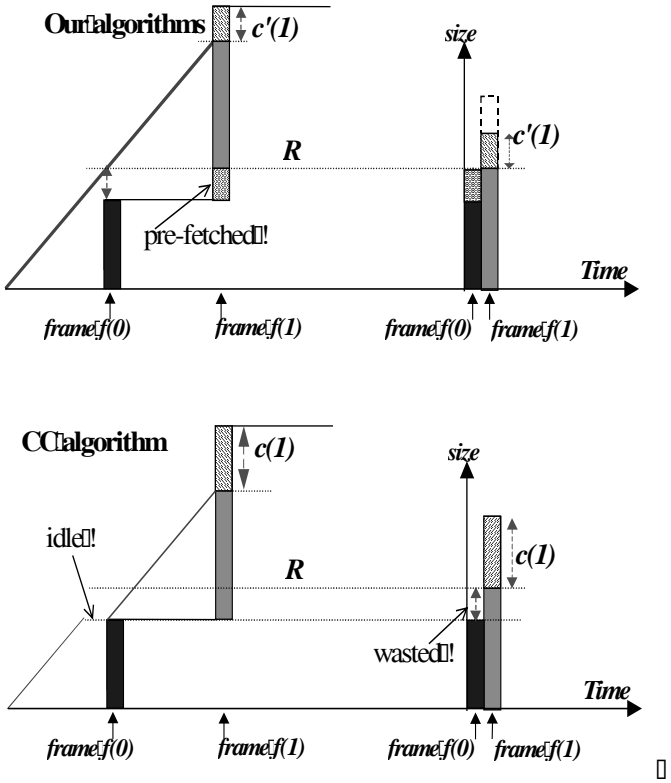
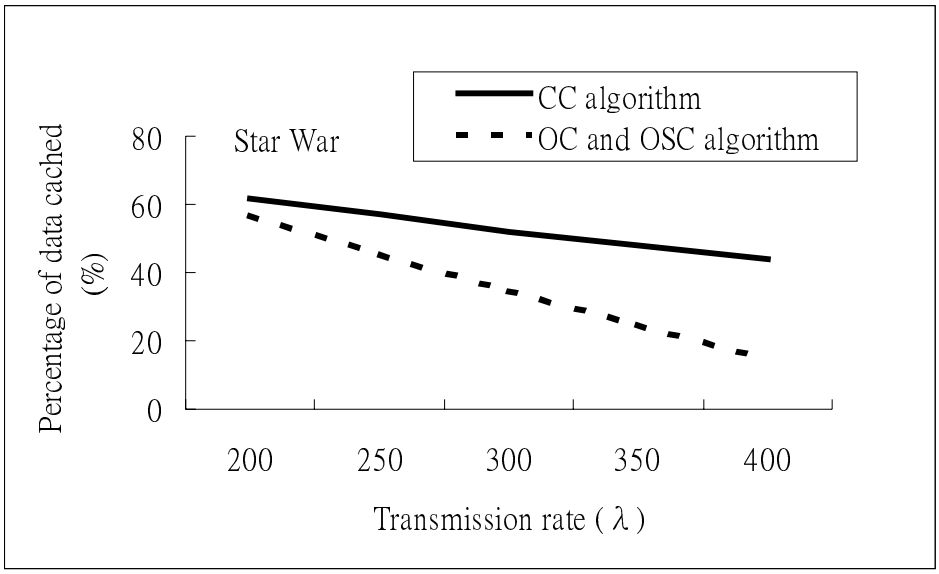
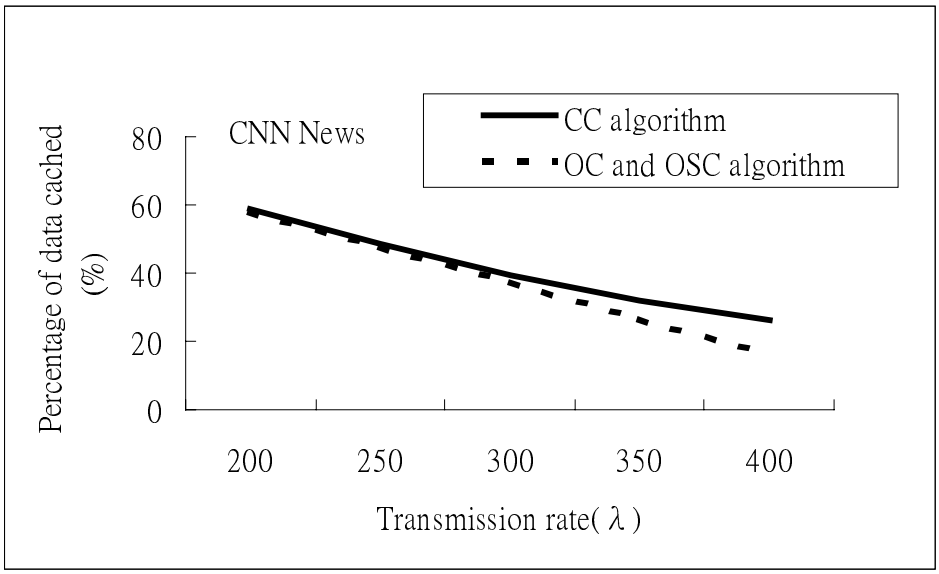


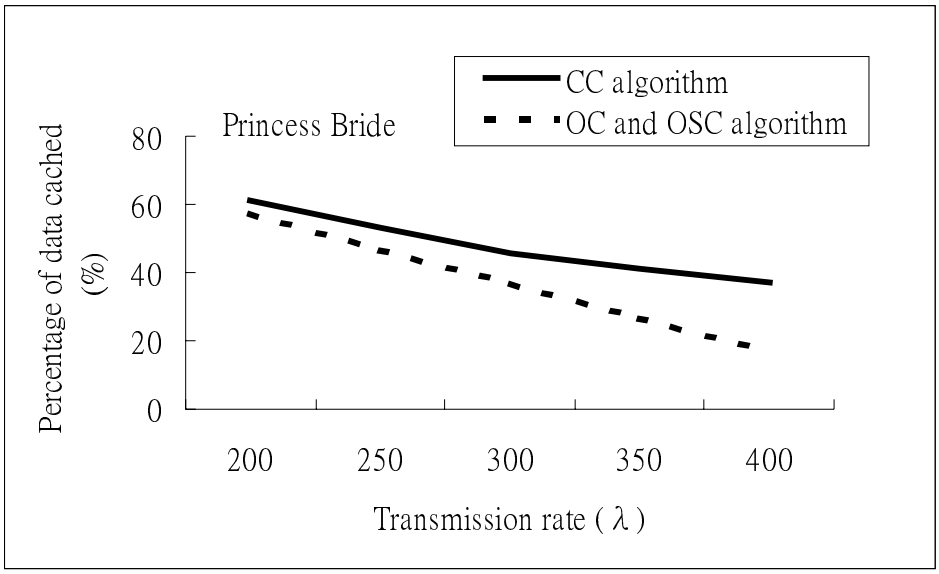
Fig. 10. Compare the processing steps of our algorithms and the CC algorithm by a simple example with two media frames.


Star War
CC algorithm
OC and OSC algorithm
Percentage of data cached (%)
80
60
40
20
0
200
250
300
350
400
Transmission rate (λ)

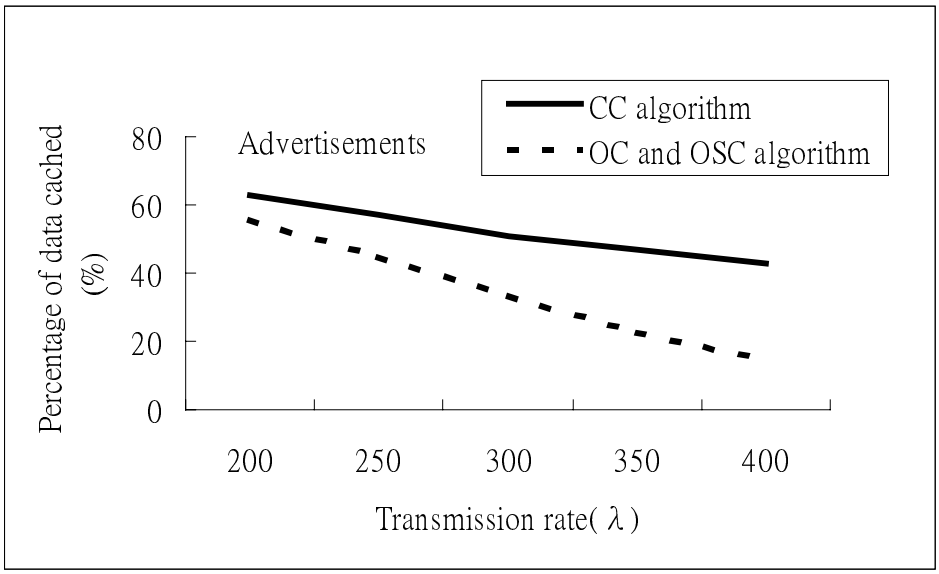
(a)


CNN News
CC algorithm
OC and OSC algorithm
Percentage of data cached (%)
80
60
40
20
0
200
250
300
350
400
Transmission rate(λ)

(b)

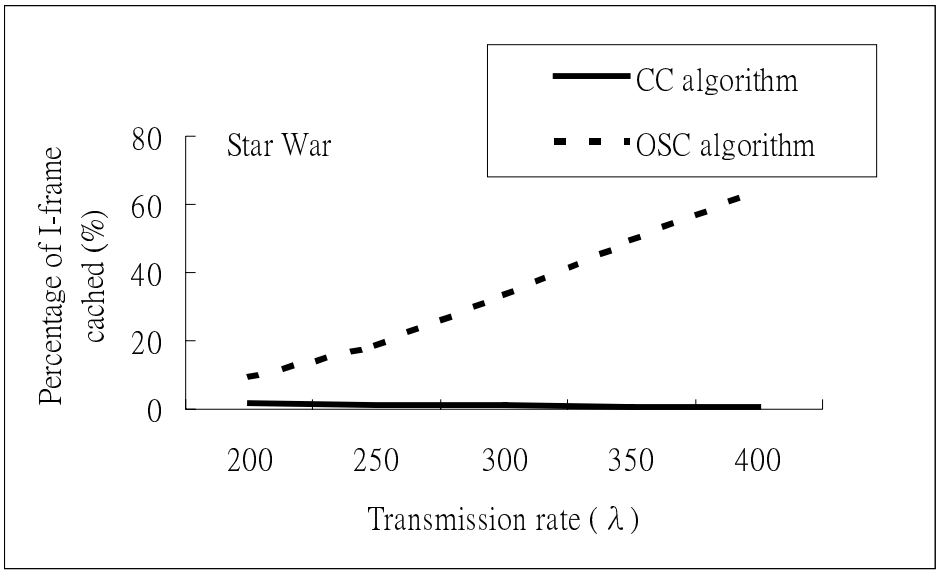


(c)

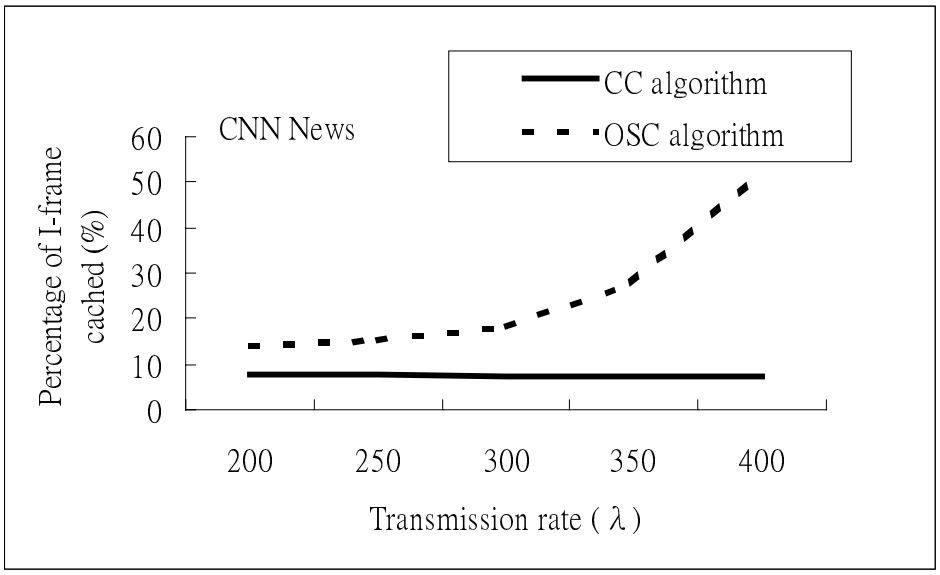


(d)

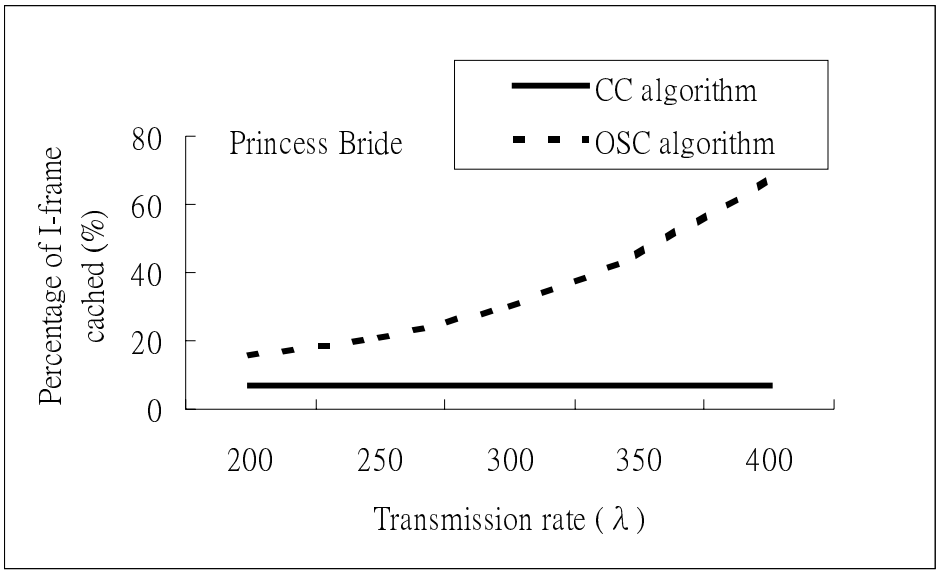
Fig. 11. The percentage of data cached in proxy cache servers for different algorithms. (a) Star War. (b) CNN News. (c) Princess Bride. (d) Advertisements.


CC algorithm
OSC algorithm
Star War
Percentage of I-frame cached (%)
80
60
40
20
0
200
250
300
350
400
Transmission rate (λ)

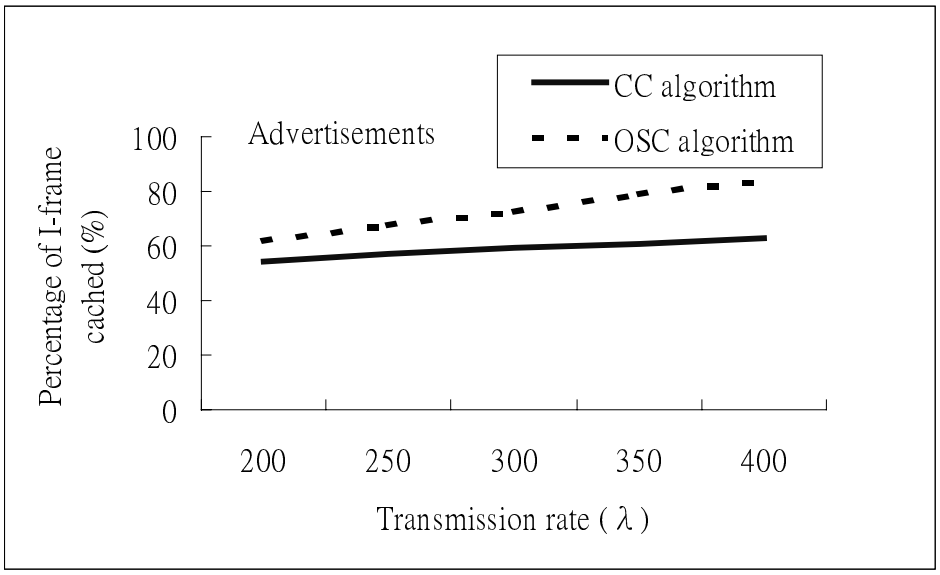
(a)


CC algorithm
OSC algorithm
CNN News
Percentage of I-frame cached (%)
60
50
40
30
20
10
0
200
250
300
350
400
Transmission rate (λ)

(b)

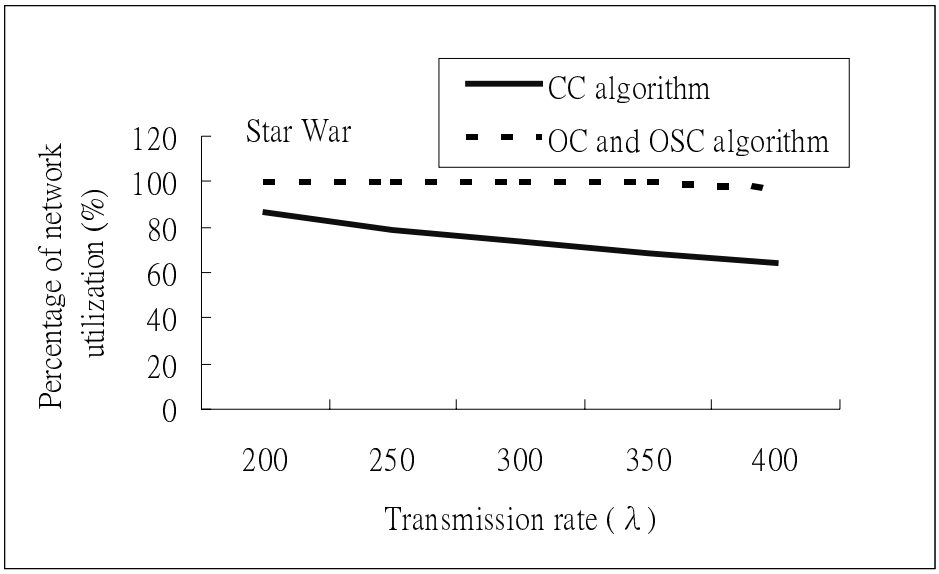


(c)

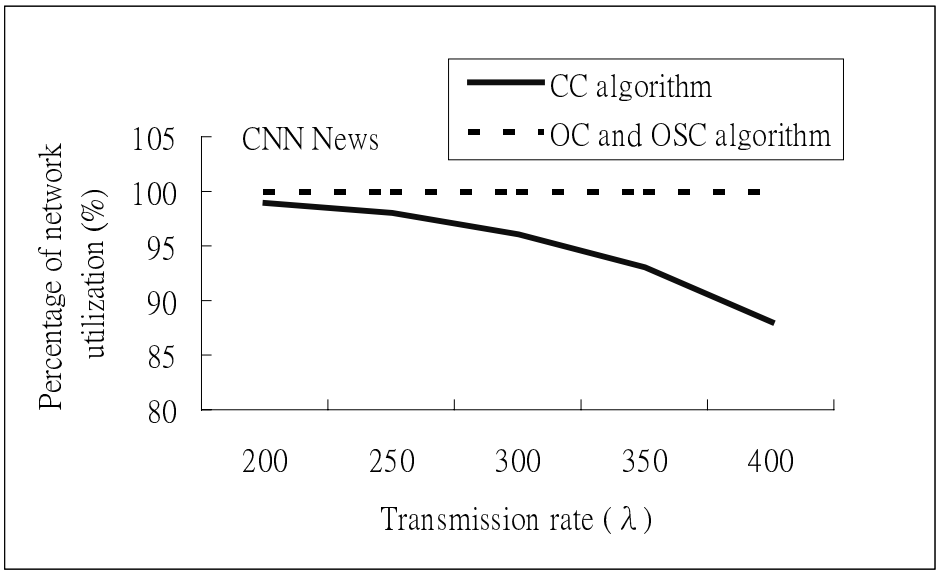


(d)

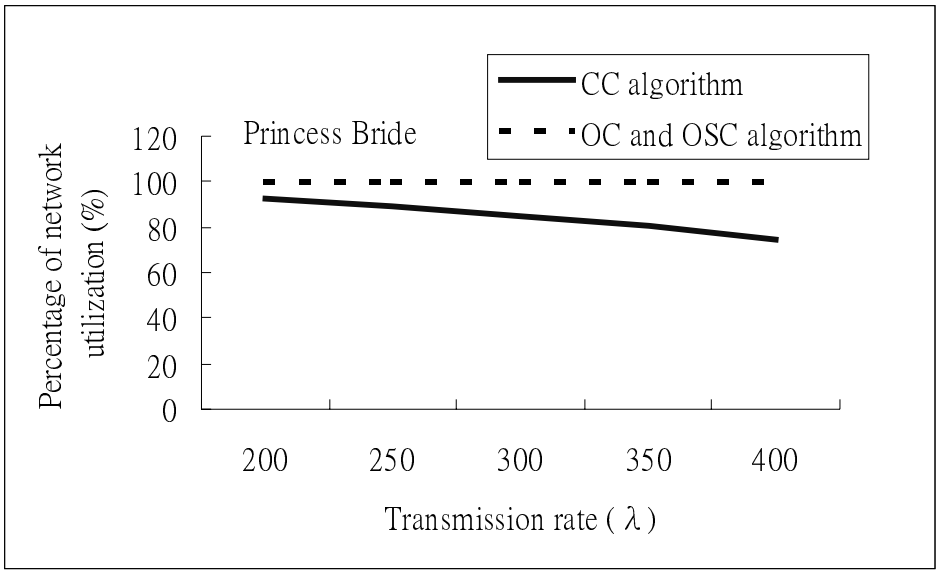
Fig. 12. Percentage of I-frames cached in proxy cache servers for different algorithms. (a)Star War. (b) CNN News. (c)Princess Bride. (d) Advertisements.


CC algorithm
OC and OSC algorithm
Star War
Percentage of network utilization (%)
120
100
80
60
40
20
0
200
250
300
350
400
Transmission rate (λ)

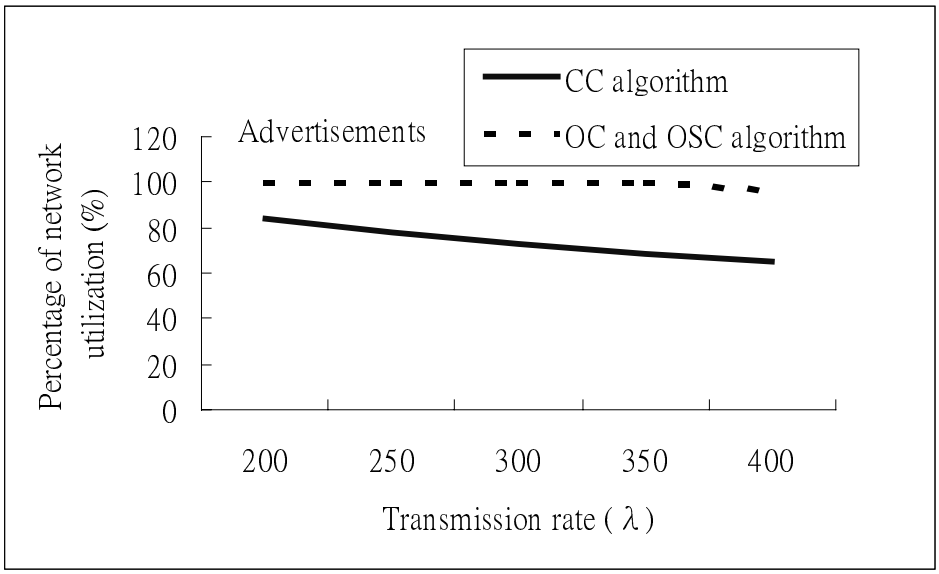
(a)


CC algorithm
OC and OSC algorithm
CNN News
Percentage of network utilization (%)
105
100
95
90
85
80
200
250
300
350
400
Transmission rate (λ)

(b)



(c)



(d)

Fig. 13. Utilization of allocated bandwidth for different algorithms. (a) Star War. (b) CNN News. (c) Princess Bride. (d) Advertisements.