Optimal Proxy Management for Multimedia Streaming in Content Distribution Networks

IBM Chitra Venkatramani · Olivier Verscheure
Pascal Frossard · Kang-Won Lee[1)]

1. Introduction

The widespread use of the Internet and the maturing digital video technology have led to an increase in various streaming media applications such as webcasts, distant learning, and corporate communications. Until the advent of broadband, the cramped last-mile bandwidth was the primary bottleneck in delivering quality streaming media over the Internet. As access providers roll out faster last-mile connections, the bottleneck is shifting upstream to the provider's backbone, peering links, and the best-effort Internet. This problem can be partially addressed by "edge delivery" of streaming objects from a nearby proxy or via content distribution networks. While the edge delivery of streaming media objects will increase scale and reach of streaming media, handling streaming objects brings additional complexities at the proxies due to the large object size, long-lived nature of the objects, and isochronous delivery requirements from the users.

A variety of techniques have been proposed in the literature to efficiently utilize the backbone network bandwidth for streaming. Some of them use multicast as a means to reduce the backbone bandwidth usage: periodic broadcasting [1], [2], [3], [4], simple batching, batching with patching [5], [6], [7] and optimized patching with classes of service [8]. While these multicast-based schemes afford very low backbone bandwidth usage, they are not in widespread use due to their dependency on network-level multicast, which is not widely available over the Internet except for limited instances such as on local area networks. Additional drawbacks of the above approaches include the batching delays, and the need for clients to be able to receive multiple simultaneous streams.

With the recent proliferation of caching proxies, many of these drawbacks can be masked by storing portions of the media object in the proxy to hide the startup latency, and by using application-level multicast when network-level multicast is not available. Related work in this area [9], [10], [11], [12], [13] combined scalable video delivery with proxy caching, where the focus was mostly on transmitting a single video. In [14], the authors studied batching and patching with prefix-caching at the proxy for multiple heterogeneous videos, but analyzed each scheme independently to determine the optimal caching unit. This restricts all assets in the system to be managed using a single scheme irrespective of the difference in their access patterns. In [9], we have investigated the video streaming problem in the context of joint server scheduling and prefix/partial caching strategy at proxy to minimize the aggregate backbone bandwidth usage. However,

1) The authors are with the IBM T.J. Watson Research Center, P.O. Box 218, Yorktown Heights, NY 10598. Contact author : K.-W. Lee(kangwon@us.ibm.com).

this work studied three different schemes in a disjoint framework, for a single asset only.

In this paper, we build on earlier work and propose a unified mathematical framework under which various server scheduling and proxy cache management algorithms for video streaming can be analyzed. More precisely, we incorporate known server scheduling algorithms (*batching, patching, and batch patching*), and proxy management algorithms (*full caching, partial caching, and no caching*, with an option to cache patch bytes or not) in our framework and analyze the minimum backbone bandwidth consumption under the optimal joint scheduling/caching strategy.

To this end, we first study the case of streaming a single video object and analyze the minimum backbone bandwidth consumption to meet the user requests under the optimal scheduling/caching algorithm. In particular, we evaluate the impact of the following parameters: (i) user request rate, (ii) proxy-to-client bandwidth constraints, and (iii) patch caching at the proxy. We then study the case of streaming multiple heterogeneous video objects under the optimal scheduling/caching algorithm. In both cases above, the optimal algorithms are determined by exhaustively searching over all possible combinations of the streaming and proxy management schemes in a given environment. Finally, we derive a simple heuristic that can be implemented at the server and at the proxy in practice. Using simulation, we validate our simple heuristic performs closely to the optimal algorithm under various resource constraints.

The following section presents the problem formulation and builds the mathematical framework under which various schemes are analyzed. Section III analyzes the effect of various parameters on the backbone rate for a single video. A practical algorithm to determine the optimal cache allocation is presented in the context of multiple assets in Section IV. Finally, Section V presents conclusions.

2. Problem formulation

We consider a CDN composed of one origin server, one edge server (or proxy cache) and a finite set of media assets. Also we consider only reliable transmissions (i.e., TCP) with bounded delay over the backbone network. We assume that the access network (i.e., proxy-to-client cloud) is lossless and multicast-enabled. We also assume that the origin server and the proxy are

TABLE 1 THE PARAMETERS USED IN THIS PAPER TO DESCRIBE THE UNIFIED MATHEMATICAL FRAMEWORK

Parameters related to video object $w \in \Omega$	
τ_w	streaming rate
T_w	playback duration
λ_w	average access rate
d_w	admissible palyback delay

Parameters for scheduling and caching for object w	
Δ_w	maximum network jitter ($\Delta_w = \Delta$)
P_w	prefix duration
b_w	virtual batching interval ($b_w = P_w + d_w - \Delta$)
W_w	patching window ($W_w = N_w \times b_w$)
α_w	binary indicator($\alpha = 1$: cache patches)

batch-patching servers with a non-zero batching interval. Finally, every stream from the origin server is constrained to go through the proxy for content adaptation purposes (e.g., adaptive FEC, rate control).

2.1 Preliminaries

Let Ω denote the finite set of media assets the CDN is willing to host. An asset $w \in \Omega$ is characterized by its streaming rate τ_w, its duration T_w, its average access rate λ_w, and its admissible playback delay d_w possibly specified in the Service Level Agreement (SLA).

We consider the following problem : Given the set Ω, find the per-asset joint scheduling and partial caching strategy that minimizes the aggregate backbone rate R under the constraints imposed by the CDN provider infrastructure. These constraints encompass the limited capacity of the proxy in terms of storage S, bottleneck bandwidth B which may be the disk or the network bandwidth. We now propose a unified framework under which various joint scheduling and partial caching strategies may be analyzed.

2.2 Unified Framework

We consider the following scenario illustrated in Fig. 1.

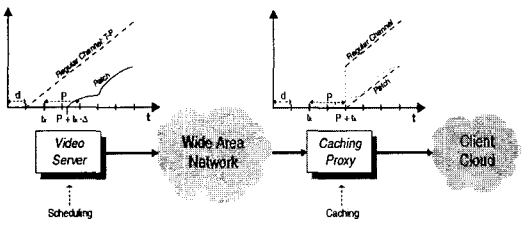

Fig.1 Unified framework for joint scheduling and partial caching strategies

The following description holds for any asset $w \in \Omega$. The proxy cache divides its time axis into intervals $[t_{i-1}, t_i]$ of duration d_w units of time. Any request arriving at time t_1 in $[t_{k-1}, t_k]$ are batched together. At time $t = t_k$, the proxy must start streaming the requested asset to the client, which can be the prefix of duration $P_w > 0$ from the proxy or the whole asset from the origin server (when $P_w = 0$). Clearly, insuring a continuous playback at the client requires sending requests to the origin server in advance to mask the effect of network jitter. Let Δ_w denote this required amount of time for asset w. For the simplicity of exposition, we set Δ_w to the maximum jitter Δ obtained from a long-term measurement. Thus, if the proxy sends a request to the origin server at time $t = t_k - \Delta$. If $P_w > 0$, the prefix is streamed to the client at time and the suffix is requested at time $t = t_k + P_w - \Delta$. Clearly, the condition $P_w + d_w - \Delta \geq 0$ must hold for lossless delivery. Let b_w denote this quantity, i.e., $b_w = P_w + d_w - \Delta$. We call b_w the *virtual batching interval* for the media asset w. Recall that d_w is called the admissible playback delay and must be greater than $\Delta - P_w$. Practically, the finest granularity is achieved by servicing requests on a frame basis. Let ε denote the smallest duration over which requests are batched, and $d_w \geq \varepsilon$. The admissible *playback delay* d_w divides the intervals $[t_{i-1}, t_i]$ into $\frac{b_w}{d_w}$ S inter vals. For simplicity, we assume b_w holds an integral number of d_w. Now, assume the most recent regular channel (RC) for asset w started at time t_s, with $t_s < t_1$ is an integral number of b_w units of time. If t_k is such that $P_w + t_k < t_s + W_w$, i.e., when patching can be applied, the proxy cache joins the RC at time $P_w + t_k$ and streams it through to the client, which buffers the stream while playing back the prefix. Also at time $P_w + t_k - \Delta$, the proxy requests a patch of duration $P_w + t_k - t_s$ and optionally caches it for future requests *within the same patching window*. However if $P_w + t_k \geq t_s + W_w$, a new regular channel of duration $T_w - P_w$ (the prefix of duration P_w is sitting in the proxy) is requested from the server at time $P_w + t_k - \Delta$.

Note that this framework encompasses various

existing server scheduling algorithms and proxy cache management algorithms developed for streaming media applications. More precisely, it can model the following caching strategies: (i) Full caching ($P_w=T_w$). (ii) Partial caching ($0<P_w<T_w$). (iii) No prefix caching ($P_w=0$). At the same time, it models the following scheduling schemes: (i) Batching ($b_w>\varepsilon$ and $N_w=0$). (ii) Patching ($b_w>\varepsilon$ and $N_w>0$). (iii) Batch patching ($b_w>\varepsilon$ and $N_w>0$). In addition, we can model the case when the proxy either temporarily caches the patch bytes ($\alpha=1$) or not ($\alpha=0$).

We now develop the set of equations for the aggregate backbone rate R_w, the proxy storage S_w, network bandwidth B_w and disk reading bandwidth D_w requirements for any media asset $w\in\Omega$.

In the following we assume a Poisson request arrival distribution such that $e^{-\lambda_w x}$ is the probability to have an empty batch of duration x units of time. Please refer to [15] for further details.

The *aggregate backbone* rate at stationary state, R_w, is given by:

$$R_w=\frac{\mu_w b_w r_w+(T_w-P_w)r_w}{I_w} \tag{1}$$

where μ_w denotes the average number of transmitted patches of asset w:

$$\mu_w=\alpha_w\frac{e^{-\lambda_w b_w(N_w+1)}-(N_w+1)e^{-\lambda_w b_w}+N_w}{1-e^{-\lambda_w b_w}}+$$
$$(1-\alpha_w)(1-e^{-\lambda_w b_w})\frac{N_w(N_w+1)}{2} \tag{2}$$

and I_w represents the interval duration between two regular channels:

$$I_w=(N_w+1)b_w+\lambda_w^{-1}, \tag{3}$$

The proxy cache size, S_w, is given by:

$$S_w=\frac{P_w r_w I_w+\Delta r_w(T_w-P_w)}{I_w}+$$
$$\frac{\mu_w[\alpha_w b_w r_w N_w b_w+(1-\alpha_w)\Delta r_w b_w]}{I_w} \tag{4}$$

The proxy batches client requests in intervals of d_w which is the maximum playback delay. The proxy streams the prefix for every batch of d_w, and forwards the patch bytes (every b_w), and the regular channel (every I_w) received from the origin server, to the client. Thus the *proxy network bandwidth*, assuming a multicast-enabled network from the proxy server to the client, B_w, averaged over I_w, is expressed as:

$$B_w=\begin{cases}\dfrac{(1-e^{-\lambda_w b_w})(N_w+1)b_w P_w r_w}{I_w d_w}+R_w & \text{if } d_w>0\\ \dfrac{\lambda_w(N_w+1)b_w P_w r_w}{I_w}+R_w & \text{if } d_w=0\end{cases} \tag{5}$$

In the subsequent sections, we first present analytical results for the single asset case, then we discuss the problem of allocating proxy space among multiple heterogeneous video assets.

3. Analysis

Our goal is to determine the best caching and scheduling schemes such that the backbone usage is minimized. From Equation 1, the backbone rate is a function of various parameters - the cached prefix, the duration T, the rate r and the popularity λ. Other parameters such as the batching interval b and the patching window factor N in the equation depend on the cached prefix and can be determined from it - for example, the value of N is chosen such that R is minimized for a particular prefix and can be obtained by differentiating Equation 1 with respect to N. To simplify the analysis, we assume that the network jitter parameter Δ is set to zero and will elaborate on its effects in the full version of the paper. Of these parameters, R is most affected by the size of the cached prefix (P) and the popularity of the videos (λ).

In Figure 2, we plot the value for the normalized backbone rate R_w against the prefix cached at the proxy (in log-scale), for various values of λ_w or the popularity of the video. We make two significant observations -

· R decreases with the size of the prefix and the decrease is almost linear beyond a prefix size of about 5% of the video.

· The popularity or λ_w has a significant effect on R. Figure 2 shows the plots for 10 values of λ in the range of 10^{-4} and 10. This range is much larger

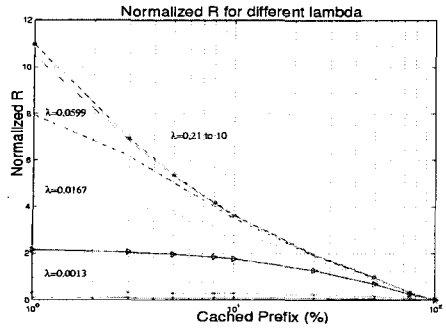

Fig. 2 As the cached prefix increases beyond 5% of the video size, the backbone bandwidth decreases almost linearly on a log-scale, to zero when the entire video is cached at the proxy.

than the range that would be obtained using a Zipf distribution of popularity, but has been chosen here to understand the effect of λ on R. We observe that the plots are distinct for smaller values of λ = 0.0001 to 0.0167, but have near-complete overlap beyond λ =0.0599 requests per batching interval. At this value, the probability of seeing at least one request per batching interval b_w becomes closer to 1. Beyond this, requests get clumped into batches at the proxy and this does not affect the backbone rate any more. This threshold value depends on the batching interval b_w of a video and is denoted by λ^*_w

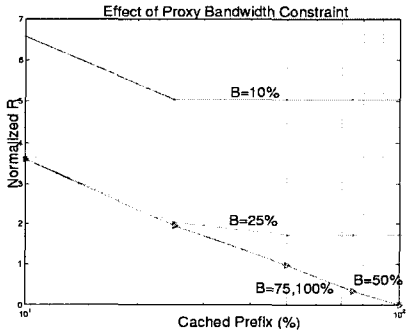

Fig. 3 As the cached prefix increases, the backbone bandwidth decreases until the band width out of the proxy becomes a bottleneck. There is no benefit in increasing the prefix beyond the value at which this happens.

3.1 Proxy Bandwidth Constraint

In the above analysis, the bandwidth out of the proxy was unconstrained, leading to the minimal backbone bandwidth usage. In Figure 3, we study the effect of constraining the network bandwidth out of the proxy. Once the proxy bandwidth is saturated the video must be streamed out from the origin server via a separate channel. The plot shows the region where the prefix is larger than 10% of the video and the points where the bandwidth constraint kicks in are clearly seen and R cannot be reduced further. This implies that during the caching phase, it is not beneficial to cache a prefix greater than his value.

3.2 Caching Patch Bytes at the Proxy

Next we examine if it is beneficial to temporarily cache the patch byte in the proxy. In our equations, the parameter α indicates if the patch bytes are cached in the proxy or not. With the next set of simulations, we determine the effect of α on the value of R. Figure 4 indicates that there is no benefit in caching the patch bytes at the proxy. In other words, this implies that whenever space is available, it is more beneficial to use it to cache the prefix of the video rather than use it to cache patch bytes. The need to cache patch bytes may arise when there is a restriction on the bandwidth available at the client or the number of simultaneous streams that the client device can handle. As described in [9], if only the prefix is cached at the proxy. the client could end up receiving up to three simultaneous streams - prefix from the proxy, patch and regular channel from the origin server. The proxy might be forced to cache the patch when the client cannot receive more than $2x$ the rated bandwidth of the stream. The corresponding increase in R is dependent on the popularity of the object and is indicated in Figure 4, for a moderately popular object, λ =0.0599.

Thus far, we have studied the effects of various scheduling and caching schemes on the usage of

the backbone, in the context of a single asset. We found that the ideal scheduling policy at the origin server is batch-patching, with prefix caching at the proxy. The proxy can determine the patching window N_w*b_w based on the prefix size and request new regular channels from the origin server when this window is crossed. For most practical scenarios, we would be operating in the linear region of the evolution of R_w with the logarithm of the prefix size as indicated in Figure 2, since we would be interested in caching videos of moderate to high popularity and would consider caching a prefix of 5% or more. Given these insights, we can devise an effective cache allocation algorithm in the proxy when dealing with multiple heterogeneous assets, as described in the following section.

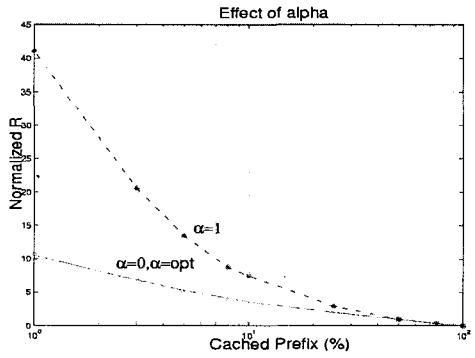

Fig. 4 We reach the optimal R value when $\alpha = 0$, which means that using space at the proxy for patching is suboptimal. Patching will be necessary to reduce simultaneous streams to clients and the cost in R is indicated when $\alpha = 1$.

4. Proxy Algorithm for Multiple Heterogeneous Video Assets

The problem of allocating proxy space to multiple heterogeneous assets can be stated as follows: *Given* a set of media assets $w \in \Omega$ characterized by their streaming rates r_w, durations T_w, access rates λ_w and admissible playback delays d_w. Given also a proxy cache of storage capacity S and network bandwidth B, and a backbone network with bounded jitter Δ. *Find* the tuples (P_w, N_w, α_w) that *minimize* the aggregate backbone rate R such that (i) $S = \sum_{w \in \Omega} S_w \leq S$ and (ii) $B = \sum_{w \in \Omega} B_w \leq B$.

Due to the heterogeneity of video assets in terms of streaming bandwidth, playback duration, and dynamically changing popularity, optimally allocating the proxy cache space to multiple video assets is a challenging problem. In [?], Wang et al. have shown that a simple heuristic based on object popularity and streaming bandwidth failed to provide a good performance. In this section, we first present a brute force optimal algorithm for multiple heterogeneous videos, then present a practical heuristic that gives a close-to-optimal result at a considerably smaller complexity.

```
PrefixAlloc(lambda)
{
  Rgain[i][j] = getR(lambda[i],j*c) -
        getR(lambda[i],(j-1)*c);
  // walk through lists, determine prefixes
  // block[i] - index of the block
  // to pick next for asset i.
  for i=1 to NumVideos
     block[i] = 1; // initialize block[i]
  // while cache is not full.
  while (allocSpace <= cacheSize){
    // find the block with maximum benefit
    for i=1 to NumVideos
      for j=2 to numBlocks
        <video> = find max(Rgain[i][grain[i]])
      RgainTotal += Rgain[video][grain[video]];
      block[video]++;
      Prefix[video] += blockSize;
      allocSpace += blockSize;
  }
}
```

Fig. 5 Algorithm for Proxy Prefix Allocation

Let us assume c to be the smallest unit of cache allocation and all allocations are in multiples of this unit. We also assume that the value of λ_w for each asset is given.[2] The algorithm first computes the value R_w for each increment of the prefix size in units of c. It then determines the incremental savings in R on growing the prefix by

2) For a practical implementation, we can estimate the value of λ_w videos by gathering and analyzing access statistics.

increments of c. Once this is determined, the algorithm greedily fills up the cache by growing prefixes such that the gain in R is maximized. Since R is monotonically decreasing as the prefix size increases as shown in Figure 2, the above greedy algorithm results in the global optimal allocation. The pseudo code of the above algorithm is presented in Figure 5.

The complexity of the algorithm is $O(n|\Omega|)$ for each value of λ_w *whenever* there is a change in λ_w where n is the number of blocks ($n = T/c$). Based on the observations we have made in the previous section, we propose the following heuristics that reduces the complexity of the allocation algorithm significantly:

· From Figure 2, we note that the backbone usage is closely related to the popularity of videos. Thus we classify video assets into a few classes based on their popularity or λ. This way, we can avoid recomputing the backbone rate R every time the λ of any video changes. We recompute the prefixes (using PrefixAlloc) only when the λ of a video changes from one popularity class to another.

· We compute the value of R_w by linear interpolation using the fact that the decrease in R_w can be approximated by a straight line when the prefix is larger than 5%. In this case, we only need to compute the value of R_w for when the prefix is 5%, for the different classes of λ. And since we know that R_w reaches zero when the entire video is cached, we can interpolate between these two points and determine the R at any other value of the prefix in getR() function.

· Since we are give a small number of popularity class and the allocation unit c, we can precompute the relative gain of each block in each popularity class and maintain a sorted list of them. The allocation algorithm now simply selects the first unallocated block in the sorted list. It is easy to show that the above algorithm provides a constant time complexity and it needs to called only when there is a change in the membership of the popularity set.

Figure 6 plots the backbone bandwidth consumption for multiple heterogeneous video assets. From the figure we observe that our simple heuristic closely achieves the optimal rate usage at all ranges of proxy cache space. Generalization of the allocation algorithm to the case when the proxy-to-client bandwidth is constrained will be presented in the full paper.

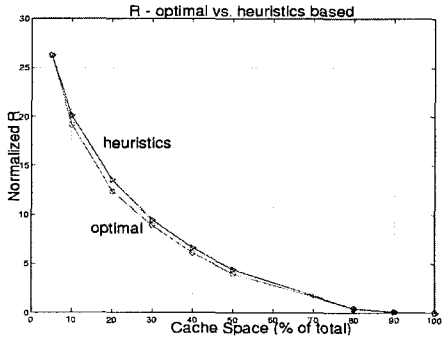

Fig. 6 Backbone bandwidth consumption vs. cache size with multiple video assets with various popularity. Caching granularity is 10% of the size of each video. The value of R decreases as the space in the cache increases

5. Conclusions

In this paper, we address the problem of efficiently streaming heterogeneous videos to clients over a distributed infrastructure consisting of origin servers and proxy caches. We build on earlier work and propose a unified mathematical framework under which various server scheduling and proxy cache management algorithms such as batching, patching, batch-patching with partial caching at the proxy, can be analyzed. We determined that prefix caching at the proxy is most effective strategy, with batch-patching enabled at the origin server. We analyze our model for a single video and derive an efficient algorithm to determine the prefixes to cache at the proxy such that the backbone bandwidth usage is minimized. We also propose simple heuristics to efficiently

determine the cache allocation for multiple, heterogeneous videos. and show that the performance of the heuristics is close to the optimal scheme.

References

[1] C.C. Aggarwal, J.L. Wolf and P.S. Yu, "A permutation based pyramid broadcasting scheme for Metropolitan VOD systems," Proc. of the IEEE International Conference on Multimedia Systems, June 1996.

[2] A.Dan, D.Sitaram and P.Shahabuddin, "Scheduling Policies for an On-Demand Video Server with Batching," Proceedings of ACM Multimedia, Oct. 1994

[3] K.A. Hua and S.Sheu, "Skyscraper Broadcasting: A new Broadcasting Scheme for Metropolitan VOD systems," Proceedings of the ACM SIGCOMM, 1997.

[4] S.Viswanathan and T.Imielinski, "Metropolitan Area Video-on-Demand Service using Pyramid Broadcasting," Multimedia Systems, vol. 4, August 1996.

[5] K.A. Hua, Y.Cai and S.Sheu, "Patching: A multicast technique for True On-Demand Services," Proceedings of ACM Multimedia, Sept. 1998

[6] S.Sen, L.Gao, J.Rexford and D.Towsley, "Optimal patching scheme for efficient multimedia streaming," Proc. of IEEE International Conference on Multimedia Computing and Systems, June 1996.

[7] Y.Cai, K.Hua and K.Vu, "Optimizing Patching Performance," Proceedings of ACM/SPIE Multimedia Computing and Networking, Jan 1999.

[8] Paul P. White and Jon Crowcroft, "Optimized Batch Patching with Classes of Service," ACM Communications Review, October 2000.

[9] O.Verscheure, C.Venkatramani, P.Frossard and L.Amini, "Joint Server Scheduling and Proxy Caching for Video Delivery," Web Caching Workshop 01, Boston, 2001.

[10] P.Frossard and O.Verscheure, "Batched Patch Caching for Streaming Media," submitted to IEEE Communications Letters, 2001.

[11] S.Ramesh, I.Rhee and K.Guo, "Multicast with cache (mcache):An adaptive zero-delay video-on-demand service," Proceedings of IEEE Infocom, April 2001.

[12] S.-H. Gary Chan, "Operation and Cost Optimization of a Distributed Servers Architecture for On-Demand Video Services," IEEE *Communications Letters*, vol. 5, no. 9, Sept. 2001.

[13] Guo Y., Sen S. and Towsley D., "Prefix Caching assisted Periodic Broadcast: Framework and Techniques to Support Streaming for Popular Videos," Tech. Rep. TR 01-22, UMass CMPSCI, 2001.

[14] B.Wang, S.Sen, M.Adler and D.Towsley, "Optimal Proxy Cache Allocation for Efficient Streaming Media Distribution," To appear in Proceedings of IEEE Infocom, 2002.

[15] C. Venkatramani, O. Verscheure, P. Frossard and K.-W. Lee, "Optimal Proxy Management for Multimedia Streaming in Content Distribution Networks," TR 2001-12, IBM Research, December 2001.

Kang-Won Lee

1992 Seoul National University, Computer Engineering, B.S.
1994 Seoul National University, Computer Engineering, M.S.
1996~2000 TIMELY Research Group, University of Illinois, research assistant
2000 University of Illinois at Urbana-Champaign, Computer Science, Ph.D.
2000~IBM T.J. Watson Research, Research Staff Member
E-mail:kangwon@us.ibm.com