

Resource Requirements of Closed-Loop Video Delivery Services

Wei Shu and Min-You Wu

Department of Electrical and Computer Engineering
The University of New Mexico

Abstract — Different video delivery techniques have been proposed for closed-loop video service, such as Batching and Patching. However, the relationship among the system resource requirement, the size of video repository, and the number of clients is not well understood yet. This article provides a capacity analysis. It is found that there is a threshold of the arrival rate, after that the system resource does not increase or slowly increases with the arrival rate. The threshold depends on the number of videos, video length or Batching time, and request distribution. The threshold for Batching and Patching are presented. These methods are scalable to only a small repository of videos. A new method, Scheduled Video Delivery (SVD), is analyzed. Content and service providers may use these analysis results to compute the system resource requirement, maximum number of videos, and the maximum number of clients that can be served, as well as making selection from various video delivery techniques.

1. Introduction

Continuous media, such as digital movies, video clips, and music, are becoming an increasingly common way to convey information, entertain and educate people. Similar to the transition from the text-dominant web onto the image-enriched web around 1990s, we are about to experience a step towards video-enhanced Internet within this decade. For convenience, we will use *video* as a general term to imply any continuous media in this article. Entertainment exhibits a great demand of video delivery services. Moreover, applications of video delivery services go much beyond this scope. Distance learning, news service, telemedicines, collaborative environment, and many other fields share the variety of demands on video delivery services. Video-on-Demand (VoD) is gaining popularity in recent years with the proliferation of broadband networks. However, limited system and network resources have delayed the widespread usage of continuous media. A careful study on a video delivery system from aspects of system resources, content attributes, and client characteristics can provide a better understanding for the future video delivery systems.

Various video delivery methods have been developed, such as Batching [1], Patching [2], and Scheduled Video Delivery (SVD) [3]. However, many problems remain unsolved. From the clients' point of view, in addition to the low cost, they want to have more contents to choose from. A content provider is willing to serve as many clients as possible, but a service provider wants to minimize the resource con-

sumed. Given a video repository and a request arrival rate, how many channels need to be reserved? A cable operator may need to know how many digital channels should be allocated to maximize profit. Given a video repository and a fixed number of channels, how many clients can be served? In a different situation, for a given number of channels and a client set, the operator might need to know how many videos can be served. From previous research, people understand that given a small number of videos, Batching may serve an unbounded number of clients. However, we do not know what is the maximum number of videos that can be served with Batching. Patching can enhance performance by generating small patching streams, but people do not know how much can be enhanced. A quantitative approach is required to answer these questions. Specifically, the correlation between the capacity of video delivery and other system parameters such as the number of videos, the video length, request distribution, and request arrival rates will provide a clear picture of the video delivery system.

It is especially worthwhile to thoroughly understand how the number of required channels increases with the request arrival rate. It is found that the number of channels required does not increase or slowly increases with the number of requests when the arrival rate reaches a certain threshold. In addition, the delivery strategies exhibit different characteristics in terms of their threshold values as well as the number of channels required after crossing over the threshold. The analysis shows that a system with a large number of videos demands more bandwidth. Batching or Patching with a video repository of more than 1,000 video objects is expensive to implement.

In this article, Section 2 presents a brief overview in video delivery systems. Section 3 models the parameters used in the analysis and Section 4 discusses various capacities of video delivery systems. Capacity analysis of Batching, Patching, and SVD is presented in Sections 5, 6, and 7, respectively. Section 8 gives the performance comparison of these methods.

2. Video Delivery System Overview

Two approaches for scalable video delivery have been developed. The open-loop approach [4, 5, 6, 7] requires no return path so it can be used for one-way cable systems, whereas the closed-loop approach [8, 9, 10, 11, 12] requires a two-way system. The open-loop system continuously broadcasts a video even if no one watches it. It can be efficient if a large number of users are viewing a limited number of videos. On the other hand, the closed-loop system is more efficient in general since it only delivers the videos that are requested by users. Two major classes of schemes have been proposed for the open-loop approach. The first one is Near Video-on-Demand [13] which simply rebroadcasts the same video in a fixed time interval, for example, every 15 minutes. It results in a long average waiting time unless it is broadcast frequently. The second scheme broadcasts the earlier segments of the video more frequently than the later segments to reduce the waiting time [4, 5, 6]. It can only improve the waiting time if the viewer watches the video

from the beginning. However, the reviewer cannot jump to the middle of a video. Normally, only dozens of videos can be provided simultaneously in an open-loop system.

The closed-loop system only delivers the requested video to users. The true VoD is a simple but expensive implementation, where the server and the network deliver videos for individual requests. It is not scalable because of its high bandwidth requirement. Batching [8, 1, 9, 14, 15] and Patching [2, 11, 16] are more efficient. These methods combine requests to minimize the number of broadcast or multicast streams, requiring less system resources. Patching is an important technique as it provides immediate response. Other methods, which are similar to Patching, include catching [17], stream tapping [18], and stream merging [19]. Most of these methods assume there is some storage space available in the client's machine and require substantial bandwidth. Many methods claimed themselves as scalable in terms that the number of required channels does not increase with the number of requests. Unfortunately, it is true only for a small number of videos, but not true for a large video repository. Five to ten channels are needed for each popular video no matter how many people are watching. However, for a video repository of 1,000 popular videos, 5,000 to 10,000 channels are required, which is usually not feasible. In fact, the number of required channels would not be so large, since not every video in the repository is popular. Nonpopular videos may only be requested once, or not at all. On one hand, the number of channels required is at most equal to the number of requests during the video length. For example, if there are 1,000 requests during the video length of two hours, at most 1,000 channels are needed. It is true in the true VoD. On the other hand, with a multicast-enabled network, the number of channels can be significantly reduced since many requests can be combined. With a huge number of requests, say, one million requests per video length, the total number of channels is bounded by the number of videos (N) multiplied by x , which is defined as the number of channels required per video. Different strategies establish their own function for x . For Near Video-on-Demand or Batching, $x = L/T$, where L is the video length and T is the repeat time interval or the batching time. For Patching, x can be roughly modeled as $x = \ln c + 1$, where c is the number of requests [20]. To our knowledge, there is no existing work in the literature to tell us what is the sufficient condition to reach this upper bound, nor on how many channels are required before the upper bound is reached. In this article, detailed analysis is provided and formulas are derived for these problems. We assume that a video will be played without pausing. However, a user can pause the video and play it later provided certain local storage is available.

3. Video Delivery System Modeling

There are three components in a video delivery system, video contents repository, client requests, and system resources. A client request can be satisfied if and only if the requested content is available in the repository and the system resource can be allocated. Capacity analyses in this article focus on a systemwide correlation among all of these components.

Video Repository

The number of video objects, N , determines the size of the video repository. In the video repository to be considered, all video objects are assumed to have the same length, L , measured in minutes. Though video objects may have different bit rates and a single video object may be variable-bit-rate (VBR), for simplicity, all objects are assumed have the same constant bit rate (CBR). Therefore, delivery of video objects is considered as a CBR stream. In most video delivery systems, a variable-bit-rate video object can be smoothly delivered by utilizing buffering strategies.

The popularity of a video object varies from time to time. To make the analysis easy to understand, all the video objects are sorted in a descending order of their popularities. Thus, video object O_1 is the most popular and video object O_N the least.

The size of the video repository has a great impact on the capacity of a delivery system. In general, users always love to have a large variety of video objects to choose from, which demands more system resources.

Client Requests

Two parameters describe the property of client requests on video objects. The first parameter, a , indicates the load of the video delivery system. Usually, clients requests are not uniformly distributed in every minute. Despite the fact that the arrival of requests can be modeled in several different ways, a is used to represent the overall mean arrival rate. For simplicity, we assume that the request arrival times are evenly distributed.

In general, not every video object in the repository is equally accessed. Some video objects are more frequently accessed compared to the others, and therefore, are called *hot* video objects. The probability that a request accesses a video object j can be modeled as:

$$p_j = \frac{C}{j}, \quad \text{where } C = \frac{1}{\sum_{j=1}^N \frac{1}{j}}.$$

This is called the Zipf distribution [21]. Zipf's law is named after the Harvard linguistic professor George Kingsley Zipf. Applying the Zipf's law to the video repository, video objects are ranked by their popularities as mentioned above. Unlike the application of Zipf's law to the frequency of English words in articles, most online contents, such as web pages and videos, exhibits a more or less concentrated accesses. Thus, the Zipf distribution can be generalized as:

$$p_j = \frac{C}{j^\alpha}, \quad \text{where } C = \frac{1}{\sum_{j=1}^N \frac{1}{j^\alpha}},$$

where α is an important parameter representing distribution of client requests. Glassman was the first to

use Zipf's law to model the distribution of web page requests [22], where about 100,000 HTTP requests were gathered. It is found that the request distribution fits Zipf's law with $\alpha = 1$ quite well. Cunha et al. [23] gathered 500,000 web accesses and found that the distribution of web requests follows a Zipf-like distribution with $\alpha = 0.982$. Breslau et al. [24] investigated the requests on six different web servers, where the α value varies from trace to trace, ranging from 0.64 to 0.83. Thus, we set $0.6 \leq \alpha \leq 1$. Most of this study assumes $\alpha = 1$. A sensitivity analysis of α will also be given. In general, parameter α reflects more detailed characteristics of client requests and will have substantial impacts on the required system resources as well. When the request arrival rate is a , the arrival rate for video object O_j can be computed by

$$\lambda_j = a \cdot p_j = \frac{C \cdot a}{j}. \quad (1)$$

System Resources

Considering the network bandwidth as well as the server's output capacity, the total available bandwidth W can be equally partitioned into M channels if every video streams has the same bit rate R , where $M = W/R$. Once M , the number of channels, is fixed, the client requests can be mapped onto the time domain of communication channels.

To efficiently utilize the system resources, we also assume a broadcast or a multicast scheme so that a number of client requests on the same video object can be combined and delivered as a single stream, occupying only one communication channel. A Hybrid Fiber/Coax (HFC) system can be used for this purpose. This requirement also can be satisfied with the Internet with multicast support.

4. Video Delivery System Capacity

Service capacity

A video delivery system is designed to provide services of a video repository to client requests. As usual, resources of such a video delivery system are limited. When the arrival rate is low, all requests can be served. For a large arrival rate, some requests may be rejected due to lack of resources. Let ϕ be the rejection rate, where $0 \leq \phi \leq 1$. With respect to arrival rate a , a video delivery system can serve $(1 - \phi)$ of the requests. Overall, from the service perspective, the *service capacity* is defined by the effective arrival rate a_E ,

$$a_E = (1 - \phi) \cdot a$$

Here, a_E represents how many client requests can be successfully served. A *perfect service capacity* is defined as $a_E |_{\phi=0}$.

Baseline requirements and system reduction ratio

From a system provider's perspective, the resource capacity of a video delivery system is measured by the number of channels required at a time, where each channel delivers one stream of video object to serve one or more client requests. For convenience, M is used to represent the number of channels as a constant if the system resource is fixed, whereas m is used as a variable to measure the number of required channels for a given arrival rate and a video repository.

Next, consider a VoD delivery system as a *baseline* system to analyze the resource requirements. VoD serves each client request individually. That is, a video stream, occupying one channel for its service duration, is issued for every client request. Since all video objects have the same length L as assumed, there are $a \cdot L$ requests arrived during the time period of a video length L and the number of video streams issued is $a \cdot L$. To meet requirements of perfect service capacity with $\phi = 0$, the number of channels required is expressed as,

$$m_{\text{baseline}} = a \cdot L \quad (2)$$

Here, each channel in the system can serve only one arrived request for its duration L . With broadcast or multicast, the requests arriving one after another can be combined and served together. As a consequence, the system resource requirement can be reduced.

With a given service capacity, system efficiency can be measured by the number of channels required relative to the baseline case. Thus, a metric, Ψ_x , is defined to be the *reduction ratio* of scheme x :

$$\Psi_x = \frac{m_{\text{baseline}}}{m_x} = \frac{a \cdot L}{m_x} \quad (3)$$

where m_x is the number of channels required to meet the given perfect service capacity by using scheme x .

Repository diversity

Given the available system resources, the achievable service capacity not only depends on the methodology used to provide the delivery service, but also varies according to the diversity of the video repository. Intuitively, a repository with a few dozens of videos can provide a much higher service capacity, compared to another repository with thousands of videos. This is because the requests for a large video repository are less possibly combined. Even with the same repository size, how the client requests are diversified on different video objects can make a great impact on the service capacity.

What to analyze

To be able to evaluate the capacity and efficiency of a video delivery system, we need to analyze different methodologies. In the following sections, we consider three methods to provide video delivery services:

- Batching
- Patching
- SVD

Analysis of service capacity can help providers to plan and design the video delivery system based on the requirements of clients and availability of resources. Furthermore, analysis can also provide better understanding for performance improvement by investigating novel approaches for a video delivery system. More specifically, questions to be addressed include:

- Given a video repository, to satisfy a certain amount of requests, how much resources are required?
- Given a fixed amount of resources available, with certain content attributes, how many requests can be served?
- Given an available amount of resources with a certain amount of requests, how large of a video repository can be supported?

5. Capacity Analysis of Batching

Batching combines the requests arrived in some time period T , such as 15 minutes, and serves them together with a single channel. It is a simple method, but the maximum waiting time is T and the average waiting time is $T/2$. More specifically, for each individual video object O_j , the batching time is T_j and the arrival rate is λ_j as shown in Equation (1). When $\lambda_j T_j \leq 1$, no request can be combined for O_j ; whereas if $\lambda_j T_j > 1$ requests can be combined. As seen from Equation (1), λ_j is monotonically decreasing with j . If $T_j = T$ for all video objects, $\lambda_j T_j$ is monotonically decreasing with j as well. Here, we are discussing the average case for λ_j . The entire video repository can be divided into two sets by v . The first set, named as S_1 , is from video object O_1 to video object O_v where requests can be combined. The other set, named as S_2 , is from video object O_{v+1} to video object O_N where no request can be combined. The value of v for Batching can be obtained by

$$v = \text{Max}(j) \mid \lambda_j T > 1.$$

Let $\lambda_v T = 1$, from Equation (1),

$$\lambda_v T = \frac{C}{v} \cdot a \cdot T = 1, \quad \text{and} \quad v = C \cdot a \cdot T.$$

Since $v \leq N$,

$$v = \text{Min}(C \cdot a \cdot T, N) \tag{4}$$

In order to understand the importance of v , we define a parameter ρ to measure what fraction of the requests belongs to the first set of videos, S_1 :

$$\rho = \frac{\sum_{j=1}^v \lambda_j}{\sum_{j=1}^N \lambda_j} = \frac{\sum_{j=1}^v \frac{1}{j}}{\sum_{j=1}^N \frac{1}{j}} = \sum_{j=1}^v \frac{C}{j}. \quad (5)$$

Let us find out requirements on the number of channels for the two sets of videos:

- S_1 . For every video object $O_j \in S_1$ during batching time T_j , we can combine $\lambda_j T_j$ requests since $\lambda_j T_j > 1$. Therefore, for O_j , at most L/T_j channels are required. The total number of channels required to serve video set S_1 is,

$$m_1 = \sum_{j=1}^v \frac{L}{T_j}$$

Notice that m_1 is independent of the arrival rate.

- S_2 . For every video object $O_j \in S_2$, since $\lambda_j T_j \leq 1$, during batching time T_j , no request can be combined. Therefore, the total number of channels required to serve video set S_2 is equal to the number of requests which belong to S_2

$$m_2 = (1 - \rho) \cdot a \cdot L$$

From calculation for S_1 and S_2 , assuming $T_j = T$, the number of channels required for arrival rate a is

$$m_{\text{batch}} = m_1 + m_2 = \sum_{j=1}^v \frac{L}{T_j} + (1 - \rho) \cdot a \cdot L = v \cdot \frac{L}{T} + (1 - \rho) \cdot a \cdot L. \quad (6)$$

The system resources required

With Batching, the system resources are measured by the number of channels required, m_{batch} , in order to meet a certain perfect service capacity with the given video repository. Besides, m_{batch} is compared to m_{baseline} to illustrate how much resource reduction can be reached with the Batching strategy instead of the simple VoD. Analysis can be addressed based on the value of ρ .

Case I: $\rho < 1$ From Equation (6)

$$m_{\text{batch}} = C \cdot a \cdot T \cdot \frac{L}{T} + (1 - \rho) \cdot a \cdot L = (C + 1 - \rho) \cdot a \cdot L > C \cdot a \cdot L. \quad (7)$$

The reduction ratio is

$$\Psi_{\text{batch}} = \frac{m_{\text{baseline}}}{m_{\text{batch}}} = \frac{a \cdot L}{(C + 1 - \rho) \cdot a \cdot L} = \frac{1}{C + 1 - \rho} < \frac{1}{C}.$$

Here $C \leq 1$ and also $C \leq \rho$ from Equation (5). Thus, $\Psi_{\text{batch}} \geq 1$. On the other hand, when $N < 10,000$, C is larger than 0.1 and Ψ_{batch} is less than 10, which implies that the improvement over the simple VoD is limited.

Case II: $\rho = 1$ From Equation (6)

$$m_{\text{batch}} = m_1 = N \frac{L}{T}.$$

The number of channels is independent of the number of requests. The reduction ratio is

$$\Psi_{\text{batch}} = \frac{a \cdot L}{\frac{N \cdot L}{T}} = \frac{a \cdot T}{N}.$$

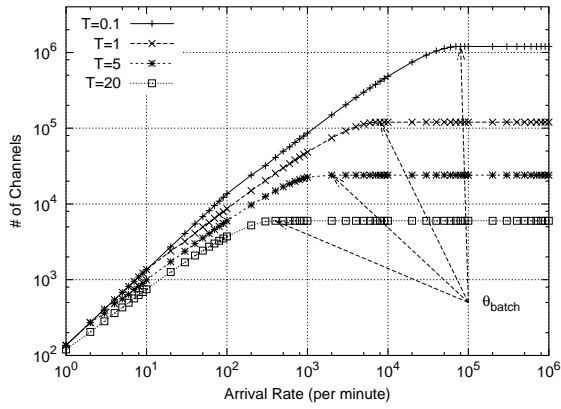
Here, Ψ_{batch} increases with a . That is, provided the system resource required could be met, the service capacity may increase with the request arrival rate.

Comparing the above two cases, Case II exhibits more efficient resource utilization, since a single channel usually serves more than one client request. In fact, this scenario is similar to Near VoD, where each video object is repeatedly streamed every T minutes and requires L/T channels in total no matter how many client requests are served. However, in order to reach Case II, $C \cdot a \cdot T \geq N$. That is, either the arrival rate is high enough; or the video repository is small, which is usually true in many commercial Near VoD services.

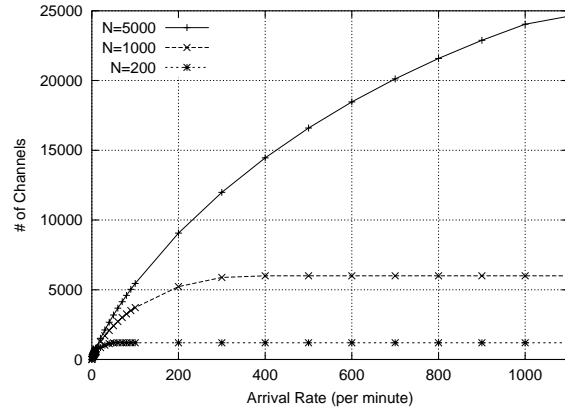
When setting the default values of $N = 1,000$, $T = 20$, $L = 120$, and $\alpha = 1$, Figure 1 shows the number of channels required with Batching for different arrival rates a , where the impacts from various parameters are given in four subfigures, respectively. Obviously, the longer the batching time, the less m_{batch} required as shown in Figure 1(a). For a short batching time such as six seconds ($T = 0.1$), there is virtually no waiting time. Therefore, it is equivalent to the true VoD so that its scalability is poor. As many as 1,200,000 channels may be required! Longer batching time results in better performance. For example, when $T = 20$ minutes, the Batching system will not require more resources when the arrival rate reaches and goes beyond 375 per minute. The value of batching time T is critical for the system performance since when $C \cdot a \cdot T$ is larger than N , the resource requirement is bounded by $N \frac{L}{T}$. On the other hand, the longer the T , the longer users must wait. Moreover, the threshold is defined as the arrival rate such that ρ just reaches 1,

$$\theta_{\text{batch}} = a \mid \text{such that } C \cdot a \cdot T = N. \quad (8)$$

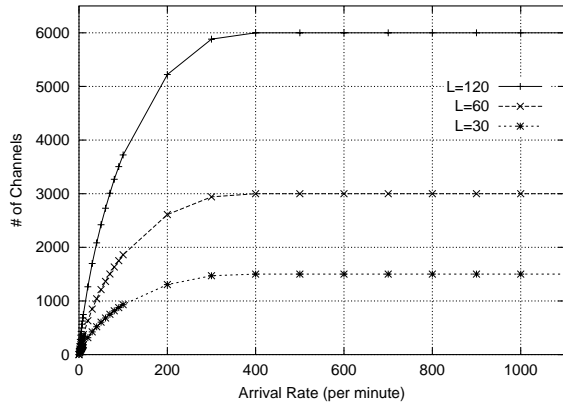
Shown also in Figure 1(a) are the threshold values for different T values. The number of channels required is bounded once the arrival rate a increases across threshold θ_{batch} . Figure 1(b) shows the performance of $N = 200, 1,000$, and $5,000$ for Batching. A system with a repository of 5000 videos requires a large amount of resources even for a moderate arrival rate, such as $a = 400$. On the other hand, only 1,200



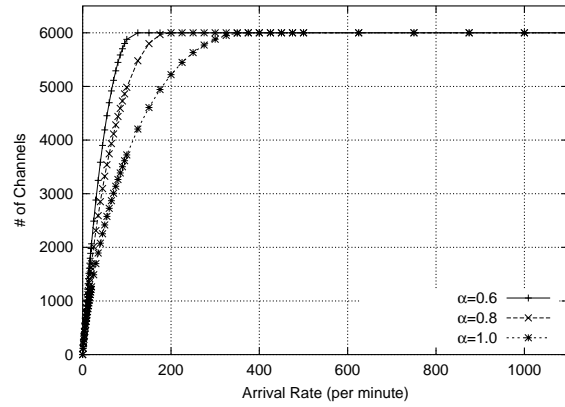
(a) m_{batch} with $N = 1000$, $L = 120$, and $\alpha = 1$



(b) m_{batch} with $T = 20$, $L = 120$, and $\alpha = 1$



(c) m_{batch} with $N = 1000$, $T = 20$, and $\alpha = 1$



(d) m_{batch} with $N = 1000$, $T = 20$, and $L = 120$

Figure 1: The number of channels required with Batching for different arrival rate a .

channels are required for any arrival rate when $N = 200$. Figure 1(c) shows the resource requirement for $L = 30, 60, \text{ and } 120$ minutes. The value of $L = 120$ minutes is the typical length of a movie. Less system resources are required for videos of shorter lengths, so providing a repository of short video-clips demands less system resources compared to a repository with typical movies. When α is not equal to 1, the value of ν and ρ should be changed to:

$$\nu = \text{Min}((C \cdot a \cdot T)^{\frac{1}{\alpha}}, N) \quad \text{and} \quad \rho = \sum_{j=1}^{\nu} \frac{C}{j^{\alpha}}.$$

The different values of α also have an impact on the resource requirement as shown in Figure 1(d). In general, the larger the α value, the better the performance, since more requests can be combined. For a smaller value of α , Batching requires more channels when the arrival rate is low.

A service provider may only have a limited number of channels for video delivery service. The provider needs to know for a given video repository what an arrival rate can be supported by the system with a certain rejection rate. Or given an arrival rate, how many videos can be in the repository? Two scenarios are addressed next, assuming the system resource is fixed in terms of M .

The service capacity provided

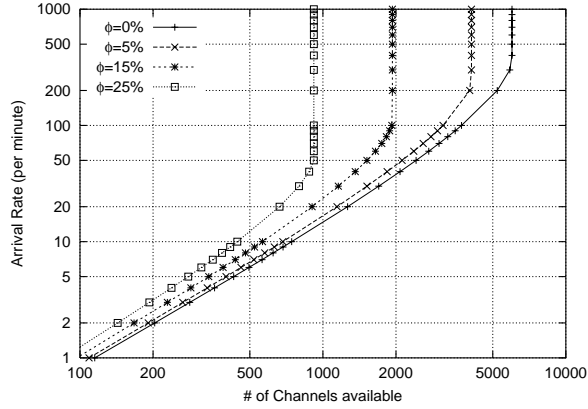
Assume the batching time T is fixed, the system resources are limited and the video repository has been established, parameters N , the video length L and α are fixed. In Figure 2(a), as the system resource M increases, the curve with $\phi = 0\%$ represents the maximum perfect service capacity, measured by arrival rate a to be served. If the arrival rate is high, the rejection rate ϕ cannot be zero any more, and from Equation (7), it can be found what is the real service capacity assuming a popularity-based rejection. That is, the system tends to reject requests on least popular video objects, since such requests are more likely occupy a single channel resource. Service capacity of $a_E = a \cdot (1 - \phi)$ is shown in Figure 2(a) for different rejection rates, $\phi = 5\%, 15\%, \text{ and } 25\%$.

The video repository supplied

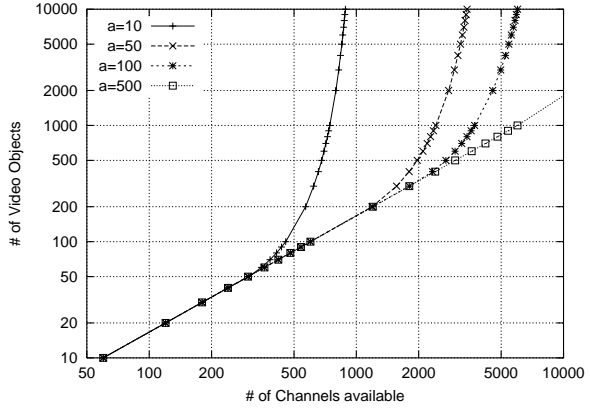
With the given system resources and a fixed arrival rate, what is the maximum number of videos in the repository that can be served? In this scenario, parameters a and ϕ are both fixed, as well as the batching time T and the video length L . Figure 2(b) shows what value of N can be set according to different arrival rates, when the available system resource M changes.

6. Capacity Analysis of Patching

Patching combines requests with patching streams. When a request misses the first part of a previous stream, it shares the rest of the stream and the server issues a patching stream to make up the request.



(a) a with $N = 1000, L = 120, T = 20$ and $\alpha = 1$



(b) N with $\phi = 0\%, L = 120, T = 20$ and $\alpha = 1$

Figure 2: The service capacity and video repository with Batching.

Patching is better than Batching since it satisfies requests immediately. However, it is more complex and many versions of Patching have been studied [19, 2, 11, 16]. The most efficient Patching method is the recursive Patching where the patching streams are merged recursively by “Patching the Patching stream” to minimize the bandwidth requirement [20, 25]. A common assumption for Patching is the receive-two model where a client can receive at most two streams at any time. Although different versions of Patching require a different number of channels, the bandwidth requirement can be roughly modeled as $(\ln(\lambda_j L) + 1)$ channels for video object O_j [20, 25].

With this model, the number of channels required can be obtained as follows. When $\lambda_j L \leq 1$, no request can be combined. Although many algorithms do not attempt merging with an existing stream that is already at least half over, it is possible that some requests can be combined partially when $\lambda_j L > 1$. Here again, we are discussing only the average case for λ_j . We divide the entire video repository into two sets according to $\lambda_j L \leq 1$ and $\lambda_j L > 1$. Thus, v can be obtained as:

$$\lambda_v L = \frac{C}{v} \cdot a \cdot L = 1, \text{ and } v = C \cdot a \cdot L.$$

Also, because the upper limit of v is N , we have

$$v = \text{Min}(C \cdot a \cdot L, N). \quad (9)$$

The number of channels required can be expressed as follow:

$$m_{\text{patch}} = \sum_{j=1}^v (\ln(\lambda_j L) + 1) + (1 - \rho)a \cdot L$$

$$\begin{aligned}
&= \sum_{j=1}^v (\ln(\frac{C}{j} \cdot a \cdot L) + 1) + (1 - \rho)a \cdot L \\
&= v + \ln((C \cdot a \cdot L)^v \cdot \prod_{j=1}^v \frac{1}{j}) + (1 - \rho)a \cdot L \\
&= v + \ln \frac{(C \cdot a \cdot L)^v}{v!} + (1 - \rho)a \cdot L,
\end{aligned}$$

where ρ is from Equation (5).

The system resources required

With Patching, in order to meet a certain perfect service capacity with the given video repository, the system resources are measured by the number of channels required, m_{patch} .

Case I: $\rho < 1$

$$m_{\text{patch}} = C \cdot a \cdot L + \ln \frac{(C \cdot a \cdot L)^{C \cdot a \cdot L}}{(C \cdot a \cdot L)!} + (1 - \rho)a \cdot L > C \cdot a \cdot L. \quad (10)$$

Thus, the reduction ratio Ψ_{patch} is less than $\frac{1}{C}$.

Case II: $\rho = 1$

$$m_{\text{patch}} = N + \ln \frac{(C \cdot a \cdot L)^N}{N!}. \quad (11)$$

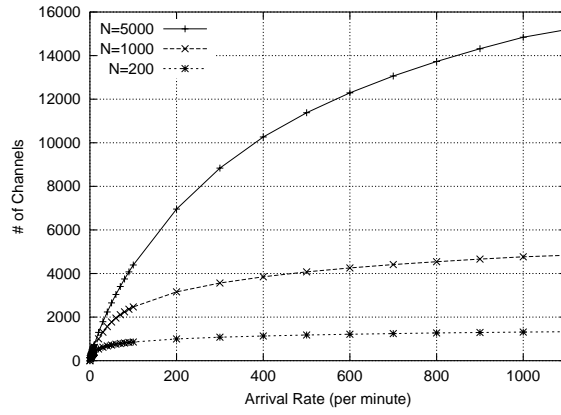
The number of channels required increases with $N \ln a$.

Similar to Batching, the default values of $N = 1,000$, $L = 120$, and $\alpha = 1$ are set. The threshold is also defined as the arrival rate such that ρ just reaches 1,

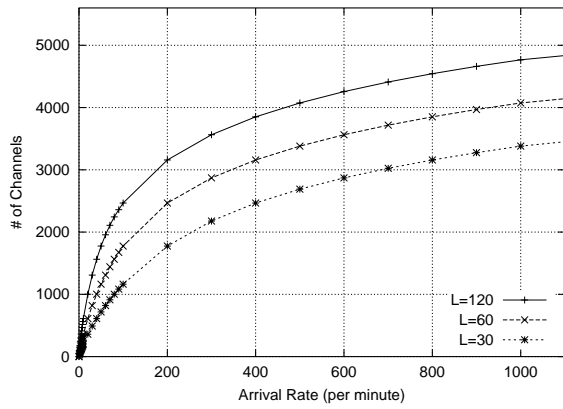
$$\theta_{\text{patch}} = a \mid \text{such that } C \cdot a \cdot L = N. \quad (12)$$

Here, θ_{patch} depends on the video length L instead. Figure 3(a) shows the performance of $N = 200$, 1,000, and 5,000 for Patching. When the arrival rate a increases across threshold θ_{patch} , the number of channels required still slowly increases with a . Patching can be combined with Batching and the number of channels required will eventually be independent of the number of requests. Figure 3(b) shows the resource requirement for $L = 30$, 60, and 120 minutes. The value of $L = 120$ minutes is the typical length of a movie. Patching behaves differently from Batching, its resource requirement for short videos is not reduced as significantly as one in Batching, since a short length results in less chances to patch. When α is not equal to 1, the value of v and ρ should be changed to:

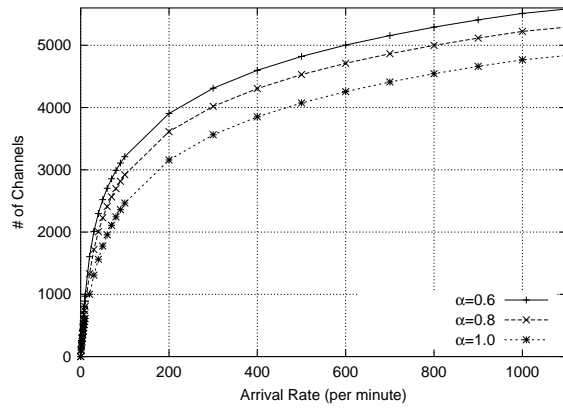
$$v = \text{Min}((C \cdot a \cdot L)^{\frac{1}{\alpha}}, N) \quad \text{and} \quad \rho = \sum_{j=1}^v \frac{C}{j^\alpha}. \quad (13)$$



(a) m_{patch} with $L = 120$, and $\alpha = 1$



(b) m_{patch} with $N = 1000$, and $\alpha = 1$



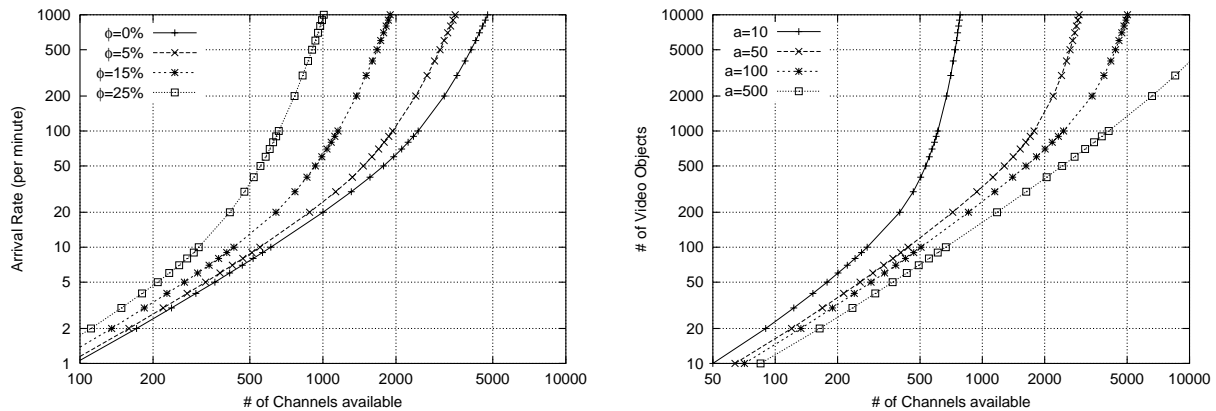
(c) m_{patch} with $N = 1000$, and $L = 120$

Figure 3: The number of channels required with Patching for different arrival rate a .

The different values of α also have an impact on the resource requirement as shown in Figure 3(c).

The service capacity provided and the video repository supplied

Similar to Batching, Figure 4(a) illustrates variations of arrival rate a when different rejection rates are allowed, $\phi = 0\%$, 5% , 15% , and 25% , as the available system resource M changes. Figure 4(b) shows what value of N can be supported according to different arrival rates.



(a) a with $N = 1000$, $L = 120$, $T = 20$ and $\alpha = 1$

(b) N with $\phi = 0\%$, $L = 120$, $T = 20$ and $\alpha = 1$

Figure 4: The service capacity and video repository with Patching.

7. Capacity Analysis of SVD

SVD has been proposed in [3]. This paradigm utilizes the property that not all contents are needed at the request time. In many situations, people can plan ahead to obtain some content before it is actually used. In the SVD paradigm, users submit requests with specification of start time. A pricing scheme ensures that the user-specified start time reflects users' real needs. The SVD system combines requests to form multicasting groups and schedules these groups to meet the deadline. With this paradigm, requests can be combined to reduce the server load and network traffic. Furthermore, the traffic can be smoothed by shifting the peak-time traffic to a non-peak time. SVD extends a user's option from click-wait-see patterns to plan-ahead alternatives, being able to submit requests with timing specification. SVD scheduling has a different objective from many existing scheduling schemes. It does not aim at minimizing the waiting time. Instead, it focuses on meeting deadlines and at the same time combining requests to form multicasting groups. The SVD paradigm behaves similarly as Batching since the average of the deadline is equivalent to a certain batching time. What is different from Batching is that the equivalent batching time decreases with the arrival rate a . We assume that the latest start time (deadline) of a request is uniformly distributed

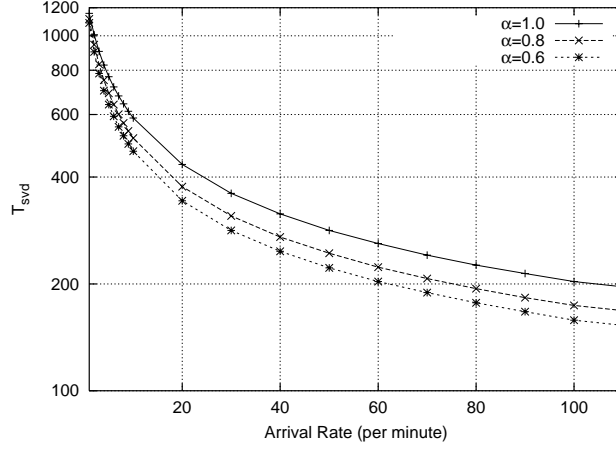


Figure 5: Relation between T_{svd} and arrival rate a .

between time 1 and time T_d . The requests arrive at different time with different deadlines. If no deadline falls between time t_0 and time $t_0 + t_t$, no channel needs to be issued. The longest t_t is the equivalent batching time T which was derived in [26]. Results show the average distance or the equivalent batching time T_j for video object O_j as below,

$$T_j = \sum_{k=1}^{T_d} \left(k \times \frac{T_d^{\lambda_j} - (T_d - k)^{\lambda_j}}{T_d^{\lambda_j k}} \prod_{i=1}^{k-1} (T_d - i)^{\lambda_j} \right), \quad (14)$$

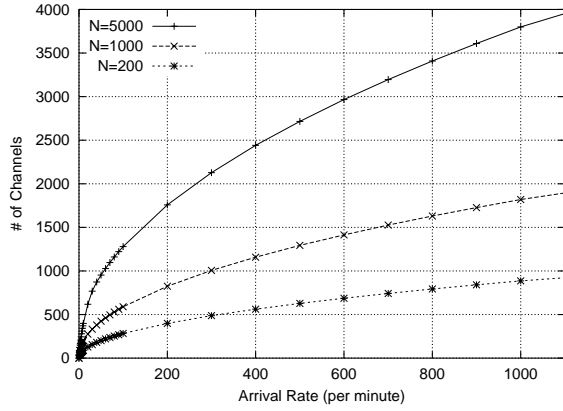
where λ_j is the arrival rate for video object O_j .

The system resources required

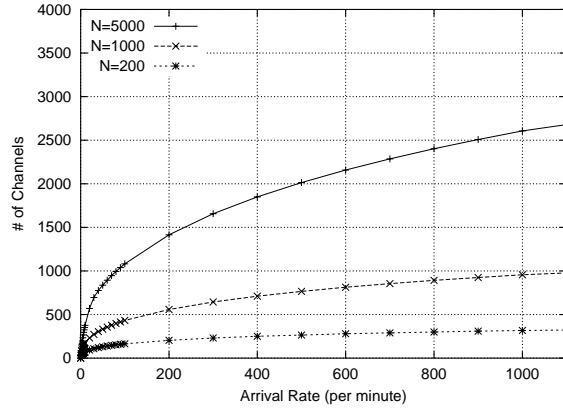
Equations (6) and (7) can be applied to SVD as well. In SVD, T_j is the equivalent batching time for video object O_j obtained from Equation (14). The average equivalent batching time for all video objects is computed as follows:

$$T_{svd} = \frac{v}{\sum_{j=1}^v \frac{1}{T_j}} \quad \text{and} \quad m_{svd} = m_{\text{batch}} \Big|_{T=T_{svd}}. \quad (15)$$

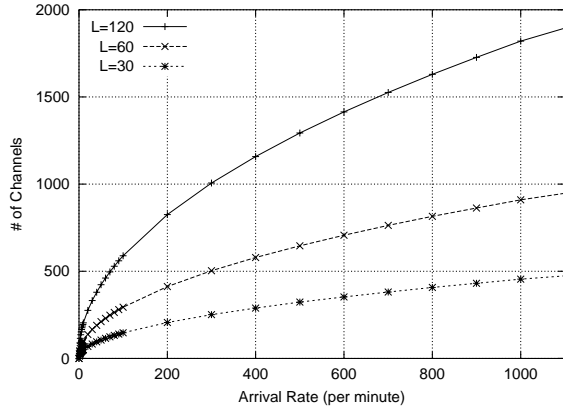
The value of T_{svd} depends on the arrival rate. Figure 5 shows the relation between T_{svd} and arrival rate a , where $N = 1,000$, $L = 120$ minutes, and $T_d = 1,440$ minutes. The T_d/L ratio is 12. The α value is set to be 1.0, 0.8, and 0.6, respectively. The larger the T_{svd} value, the smaller the number of channels. The upper bound of m can be expressed as $m \leq N \times L/T_{svd}$. When $a = 1$, the T_{svd}/L ratio can be as large as 9, and less than 111 channels are required. When $a = 100$, the T_{svd}/L ratio is reduced to about 1.5, and about 600 channels are required.



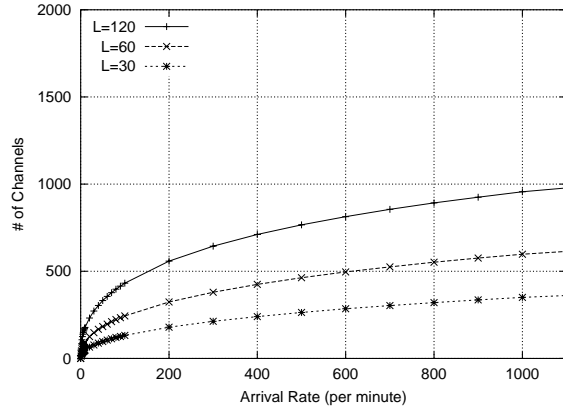
(a) m_{SVD} with $L = 120$, and $\alpha = 1$



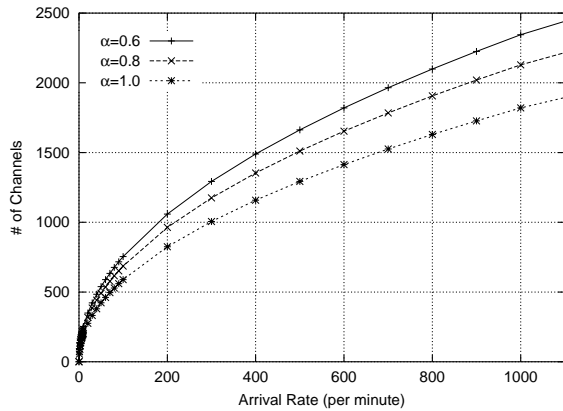
(b) m_{SVDP} with $L = 120$, and $\alpha = 1$



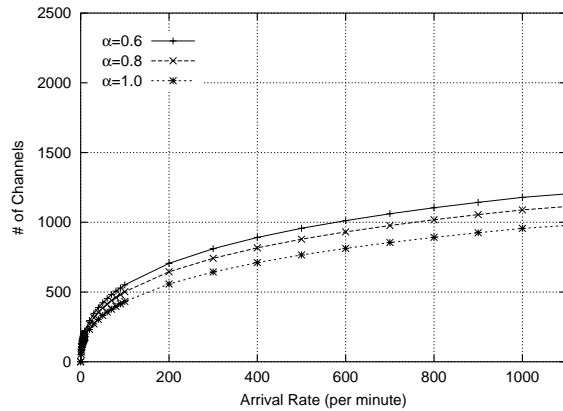
(c) m_{SVD} with $N = 1000$ and $\alpha = 1$



(d) m_{SVDP} with $N = 1000$ and $\alpha = 1$



(e) m_{SVD} with $N = 1000$ and $L = 120$



(f) m_{SVDP} with $N = 1000$ and $L = 120$

Figure 6: The number of channels required with SVD and SVDP for different arrival rate a .

Finally, performance of SVD can be further improved by combining SVD with Patching (SVDP). For a video object whose number is between 1 and v , all requests are batched into L/T_j channels, which are then patched into $\ln(L/T_j) + 1$ channels. When $v < C \cdot a \cdot L$, for a video object whose number is between $v + 1$ to $C \cdot a \cdot L$, no request is batched so the Patching formula is applied. Thus, the formula for SVDP becomes:

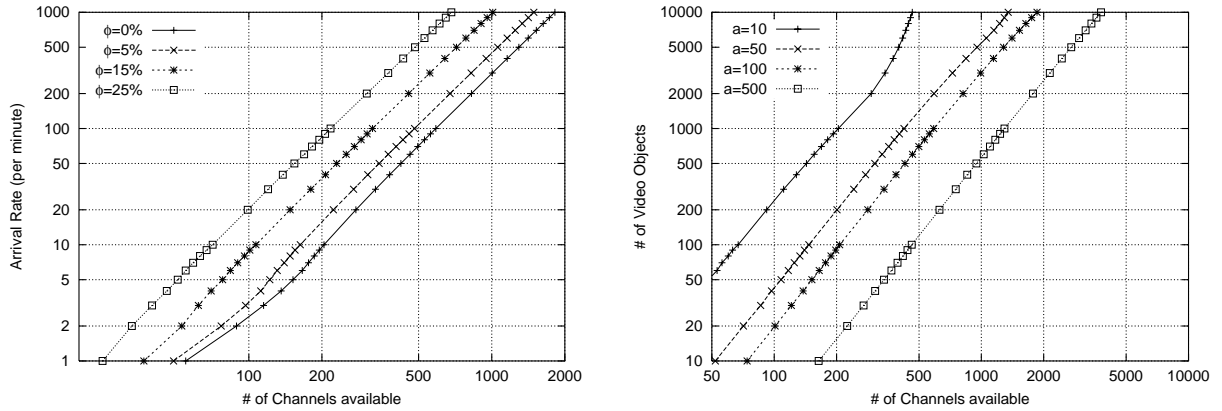
$$m_{\text{svdp}} = \sum_{j=1}^v (\ln \frac{L}{T_j} + 1) + \sum_{j=v+1}^{CaL} (\ln(\frac{C}{j}aL + 1)) + (1 - \rho)aL. \quad (16)$$

SVD can provide long equivalent batching time T when a is small. Better than Batching, SVD can provide immediate response though it encourages users plan ahead. Its drawback is the same as Patching, that is, after the arrival rate reaches the threshold, the number of channels required still slowly increases. Here, the threshold is again defined as the arrival rate so that ρ just reaches 1,

$$\theta_{\text{svd}} = a \mid \text{such that } C \cdot a \cdot T_{\text{svd}} = N. \quad (17)$$

When setting the default values of $N = 1,000$, $L = 120$, and $\alpha = 1$, Figure 6 shows the number of channels required with SVD and SVDP for different arrival rates, where the impacts from various parameters are given in six subfigures, respectively. SVDP performs well. Less than 1,000 channels are needed for a video repository of 1,000 videos.

The service capacity provided and the video repository supplied



(a) a for SVD with $N = 1000$, $L = 120$, and $\alpha = 1$

(b) N for SVD with $\phi = 0\%$, $L = 120$, and $\alpha = 1$

Figure 7: The service capacity and video repository with SVD.

Similar to Batching and Patching, Figure 7(a) illustrates a variation of arrival rate a when different

rejection rates are allowed, $\phi = 0\%$, 5% , 15% , and 25% , as the available system resource M changes. Figure 7(b) shows what value of N can be supported according to different arrival rates.

8. Comparison and Conclusion

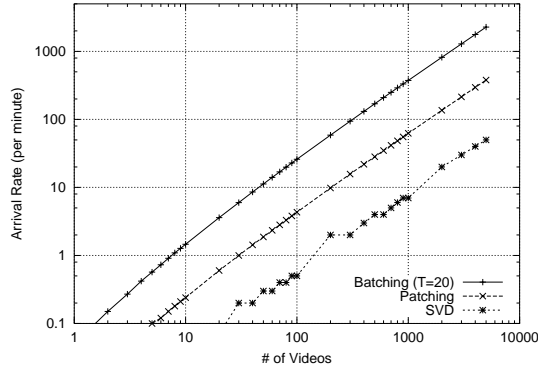
Analysis results are compared among Batching, Patching, SVD, and SVDP. Unless mentioned otherwise, $N = 1,000$. The video length $L = 120$ minutes and $T_d = 1,440$ minutes. The α value is set to be 1.

Figure 8(a) shows the relation between the threshold and the number of videos for Batching, Patching, and SVD. The threshold is expressed by the a value where the system resources become bounded, therefore, do not increase or slowly increase with a . It can be seen that the threshold of Batching is higher than that of Patching, which is higher than that of SVD.

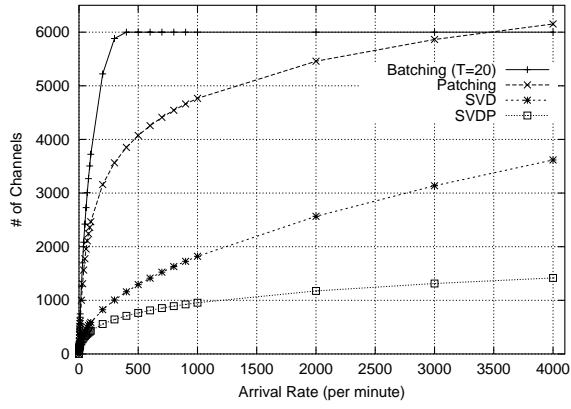
We now compare the performance of different methods in Figures 8(b) and (c). Figure 8(c) shows the details of the behavior of these methods when arrival rate a is low. Here, the batching time T is set to be 20 minutes. For Batching, the number of channels remains 6,000 when $a > 375$ per minute. The threshold of Patching is $a = 63$ and that of SVD is only $a = 7$. However, after the threshold the number of channels slowly increases. SVDP shows a much better performance. Its threshold is the same as SVD without Patching but the number of channels increases much slower. The corresponding reduction ratio Ψ_X is shown in Figures 8(d) and (e). For Patching, when the arrival rate reaches 63 per minute, Ψ_X is 16 and 480 channels are required. When the arrival rate reaches 1,000 per minute, Ψ_X is 25 and 4,766 channels are required. For SVDP, when the arrival rate is 7 per minute, only 157 channels are required. And when the arrival rate reaches 1000 per minute, Ψ_X is 125 and 956 channels are required. Simulation has been conducted to verify the analysis results [3]. The difference between analysis and simulation is within 10%.

Consider the video repository in each scheme. Assume we have 1,000 channels in a HFC system with a large request arrival rate, say 500. Batching is able to handle 166 2-hour videos with $T = 20$ minutes. Patching can handle 250 videos. SVD, however, is able to have 1,000 videos in its repository. With more channels and/or a lower arrival rate, a larger repository can be served. Generally speaking, Batching and Patching are more suitable for a small video repository and a large audience; VoD is better for a large repository and diverse clients; and SVD is in between.

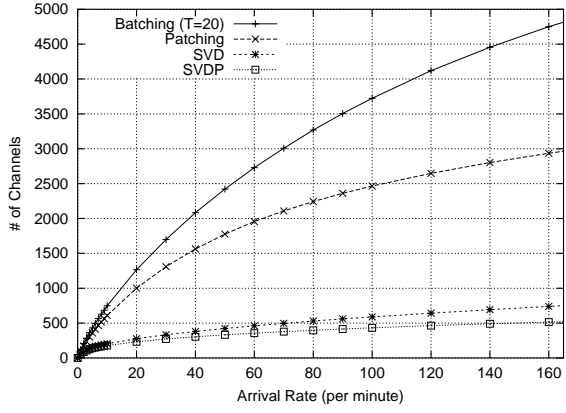
Capacity analysis of the closed-loop on-demand video service has been studied in this article. Performance models of Batching, Patching, and SVD are given. Analysis based on these models shows that the threshold depends on the size of video repository, the video length or batching time, and the arrival rate. The major conclusion is that the current methods are not scalable to a large repository of videos. This



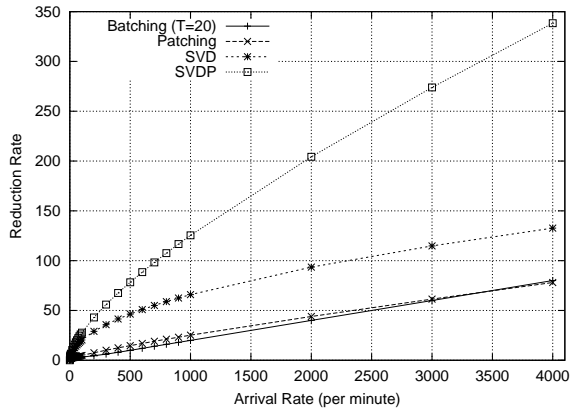
(a) $\theta_{\text{batch}}, \theta_{\text{patch}}, \theta_{\text{svd}}$ with $L = 120$, and $\alpha = 1$



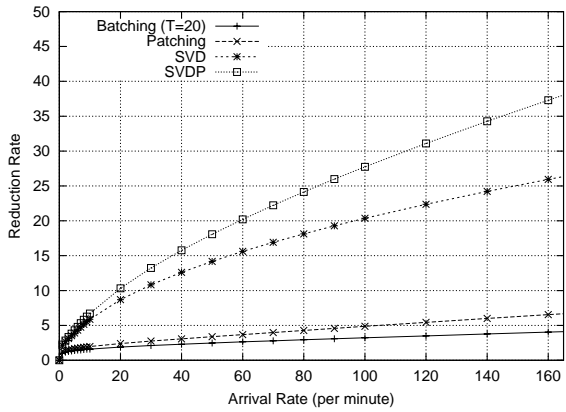
(b) $m_{\text{batch}}, m_{\text{patch}}, m_{\text{svd}}, m_{\text{svdp}}$, with $L = 120, \alpha = 1$



(c) Subregion of m for small a , with $L = 120, \alpha = 1$



(d) $\Psi_{\text{batch}}, \Psi_{\text{patch}}, \Psi_{\text{svd}}, \Psi_{\text{svdp}}$, with $L = 120, \alpha = 1$



(e) Subregion of Ψ for small a , with $L = 120, \alpha = 1$

Figure 8: Comparison of different video delivery strategies.

analysis provides insights for various video delivery techniques. Content and service providers are able to compute the system resource requirement, maximum number of videos that can be in a repository, and the maximum number of clients that can be served under a certain condition. They also can select from various video delivery techniques that is the best for their goals.

Acknowledgments

The authors would like to thank the anonymous reviewers for their thorough comments.

References

- [1] A. Dan, D. Sitaram, and P. Shahabuddin, "Scheduling policies for an on-demand video server with batching," in *ACM Multimedia*, pp. 15–23, 1994.
- [2] K. A. Hua, Y. Cai, and S. Sheu, "Patching: A multicast technique for true video-on-demand services," in *6th ACM Int'l. Multimedia Conf. (ACM Multimedia '98)*, pp. 191–200, Sept. 1998.
- [3] M. Wu, S. Ma, and W. Shu, "Scheduled video delivery for scalable on-demand service," in *International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSS-DAV)*, May 2002.
- [4] S. Viswanathan and T. Imielinski, "Metropolitan area video-on-demand service using pyramid broadcasting," *Multimedia Systems*, vol. 4, no. 4, pp. 197–208, 1996.
- [5] C. C. Aggarwal, J. L. Wolf, and P. S. Yu, "A permutation-based pyramid broadcasting scheme for video-on-demand systems," in *International Conference on Multimedia Computing and Systems*, pp. 118–126, 1996.
- [6] K. A. Hua and S. Sheu, "Skyscraper broadcasting: A new broadcasting scheme for metropolitan video-on-demand systems," in *SIGCOMM*, pp. 89–100, 1997.
- [7] Y. Birk and R. Mondri, "Tailored transmissions for efficient near video-on-demand service," in *IEEE International Conference on Multimedia Computing and Systems*, 1999.
- [8] C. Aggarwal, J. Wolf, and P. Yu, "On optimal batching policies for video-on-demand storage servers," in *IEEE International Conference on Multimedia Computing and Systems*, June 1996.
- [9] A. Dan, D. Sitaram, and P. Shahabuddin, "Dynamic batching policies for an on-demand video server," *ACM Multimedia Systems*, no. 4, pp. 112–121, 1996.
- [10] A. Dan, Y. Heights, and D. Sitaram, "Generalized interval caching policy for mixed interactive and long video workloads," in *SPIE's Conf. on Multimedia Computing and Networking*, pp. 344–351, Jan. 1996.
- [11] L. Gao and D. F. Towsley, "Supplying instantaneous video-on-demand services using controlled multicast," in *ICMCS, Vol. 2*, pp. 117–121, 1999.
- [12] L. Golubchik, J. C. S. Lui, and R. R. Muntz, "Reducing i/o demand in video-on-demand storage servers," in *Measurement and Modeling of Computer Systems*, pp. 25–36, 1995.
- [13] R. Steinmetz and K. Nahrstedt, *Multimedia: Computing, Communications, and Applications*. Prentice Hall, 1995.

- [14] S. Sheu, K. A. Hua, and T. H. Hu, "Virtual batching: A new scheduling technique for video-on-demand servers," in *the 5th DASFAA*, Apr. 1997.
- [15] K. C. Almeroth and M. H. Ammar, "The use of multicast delivery to provide a scalable and interactive video-on-demand service," *IEEE Journal of Selected Areas in Communications*, vol. 14, no. 6, pp. 1110–1122, 1996.
- [16] S. Sen, L. Gao, J. Rexford, and D. Towsley. Optimal Patching Schemes for Efficient Multimedia Streaming, Proc. 9th Int'l Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'99), Basking Ridge, NJ, June 1999.
- [17] L. Gao, Z.-L. Zhang, and D. Towsley, "Catching and selective catching: Efficient latency reduction techniques for delivering continuous multimedia streams," in *7th ACM Int'l. Multimedia Conf. (ACM Multimedia '99)*, pp. 203–206, 1999.
- [18] S. Carter and D. Long. Improving video-on-demand server efficiency through stream tapping. In *ICCCN 97*, pages 200–7, Las Vegas, NV, USA. IEEE Computer Society Press, 1997.
- [19] D. L. Eager, M. K. Vernon, and J. Zahorjan, "Optimal and efficient merging schedules for video-on-demand servers," in *7th ACM Int'l. Multimedia Conf. (ACM Multimedia '99)*, pp. 199–202, Nov. 1999.
- [20] J. E.G. Coffman, P. Jelenkovic, and P. Momcilovic, "Provably efficient stream merging," in *Sixth International Workshop on Web Caching and Content Distribution (WCW'01)*, Mar. 2001.
- [21] G. Zipf, *Human Behavior and the Principle of Least Effort*. Reading, MA: Addison-Wesley, 1949.
- [22] S. Glassman, "A caching relay for the world wide web," in *The 1st International Conference of World Wide Web*, 1994.
- [23] C. Cunha, A. Bestavros, and M. Crovella, "Characteristics of WWW client-based traces," Tech. Rep. TR-95-010, Boston University Department of Computer Science, 1995.
- [24] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker, "Web caching and zipf-like distributions: Evidence and implications," in *IEEE Infocom*, 1999.
- [25] A. Bar-Noy and R. Ladner, "Competitive on-line stream merging algorithms for media-on-demand," in *SODA01*, 2001.
- [26] W. Shu and M. Wu, "Scalability of closed-loop video delivery service," in *the IEEE International Conference on Multimedia and Expo (ICME2003)*, July 2003.