Available online at www.sciencedirect.com
SCIENCE DIRECT®
Signal Processing: *Image Communication* 18 (2003) 515–526
SIGNAL PROCESSING: IMAGE COMMUNICATION
www.elsevier.com/locate/image

Neighbors-buffering-based video-on-demand architecture

Tarik Taleb*, Nei Kato, Yoshiaki Nemoto

Graduate School of Information Science, Tohoku Univ. Aoba05, Aramaki, Aoba-ku, Sendai 980-8579, Japan

Received 13 November 2002; accepted 14 February 2003

Abstract

Since the number of Internet users is rapidly increasing day by day and even the most powerful server system will always be resource limited, one of the challenges faced by video-on-demand system designers is how to configure a system that can support a potentially large number of customers and a large multimedia library to satisfy users' needs at affordable rates.

In this paper, we propose an approach to provide a significantly scalable video-on-demand service in a multicast environment. The basic idea is to repeatedly transmit popular movies on staggered channels. If a request comes in between staggered start times, the user joins to the most recently started multicast session and then requests the missing part from a nearby neighbor. The user must have enough buffer space to buffer data between staggered transmissions. We refer to our proposed architecture as Neighbors-Buffering Based Video-on-Demand (NBB-VoD) architecture.

Analytical results show that our proposal achieves a better performance than the already existing systems (True-VoD, Near-VoD, and Unified-VoD) in terms of both scalability and disk-bandwidth requirements.

Keywords: Video-on-demand systems; Multicasting; Scalability

1. Introduction

With the vast improvements in multimedia technologies, video-on-demand (VoD) will become the key residential service in the emerging high-speed networks. In fact, the growth in the number of VoD providers and the operators they work with is not showing any sign of slowing and is a clear testament to the strength of the market [15]. Along with the growth in usage, designing video-on-demand systems has been an active area of research, and interactive VoD services are likely to form a significant component of the workload of the future network applications.

In typical proposals for video-on-demand, customers are serviced individually by being allocated a transmission channel and a set of server resources. To provide such a service, known as True Video-on-demand (True-VoD) [12], the system must reserve a dedicated video channel at the video server and the network for each user. As the number of customer requests increases, the quality of the service can be maintained only by increasing server resources and network bandwidth, which ultimately leads to an expensive to operate, and non-scalable system.

An efficient implementation of multicast technology permits the simultaneous servicing of many

*Corresponding author. Tel.: +81-22-217-7140; fax: +81-22-263-9306.
E-mail address: taleb@nemoto.ecei.tohoku.ac.jp (T. Taleb).

0923-5965/03/$ - see front matter
doi:10.1016/S0923-5965(03)00039-0

users without overloading either the network or the server resources, and thus providing an effective-cost and large-scalable video-on-demand system, known as Near video-on-demand (Near-VoD) [1–3,7,16].

Near-VoD makes use of multicast delivery to service more than one customer with a single set of resources to substantially reduce the system cost and achieve scalability [3,5,9]. In this system, each movie can be multicast using a predetermined number of channels. For each channel, the assigned movie is repeated over and over, and channels transmitting the same movie are offset by a time slot. Movies are available only at the beginning of these slots (say 15–30 min). A customer making a request after the start of a multicast channel will thus have to wait till the upcoming channel starts transmitting the movie. This introduces a significant start-up delay to the customer, which effectively contradicts the on-demand nature of the service.

Unified video-on-demand (Unified-VoD) system unifies the existing True-VoD and Near-VoD systems by integrating unicast with multicast transmissions [11]. In this system, requests arriving after the beginning of a multicast channel, will be immediately served by a unicast stream instead of being scheduled for the upcoming multicast channel. By so doing, the system can reduce the start-up delay in a multicast environment. However, since even the most powerful server system will always be resource limited, some requests may be denied when a large number of users issue requests in a short period of time for a certain number of popular movies. Such cases may happen during the last six evening hours of weekends, known as ''prime time'', when the number of requests for particular popular movies would be considerably high [13].

In this paper, we propose a technique to provide an interactive and significantly scalable video-on-demand service in a multicast environment. Our proposed mechanism attempts to further increase the system capacity by reducing the effective request arrival rate to the video server. The basic idea of our proposal is to take advantage of appropriate buffering of participants of a particular session to satisfy the maximum number of new requests, willing to join the same session, instead of using unicast channels. A number of local video servers (LVSs) are distributed in a wide service-area network (WSA). Each of them serves a particular service area and the WSA central server serves a group of these local areas. Popular movies are locally replicated at each LVS and are repeatedly transmitted on staggered channels. If a request comes in between staggered start times, the user joins to the most recently started multicast session and then requests the missing part from a nearby neighbor. The user must have enough buffer space to buffer data between staggered transmissions. The impact of our proposal on the system is to increase system capacity and make better utilization of available unicast streams. We refer to our proposed architecture as Neighbors-Buffering Based video-on-demand (NBB-VoD) architecture.

By storing frequently requested movies on local servers, most client requests can be locally served resulting in reducing the transmission cost and the backbone WSA total bandwidth requirement. NBB-VoD can clearly achieve further reduction in the traffic load and network congestion. When a user accesses the local video server LVS, the service manager attempts to satisfy his request by establishing an *interconnection* between the user and his nearest *neighbor* to transmit one portion of the video data while the rest is delivered to the user through a multicast channel directly from the video server. If a large number of requests are satisfied in a similar way, then reduction in the backbone LSA bandwidth requirement can be significant. This reduction in the bandwidth requirement yields reduction in the video traffic, which is inherently bursty over both short and long time scales. This reduction yields also an alleviation of the congestion in both the network and the bottleneck due to the local video server.

We believe that NBB-VoD may achieve magnificent performance to provide video-on-demand services to university campus networks where users may have access to the same VoD applications (distance learning) at nearly same time, to mobile multi-user platforms such as airplanes, ships, and trains (via satellite links) where users are close to each other and may desire to watch the same movie.

The remainder of this paper is organized as follows. In the next section, we provide an overview of a typical VoD system architecture and discuss its major components. Section 3 presents our proposal NNB-VoD. We analytically develop our proposal in Section 4 and discuss its analytical results in Section 5. We discuss some implementation issues in Section 6. Concluding remarks are in Section 7.

2. Video-on-demand architecture

Besides the network, there are three other major components in a video-on-demand system. The customer interface device, also called the Set-top-Box (STB), enables the user to control the display and to interact with the program. The service manager uses information about outstanding requests and the availability of resources to accept or reject requests. The video server is responsible for receiving and processing manager signaling and control information. It must retrieve requested data from a variety of storage media while still meeting real-time delivery deadlines in order to achieve a continuous and seamless play-out.

Video servers placement is an important consideration in emerging distributed multimedia applications delivery over wide service-area networks (WSAs), and has been the subject of recent research projects [10,6,8].

Fig. 1 depicts a typical architecture of a VoD system with a two-tiered architecture. For simplicity of discussion, each WSA is assumed to comprise several local service-areas (LSAs) interconnected by a backbone WSA. In order to achieve high performance, LSAs should be decided in a way that the mechanisms for accessing and delivering data in a VoD server are very fast and reliable, as well as being scalable and easy to adapt to users' needs. By storing most popular movies in local video servers, most client requests can be locally served. Consequently, the transmission cost and The WSA backbone bandwidth requirement can be reduced, and high scalability and reliability can be obtained. We assume that the network connecting video servers and users' STB is multicast capable and provides a sufficiently fast channel for delivering control messages.

3. Operational overview of our proposal NBB-VoD

3.1. Key components of our proposal

We assume all the local video servers (LVS) and local service managers (LSM) are similar. We thus can focus on only one of them. We assume that

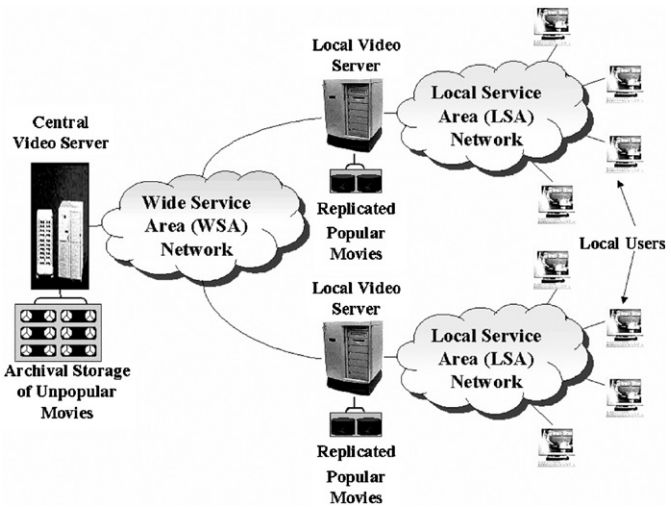

Fig. 1. Server placement.

there are N_u unicast channels, N_m multicast channels, and a certain number of replicated movies at each local video server. Similarly to Near-VoD, each movie is assigned a predetermined number of multicast channels, and for each channel, the movie is periodically repeated over the service time. Data transmission from multicast channels is possible only at the beginning of slots.

At the user side, we assume that all users' devices are similar and have additional storage to cache video data for later playback as proposed in [11]. We intend to use buffering mechanisms similar to those proposed in [3]. Each user's buffer is divided into two parts. One part is used to hold frames that have been already displayed and should be large enough to store a slot's worth of frames. The second part is responsible for providing continuous playback and holding a small amount of un-played frames.

3.2. Session profile

A session is formed by having multiple clients receive the same VoD application and is identified by a unique multicast address [14]. A session group G_n is defined as a group of users listening to the same multicast channel C_n.

When a user A desires to join a particular session, the service manager will provide the user with the session's multicast address and its size.[1] Before the actual reception of the requested video data, the user A will be asked to multicast a *session packet* to all the members of the session. In response, each member should send a *reply packet* to the user A. This latter uses these *reply packets* to estimate the one-way distance[2] between him/her and other members as explained below.

The *session packet* and *reply packets* contain a source-ID and a timestamp. Assume that user A sends a *session packet* P_s at time t_1 and user B receives the *session packet* at time t_2. In response to the *session packet*, user B immediately issues a *reply packet* P_r at time t_3 marked with $(t_3, \triangle)$ where $\triangle = t_2 - t_1$. Upon receiving P_r at time t_4, user A can estimate the latency from user B to user A as

$$D_{AB} = \frac{(t_4 - t_3) + \triangle}{2}.$$

Once distance calculation is done, the user will send a *report packet* to the service manager including information of the one-way distance between the user A and other members. The manager will then use this information to update the *session profile*. Each *session profile* is identified by a name (i.e., movie's name). In addition to information about movie sequence statistics (i.e. frame rate) and users' buffer size, the *session profile* contains the following major elements:

- Client ID—defines a user/member of the session.
- Start-time—defines the time a user started viewing a movie.
- Multicast-channel—defines the multicast channel a user is listening to.
- Buffer-contents[3]—indicates the range of frames a user has in his buffer.
- Interconnections—refers to clients that are connected to a member.
- Establishment-time—defines the time an interconnection was established between two users.
- Expiration-time—defines the time an interconnection will expire.
- Distance—indicates the one-way distance between a user and the other members.

Fig. 2 shows how our proposed *session profile* should typically look. For instance, Client 1 started playing the movie at 21:30:00, is listening to the multicast channel "Titanic-1", and has frames spanning the range from the 1500th frame to the 16500th frame in his buffer. An *interconnection* has been established between Client 1 and Client n at 21:37:00 and will expire at 21:44:00.

3.3. Admission control and scheduling mechanisms

When a user A generates a request at time t_r for a particular movie, the service manager first checks

[1] Number of already existing members.

[2] Distance is calculated in time unit.

[3] Knowing movie sequence statistics (frame rate), user's Set-Top-Box characteristics (buffer size), and the start time of reception, the service manager can estimate the buffer content of each user at any time.

Session Name (Titanic)								
Now = 21:41:00			Buffer Size = 10 min		Frame Rate = 25 frames/s			
Client ID	Start Time	Multicast Channel Number	Buffer Contents (Frames)	Inter-Connections	Establishment time	Expiration time	Other Clients	Distance (ms)
Client 1	21:30:00	Titanic-1	1500th to 16500th	Client n	21:37:00	21:44:00	Client 2	07
							Client 3	32
							||	||
							Client n	27
Client 2	21:33:00	Titanic-1	1st to 12000th	None	---	---	Client 1	17
							Client 3	41
							||	||
							Client n	23
|| || ||	|| || ||	|| || ||	|| || ||	|| || ||	|| || ||	|| || ||	|| || ||	|| || ||
Client n	21:37:00	Titanic-1	1st to 6000th	None	---	---	Client 1	07
							Client 2	23
							||	||
							Client (n-1)	45

Fig. 2. An example of a session profile.

the start time of the nearest upcoming multicast channel C_n transmitting the requested movie. Let t_n be this start time.

If the waiting time, $t_n - t_r$, is smaller than a predetermined admission threshold δ as follows:

$$t_n - t_r \leqslant \delta.$$

The user A will be immediately informed that his/her request has been accepted and scheduled for the upcoming channel. This parameter δ depends on how long we are willing to let customers wait, and should not be more than few seconds to guarantee short latency service. In our numerical results, δ is set to 90 s.

If the waiting time is bigger than the admission threshold δ, the service manager will then check the *session profile* to see if there is any other user containing the requested data in his/her buffer. If an appropriate user B with the desired data is found, the service manager will establish an *interconnection* between the two users A and B. The user A will then receive the already transmitted (from the nearest previous multicast channel C_{n-1}) portion of the movie from the user B and start playing the movie as soon as data become available. Simultaneously, user A will receive the remaining portion of the movie from the nearest previous multicast channel C_{n-1} and store it in his/her local storage for later playback. The *interconnection* between the two users can be released in such a manner that the two portions of the video data must be synchronized to guarantee a continuous playback of the video.

If no user with the desired data is found, the service manager will assign a free unicast stream to the user A to transmit the already-sent part of the movie. At the same time, the user A will cache data from the nearest previous multicast channel C_{n-1} and proceed in the same way as above.

3.4. Interconnections management algorithm

We now explain how the service manager establishes *interconnections* among participants of a particular session. Fig. 3 shows our proposed algorithm for managing *interconnections* establishment among clients. In order to protect user's devices from multiple simultaneous connections and to guarantee quality of service (QoS), we set the maximum number of connections a set-top-box can handle simultaneously without any damages to *Max-conn*. This parameter can be chosen empirically and depends on the characteristics of the set-top-box hardware.

We assume a user A generates a request for a particular video data at time t_r. Let *Neighbor(A)* be the set of clients whose buffer contains frames requested by client A. We note by *Near(A)* the nearest user to client A among elements of *Neighbor(A)*. Let *Conn(B)* denote the number of *interconnections* already established between client B and other users. Recall that each of these *interconnections* has an expiration time that can be deduced from the *session profile*. Let *Min-expire(B)* be the nearest expiration time.

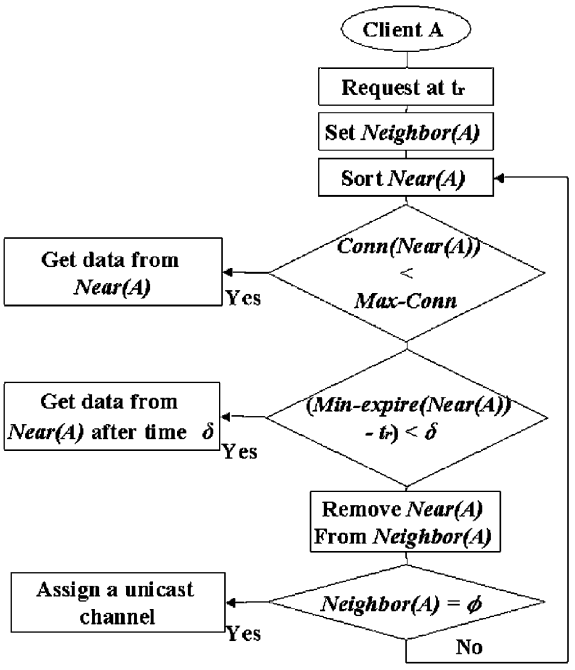

Fig. 3. Interconnections management algorithm.

When user A issues a request at time t_r to join a particular session, the service manager first checks the *session profile* and sorts all the users who have the requested frames within their buffers, *Neighbor(A)*. The service manager will then sort the nearest *neighbor*, *Near(A)*, and checks the number of connections established to him/her, *Conn (Near(A))*. If this number equals *Max-conn*, the service manager will check the nearest expiration time, *Min-expire(Near (A))*. If *Min-expire(Near (A))*$-t_r$ is smaller than the predetermined threshold δ as follows:

$$Min - expire(Near(A)) - t_r \leqslant \delta.$$

The user A will be requested to wait for a certain time till an *interconnection* is established between users A and *Near(A)*. Otherwise, the service manager will remove *Near(A)* from the set *Neighbor(A)*, sort again the nearest neighbor among elements of the new set *Neighbor(A)*, and proceed in the same way as explained above. This operation should be done till an available *neighbor* with the appropriate video data is retrieved. If the set *Neighbor(A)* becomes empty before an *interconnection* could be successfully established, the service manager will then assign a free unicast channel to satisfy user A's request and operate according to the proposed admission control algorithm.

3.5. Dealing with packet losses

An important issue is how packet losses, in a congested network, would affect the buffer content's prediction and how the service manager should deal with packet loss occurrence. Measurement study of Internet traces shows that even in case of multiple retransmissions, the time required for a host to recover a packet loss, over a terrestrial wide-area network (WAN), is in order of few seconds. We note by χ the maximum time needed to recover a packet loss in a local service-area (LSA). We set this parameter to 15 s in our numerical results, unless specified otherwise. We assume that end-users are designed to be resilient to packet losses and delays in network, that all the losses occur in the network, and that there is no

loss caused by the deficiency of the server or the client.

Let us illustrate how the service manager should deal with packet losses via a simple example. As explained before, when a new user A issues a request, the service manager initially attempts to establish an *interconnection* between him/her and the nearest client, *Near(A)*, that has the required frames buffered. We envision two cases:

Case 1: the nearest client started receiving video data a time longer than χ ago, and has consequently more than χ's worth of frames buffered.

Case 2: the nearest client has just started receiving video data within a time shorter than χ.

In the former, even if *Near(A)* misses a packet, this loss would be recovered while client A is getting data from *Near(A)* and the packet loss would not affect the service manager's estimation of client *Near(A)*'s buffer content. However, in the latter, there is a risk that the missed packet would not be retransmitted to *Near(A)* before it should be sent to the new user A. To avoid such cases, the service manager should not consider users that have joined a session in a time shorter than χ in the *Interconnections Management Algorithm* to satisfy new requests.

4. System key parameters

We assume that there are M popular movies of average length L replicated locally at each LVS. In our numerical results, we consider the case of 10 popular movies ($M = 10$) and L is set to 90 min. As in Near-VoD, each movie is assigned a predetermined number of multicast channels, and for each channel, the movie is periodically repeated over VoD service time. Multicast channels transmitting the same movie are offset by a slot time W. It is assumed that all multicast and unicast streams are statistically identical with a transmission capacity C.

The unicast channels share the same request queue and serve incoming requests in the first-come-first-served (FCFS) discipline. They can thus be considered as a True-VoD virtual server. We define *unicast usage time* as the time from when a unicast channel is assigned to a user to when it is released. We note by T the *average service time* of unicast channels which can be measured from empirical data. Unless specified otherwise, we set T to 4 min in our analytical measurements.

If we assume that the service time of unicast channels is exponentially distributed with mean T, the unicast channels can be modeled then as an $M/M/n/n+N$ queue [4], where N is the queue capacity. No queuing is assumed in our analysis ($N = 0$), for the simple reason that queuing may cause longer service response delay in case of high arrival rates, which may ultimately effect the short-latency nature of VoD service [12].

At the user side, we assume that all users' devices are similar and contain sufficient extra buffering to hold a time slot's worth of frames. The request arrival process is assumed to be Poisson with arrival rate λ. This assumption is appropriate because the number of VoD users is typically large and users generate the service requests independently.

In our proposal, significant gains can be achieved in the case of popular movies. Hence, we focus in our evaluation model only on popular movies that are replicated at each LVS. Fig. 4 shows the request probability distribution for the M popular movies. We assume that the viewing probabilities of videos follow a normalized geometric distribution. We classify the M popular movies in order of their popularity. The

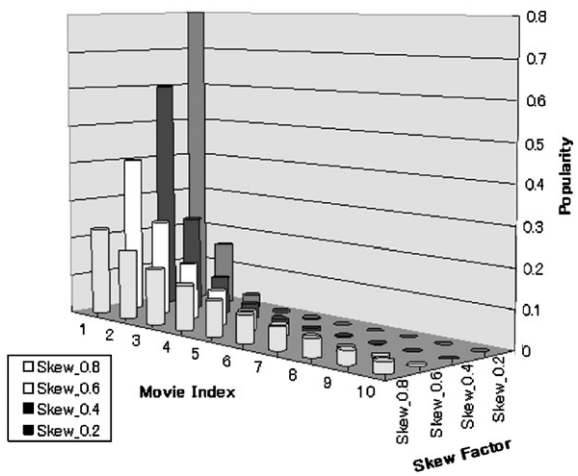

Fig. 4. Movies viewing probability model.

Table 1
System parameters

Factor	System parameters and range of values
Arrival rate λ	0.1–2 (request/s)
Number of unicast channel N_u	300–700
Number of popular movies M	10
Movie length L	90 min
Skew factor θ	0.2–0.8
Admission control threshold δ	90 s
Average service time T	4 min
Slot time W	10–25 min
Queue capacity N	0

probability that ith video is selected is given then by

$$P_i = \frac{(1-\theta)\theta^i}{\theta(1-\theta^M)} \quad \text{where } i = 1, 2, \ldots, M. \tag{1}$$

The parameter θ is called the skew factor. Fig. 4 shows that setting θ to larger values yields a uniform distribution while setting θ to values close to 0 yields highly skewed distribution. Table 1 shows a complete list of the system parameters and the range of values studied.

5. Analytical results

5.1. Performance gain in terms of disk bandwidth

We calculate the average disk-bandwidth requirements for a particular session group. Assuming the request arrival process to be Poisson process with arrival rate λ, the requests inter-arrival times are mutually independent and identically distributed. We assume that request arrivals are separated by τ time units and that the first request arrives τ seconds after the start of the most recent channel (Fig. 5).

5.1.1. Case of unified video-on-demand

Required unicast bandwidth during one slot time W is

$$BW_u = \tau \cdot C + 2\tau \cdot C + \cdots + \beta\tau \cdot C = \frac{\beta(\beta+1)}{2}\tau \cdot C,$$

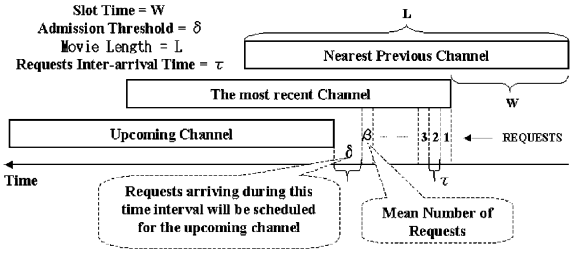

Fig. 5. Performance evaluation in terms of disk bandwidth requirement.

where $\beta = \lfloor (W-\delta)\lambda \rfloor$ is the mean number of requests that arrive during one slot time and necessitate unicast channels' usage. On the assumption of Poisson process, the probability density function of τ is

$$f(\tau) = \lambda e^{-\lambda\tau}.$$

Hence, the average value of unicast bandwidth demand is

$$BW_u^{avg} = \frac{\beta(\beta+1)}{2} \cdot \frac{C}{\lambda}\{1 - (1 + \lambda(W-\delta))e^{-\lambda(W-\delta)}\}.$$

On the other hand, as a multicast stream starts transmitting video data after the arrival of the first request and requests with a waiting time less than δ are scheduled for the upcoming multicast channel, multicast bandwidth demand during one slot time is

$$BW_m = \begin{cases} (L-\tau)\cdot C, & 0 \leqslant \tau \leqslant W-\delta, \\ L \cdot C, & W-\delta < \tau \leqslant W. \end{cases}$$

On the assumption of Poisson process, the average value of multicast bandwidth demand during one slot time is as follows:

$$BW_m^{avg} = \frac{C}{\lambda}\{L\lambda - 1 + (1 + \lambda(W-\delta))e^{-\lambda(W-\delta)} - L\lambda e^{-\lambda W}\}.$$

Hence, the average value of total bandwidth demand in case of Unified-VoD is

$$BW_{Unified}^{avg} = \frac{C}{\lambda}\left\{\frac{\beta(\beta+1)}{2} + L\lambda - 1 + \left(1 - \frac{\beta(\beta+1)}{2}\right) \times (1 + \lambda(W-\delta))e^{-\lambda(W-\delta)} - L\lambda e^{-\lambda W}\right\}. \tag{2}$$

5.1.2. Case of our proposal NBB-VoD

We assume that all users' devices are similar and contain sufficient extra buffering to hold one slot time's worth of frames.

For the first request, we assign a unicast channel to transmit τ time units' worth of data, while the remainder of the video data will be transmitted via a multicast channel. For the upcoming requests, we consider two cases, namely $0 \leqslant \tau \leqslant \chi$ and $\chi < \tau \leqslant W - \delta$.

In the former case, the first $\zeta = \lfloor \chi\lambda \rfloor$ requests will be assigned unicast channels, while the remaining $\beta - \zeta$ requests will be satisfied from the previous users' buffering as follows. For the kth new request, the nearest and most available (old) user will be requested to send his buffer contents to the new user. At the same time, the new user will get the rest of data from the same multicast channel as the old user. The service manager will proceed in the same manner to satisfy new requests attempting to establish *interconnections* between users and their nearest neighbors. By so doing, the required unicast bandwidth in case of $0 \leqslant \tau \leqslant \chi$ is

$$BW_{\mathrm{u}} = \frac{\zeta(\zeta+1)}{2}\tau \cdot C.$$

In case of $\chi < \tau \leqslant (W - \delta)$, for the first request, we assign a unicast channel to transmit τ time units' worth of data, while the upcoming requests will be satisfied from their neighbors' buffering. Hence, the required unicast bandwidth is

$$BW_{\mathrm{u}} = \begin{cases} \dfrac{\zeta(\zeta+1)}{2}\tau \cdot C, & 0 \leqslant \tau \leqslant \chi, \\ \tau \cdot C, & \chi < \tau \leqslant (W - \delta). \end{cases}$$

On the assumption of Poisson process, the average of unicast bandwidth requirements in case of NBB-VoD is

$$BW_{\mathrm{u}}^{\mathrm{avg}} = \int_0^{\chi} \frac{\zeta(\zeta+1)}{2}\tau \cdot Cf(\tau)\,\mathrm{d}\tau + \int_{\chi}^{W-\delta} \tau \cdot Cf(\tau)\,\mathrm{d}\tau.$$

As multicast bandwidth requirements in case of NBB-VoD is identical to that of Unified-VoD, the average value of total bandwidth demand in case of NBB-VoD is

$$BW_{\mathrm{NBB}}^{\mathrm{avg}} = \frac{C}{\lambda}\left\{\frac{\zeta(\zeta+1)}{2} + L\lambda - 1 + \left(1 - \frac{\zeta(\zeta+1)}{2}\right)(\lambda\chi + 1)\mathrm{e}^{-\lambda\chi} - L\lambda\mathrm{e}^{-\lambda W}\right\}. \quad (3)$$

Finally, we deduce the performance gain G of NBB-VoD over Unified-VoD as

$$G = \frac{BW_{\mathrm{Unified}}^{\mathrm{avg}}}{BW_{\mathrm{NBB}}^{\mathrm{avg}}}. \quad (4)$$

We plot the performance gain of NBB-VoD over Unified-VoD versus arrival rate λ in Fig. 6. Our results show clearly that our proposal significantly outperforms Unified-VOD. As the arrival rate increases, our proposal shows better performance in terms of disk-bandwidth mainly for large values of time-slots, in other words when only few multicast channels are assigned to the movie. This gain can be effectively exploited to improve resources (disk-bandwidth) utilization at the local video server (LVS).

5.2. Performance gain in terms of scalability

We compute numerical results from the $M/M/n/n+N$ queuing model to evaluate the performance of NBB-VoD over Unified-VoD in terms of blocking probability.

At each LVS, we distribute the N_{u} available unicast channels among the M popular movies in

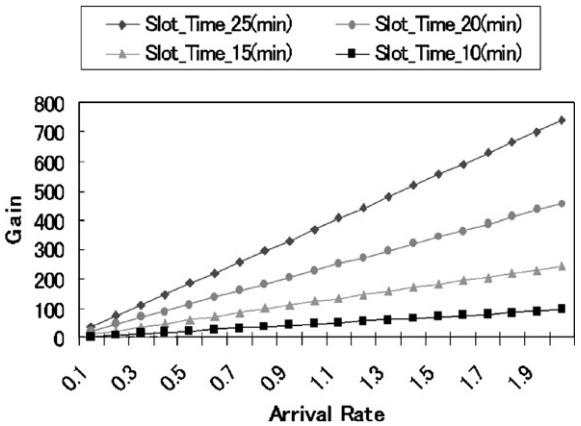

Fig. 6. Performance gain in terms of disk bandwidth requirement.

function of their popularities. In other words, for the kth movie, we assign $P_k \cdot N_u$ unicast channels. In our performance evaluation, we mainly focus on the performance of our proposal in case of requests for the first popular movie. Similar results are obtained for other popular movies.

Assuming that the requests arrival rate for a particular movie in case of Unified-VoD is λ, as NBB-VoD attempts to reduce the number of effective requests to the video server, this arrival rate should be reduced by a factor c in case of NBB-VoD. Intuitively, this factor depends largely on the movie popularity.[4] In our analytical results, we assume that this factor is equal to the movie popularity.

First, we investigate the impact of the number of unicast channels in a LVS on the blocking probability. Fig. 7 illustrates the blocking probability versus arrival rate for different number of unicast channels. The skew factor θ is fixed to 0.8. As the number of requests increases, so does the blocking probability in both Unified-VoD and NBB-VoD. It is observed also that blocking probability decreases as the number of available unicast channels increases. Fig. 7 shows also that the Unified-VoD system has a higher blocking probability than our proposal NBB-VoD as the number of requests increases.

To investigate the impact of movie popularity on system performance, we plot the blocking probability as a function of the requests arrival rate for different skew factors θ in Fig. 8. We fix the number of unicast channels to 300. The results show that the blocking probability increases for larger values of θ in both systems. This increase can be explained in terms of the number of unicast channels assigned to each popular movie: the larger the skew factor θ, the higher the popularity of the first popular movie, the more assigned unicast channels. For smaller arrival rates, blocking probability increases sharply and more rapidly in Unified-VoD than in NBB-VoD. However, the two systems have similar blocking probability as they approach their capacity (larger arrival rates λ).

[4] The more popular the movie, the larger the factor c.

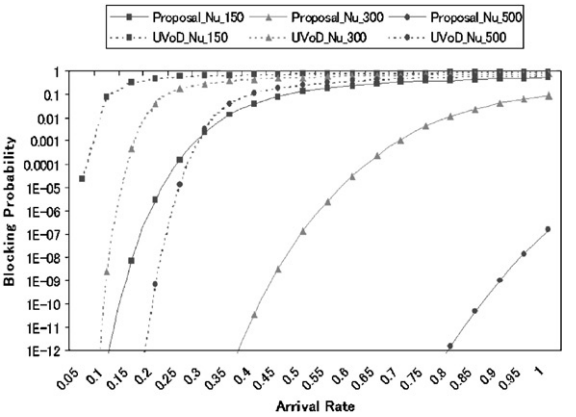

Fig. 7. Blocking probability vs. arrival rate for different number of unicast channels ($\theta = 0.8$).

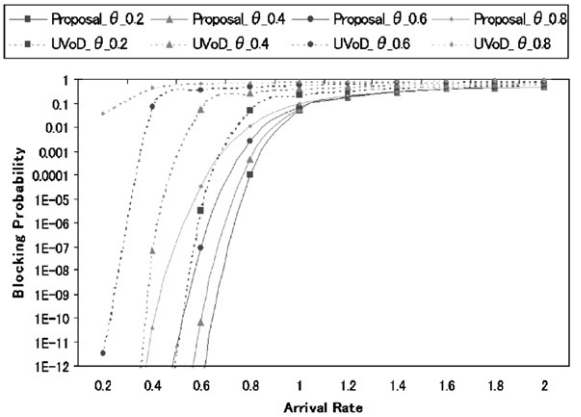

Fig. 8. Blocking probability vs. arrival rate for different skew factors ($N_u = 300$).

6. Implementation issues and discussion

It should be emphasized that there are several implementation issues that must be resolved when applying our proposal to practice. For instance, user's devices should be capable of sending data among themselves in a secure way that prevents illegal intruders from having any unauthorized access. Our proposal could be costly in terms of other resources. For example, as we explore options for optimally making use of user's buffering to increase the system capacity, new architecture for STB, more network bandwidth, and management software may be required. This will

intuitively incur more substantial overhead at both the user and system sides. Since our proposal attempts to service a large population of customers, this additional cost can pay for itself in a short time of VoD service utilization. Evaluation of the cost of the new requirements is outside the scope of this paper and will be left as a separate issue for software developers and hardware designers.

In our analysis, we considered wide-bandwidth networks especially designed for providing video-on-demand service. We assumed also, that each end-user has sufficient bandwidth resources to establish up to *Max-conn interconnections* and is capable to stream its buffer content to other users. However, we are aware that these assumptions are not valid for networks where upstream bandwidth is limited (providing VoD service to cable modem or DSL subscribers with asymmetric links) and can become congested if users are asked to serve-up video for other clients. In such networks, the selection of the nearest user to a new client A, *Near A*, should be based not only on the distance between the two clients (A and *Near A*), but also on the available bandwidth of the connection from *Near A* to A. A weighted combination of the two parameters should be taken into account. Besides, the parameter *Max-conn* should not be constant, but rather needs to adapt to the available bandwidth.

7. Concluding remarks

In this paper, we have developed a technique to provide a significantly scalable VoD service. Our proposal NBB-VoD further enhances and strengthens the multicast group concept by maximizing information and data sharing among members of a particular session.

We first introduced a hierarchical distributed architecture for our system. Popular movies are locally replicated to achieve high availability and scalability. One of the major objectives of this architecture is to reduce the backbone WSA bandwidth requirement. Thereafter, we presented an admission control algorithm and a set of mechanisms to update the *session profile* and to handle *interconnections* among users.

The importance of our proposal is verified by numerical results, which show that NBB-VoD significantly outperforms Unified-VoD in terms of the disk bandwidth requirement. For different movies with different viewing probability, NBB-VoD guarantees lower requests blocking probability than Unified-VoD.

NBB-VoD may achieve magnificent performance to provide video-on-demand service to university campus networks where numerous users may have access to the same VoD feature at nearly the same time, and to mobile multi-user platforms such as airplanes, ships, and trains (via satellite links) where users are close to each other and may desire to view the same movie.

References

[1] E.L. Abram-Profeta, Providing unrestricted VCR functions in multicast video-on-demand servers, in: Proceedings of the IEEE International Conference on Multimedia Computing and Systems, Austin, TX, July 1998, pp. 66–75.

[2] C.C. Aggarwal, J.L. Wolf, The maximum factor queue length batching scheme for video-on-demand systems, IEEE Tran. Comput. 50 (2) (February 2001) 97–110.

[3] K.C. Almeroth, The use of multicast delivery to provide a scalable and interactive video-on-demand service, IEEE J. Sel. Areas Commun. 14 (6) (August 1996) 1110–1122.

[4] G. Bolch, Queueing Networks and Markov Chains, Wiley-Interscience, New York, 1998.

[5] M.K. Bradshaw, B. Wang, Periodic broadcast and patching services—implementation, measurement, and analysis in an internet streaming video testbed, UMass Computer Science Technical Report 2000-56.

[6] D. Eager, M. Ferris, Optimized regional caching for on-demand and data delivery, in: Proceedings of the Multimedia Computing and Networking (MMCN' 99), San Jose, CA, January 1999.

[7] L. Golubuchik, J.C.S. Lui, Adaptive Piggybacking: a novel technique for data sharing in video-on-demand storage servers, Multimedia Systems 4 (3) (1996) 140–155.

[8] R.H. Hwang, Y.C. Sun, Optimal video placement for hierarchical video-on-demand system, IEEE Trans. Broadcasting 44 (December 1998) 392–401.

[9] K. Hua, Y. Cai, Patching: a multicast technique for true video-on-demand services, in: Proceedings of the ACM Multimedia, Bristol, UK, September 1998.

[10] Y.C. Lai, Y.D. Lin, H.Z. Lai, A hierarchical network storage architecture for video-on-demand services, in: Proceedings of IEEE 21st Conference on Local Computer Networks, Minneapolis, MN, October 1996.

[11] J.Y.B. Lee, UVoD: an unified architecture for video-on-demand services, IEEE Commun. Lett. 3 (9) (September 1999) 277–279.
[12] V.O.K. Li, Performance model of interactive video-on-demand systems, IEEE J. Sel. Areas Commun. 14 (6) (August 1996) 1099–1109.
[13] T. Little, D. Venkatesh, Prospects for interactive video-on-demand, IEEE Multimedia 1 (1994) 14–24.
[14] C.K. Miller, Multicast Networking and Applications, Addison-Wesley, Reading, MA, 1999.
[15] J.P. Ourand, Growing market and shrinking costs, Commun. Technol. Internat. (August 2000).
[16] H.k. Park, Multicast delivery for interactive video-on-demand service, in: Proceedings of the 12th International Conference on Information Networking, January 1998, pp. 46–50.