Time Domain Modeling of Batching under User Interaction and Dynamic Adaptive Piggybacking Schemes[1]

W. Ke, P. Basu and T.D.C.Little

Department of Electrical and Computer Engineering
Boston University, Boston, Massachusetts 02215, USA
(617) 353-9877
{*ke,pbasu,tdcl*}*@bu.edu*

MCL Technical Report No. 09-30-2000

Abstract– Provision of Video-on-Demand services requires sustained periods of high bandwidth network and server capacity. Aggregation schemes can be used to increase the supported customer population under the constraints of this resource. To this end, a number of aggregation algorithms have been proposed. However, the approaches are difficult to test in the large scale, and model due to system behavioral complexity. This situation is exacerbated by the modeling of customer interactions. Therefore, we explore a new technique for modeling such scenarios by using time domain analysis. By this we refer to the process of estimation of system characteristics such as the average number of channels used, average number of customers, etc., as a function of time.

In this paper, we show that time domain modeling of two VoD aggregation schemes is analogous to finding the time response of linear systems using the well known convolution theorem. We model two aggregation schemes using this technique: (1) Batching by time-out and (2) Adaptive Piggybacking employing Snapshot-RSMA [1, 2, 3]. We delineate the requirements of the VoD system under which this technique can be employed and show that it yields highly satisfactory estimation results. We also propose a sampling methodology which gives good estimation results when the modeling of the aggregation scheme is mathematically too complex.

Keywords: Video-on-demand, time-domain modeling, batching, adaptive piggybacking, estimation methods, system characterization

[1]*Proc. SPIE Multimedia Computing and Networking (MMCN 2002)*, San Jose, CA January 2002. This work is supported in part by the NSF under grant No. NCR-9523958. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

1 Introduction

A VoD system is typically composed of the following elements: **server**, **network** and **client**. The server is the element in which videos are stored in digital format. The network is the delivery channel and connects the server to multiple clients. The transmission capacity of the network is limited and logically divided into *channels*. Each channel is capable of transmitting video data at a rate sufficient for VoD purposes, for instance, 4 Mbps for MPEG-2 streams. In this study we are considering data networks that support both unicast and multicast delivery.

A *stream* indicates a sequential transmission of data in time through the network. A *video* (or movie) is composed of multiple frames which must be displayed at a pre-specified rate. Thus one characteristic of a video is its *length* (L), which in this paper is defined as the time taken to display the video at its normal display rate (or normal content progression rate). When a client makes a request to the video server, the server responds by allocating an available channel and initiating a stream. The data is thus "streamed", i.e., sent sequentially in time through the network from the server to the client. Since the delivery is not instantaneous, we will use the term *stream position* as a reference to the part of the video that is being shown in respect to the video's length. Thus if the video length is characterized by L units of time, a stream position is a real number p, $0 \leq p \leq L$.

In our model, the maximum channel capacity at the video server, measured in terms of the number of channels available, cannot change dynamically, and a channel can be used either to multicast or to unicast. Multicasting supports multiple clients while unicasting supports only one, but the amount of bandwidth each occupies at the server is assumed to be the same in this study. Therefore it is advantageous for the server to serve as many clients as possible through a single multicast channel. It has been shown that the access pattern to a video database (e.g., movie rental stores) is skewed, so that often a large percentage of the population actually accesses only a small number of videos [4]. This encourages the possibility of serving multiple clients through a single channel, as many will be requesting the same video. However, since such requests come from different clients scattered in the network, they tend to be separated in time. *Aggregation schemes* are algorithms that attempt to aggregate multiple client requests separated in time to be served by a single channel. In this paper, we will study two aggregation schemes: *Batching* by time-out and *Adaptive Piggybacking* employing *Snapshot-RSMA* [1, 2, 3].

Batching delays the response to a client's request for video. This is done in the hope that other clients will eventually make requests to watch the same video. In this manner the video server can serve a large number of clients while incurring the transmission cost of a single channel. Obviously, this scheme imposes a non-zero waiting time on its clients. If the VoD system allows interactions, a user who is batched with others upon interaction is given a solo channel for the remainder of the movie. The variable under the control of the VoD system designer in this scheme is the *waiting* or *batching period* W.

Adaptive Piggybacking [1] (or Rate Adaptation [3]) refers to the possibility of changing the content progression rate without any significant visible effect to the client. This allows the server to aggregate users whose requests are separated by a short interval of time. These users are eventually served by a single multicast channel. Many algorithms have been proposed on how this merging can be achieved [1, 2, 3, 5]. We try to offer a time domain model of an algorithm that was originally proposed by Aggarwal et al. in [2]. This algorithm couples dynamic programming with a snapshot algorithm. Basu et al. [3] identified the nature of the stream merging problem as isomorphic to a special case of the Rectilinear Steiner Minimal Arborescence (RSMA) problem[2] [3]. Thus, in this paper, we refer to this algorithm as *Snapshot-RSMA*.

This algorithm yields optimal solution in the static scenario, as shown by Aggarwal et al. [2]. By static we refer to the scenario in which there will be no new client request arrivals and existing clients will not interact. And optimal in the sense of minimizing the cost, which was defined as the number of frames transmitted over the network [2]. To deal with dynamic arrivals of client requests, Aggarwal et al. suggested applying the dynamic programming algorithm at regular periods. At the end of one period, the VoD system would survey the stream positions of clients that arrived during that period (i.e., take a *snapshot*) and apply the dynamic programming algorithm to that set of streams. In this paper, the process of applying the dynamic algorithm is termed *recomputation*, and the interval between consecutive recomputation operations is the variable under the control of the VoD systems designer (we refer to this as the *snapshot* or *recomputation interval* R).

Batching systems are simpler to analyze but the introduction of interactions requires more complex models. Models proposed need be able to take into account client interaction processes to predict the bandwidth required to support the VoD system. Adaptive Piggybacking using Snapshot-RSMA is even harder to model analytically. The outcome of the algorithm is highly dependent on the initial conditions of the system but to the best of our knowledge no closed form analytical expressions have been proposed to model the behavior the algorithm

[2]This problems consists of building a rectilinear directed tree in a grid.

imposes on the streams. This makes mathematical modeling of the system extremely difficult. In this paper, we propose a modeling technique that treats the problem in the time domain. We show that this is a feasible way to model the problem and estimations based on this model yield results that are fairly close to the ones obtained from simulations. We mention related work in the area in Section 2. Section 3 discusses the theoretical variables used to describe a VoD system. Sections 4 and 5 show our technique applied to the Batching scheme and Snapshot-RSMA scheme respectively. Finally, we review our findings and summarize our results in Section 6, with some comments on current research directions.

2 Related Work

Batching schemes have been extensively studied in the literature [6, 7]. Dan et al. [6] studied the problem of capacity planning with statistical guarantees on system performance. Scheduling policies for batching schemes have also been proposed [7, 8]. Adaptive Piggybacking was first proposed by Golubchik et al. [1] with some simple merging algorithms. S. W. Lau et al. studied the complexity of the scheme and proposed some new heuristic merging methods in [5]. The main emphasis of [5] is on statistical bandwidth estimation using the proposed heuristics. An optimal solution to the static scenario of Adaptive Piggybacking was proposed by C. C. Aggarwal et al. [2], in which the authors also proposed the *snapshot* algorithm to deal with the dynamic scenario. Basu et al. [3] pointed out that the nature of the problem is isomorphic to an *RSMA* problem, which is a more generic framework for the binary tree merging proposed in [2]. More recently, Golubchik et al. [9] proposed a common framework to characterize aggregation schemes in general called *sync-classes*. Work in [9] does not emphasize any specific merging algorithm, but offers the tools by which different cost functions and user behavior models can be integrated into the aggregation schemes in an unified way. Such integration allows the system to compute the *optimal* merging pattern, with respect to the cost function utilized.

Our work emphasizes modeling in the time domain. An accurate time-domain model can yield satisfactory estimates on bandwidth utilization (by estimating the number of channels in time), as well as gives insight into the different trade-offs when changing system control variables. The simplicity of our scheme is useful for a VoD system engineer during initial design phases. Our results show the viability of modeling VoD systems using the principle of convolution. This principle is usually applied to finding the time response of linear time-invariant (LTI) systems. Of course, with stream interactions and merging of streams from different recomputation intervals, the superposition property, which is necessary for the application of the convolution theorem, cannot be satisfied generally. However, in many instances, when applied, this principle actually yields very satisfactory results. This prompts us to find the conditions under which it *can* be applied, rather than dismissing it altogether. Besides, being able to treat the VoD system in this manner means that we need to study mainly the *impulse response* of the system, which hopefully is simpler than treating the whole set of streams in the VoD system. By *impulse response* we mean the behavior of the VoD system, in terms of the number of channels required to support streams that arrive during one batching period or one recomputation interval. We exemplify the application of this principle through modeling the behavior in time of the number of channels in 2 aggregation schemes.

In the next section we will lay the foundations of our model and show the reasoning that prompted us to propose our technique.

3 Modeling a VoD system

We begin our analysis in this section with a search for proper models for client arrival process and client interaction processes. We then follow with a discussion of the two aggregation schemes and show how their characteristics suggest the application of the convolution theorem as a guiding principle for time-domain modeling.

A study conducted by Paxson and Floyd [10] showed that *individual* sessions of Internet utilization can be modeled as a Poisson process. We adopt this result and model the client request arrival in the VoD system as a Poisson process with mean rate λ_A. Since we are modeling an aggregation scheme, we study its behavior for a single video only. Thus, the rate λ_A is in fact a composite of the overall client arrival rate at the VoD server multiplied by the probability of accessing a particular movie in the video database[3].

Client interaction is also modeled as a Poisson process, with each client interacting with rate λ_i. This approach is adopted in a similar manner by other authors [1, 3, 6, 12]. The length of each interaction is exponentially distributed with mean $\mu \ll L$, in which L is the length of the video. Since the interaction processes are independent of each other, the resultant aggregate process is also Poisson, with rate given by the number of clients in the system multiplied by λ_i. We assume that these rates remain constant over time. For

[3]Video popularity is often modeled in the literature using a Zipfian distribution[11].

simulation purposes, we use a single value as the interaction rate of the system. This value was determined by first estimating the average number of users that will be present in the system when there are no interactions and then multiplying it by λ_i.

Notice that in the absence of interactions, the client that arrives at time t_0 leaves the system at time $t_0 + L$. Thus when the VoD system simulation is started, the number of clients in time steadily increases until L. Starting from this point, clients leave the system as well as arrive at the system. From the simulation runs we can actually see that the number of users reaches a plateau [3]. We refer to the system in this condition as having reached *steady-state*. Note that even in the presence of interactions, since we assumed that their duration is exponentially distributed with mean $\mu \ll L$, this steady-state condition will still be reached. Although in the presence of interactions it is harder to establish when exactly the steady-state is reached. Also when employing adaptive piggybacking, different users can receive different content progression rates, which increases the difficulty in determining when steady-state is reached. We therefore use a heuristic value, determined from observing the simulation runs, and consider that the system has reached steady-state after $10,000$ seconds of simulation time.

One thing we need to mention is that during this study there were no $QUIT$ interactionss. $QUIT$ interactions are those in which the client chooses to leave the VoD system before the completion of the display of the video. The absence of $QUIT$ simplifies the problem because the number of clients in the system can then be tracked roughly by only the arrival rate λ_A and the video length L. The types of interactions considered in this work are *Fast Forward* (FF), *Rewind* (RW) and *Pause* (PAU). The video length L is the third parameter that influences the performance of the aggregation scheme. We have set it to be 2 hours (7,200 sec.) in our simulations. We have not explored directly its influence in our study, but its presence is noted and accounted for during our analysis.

In our modeling attempts, we noticed that both aggregation schemes mentioned previously have a component that repeats itself in time. For the Batching scheme, client requests are batched during a waiting period and then released. For the Snapshot-RSMA algorithm, client requests arrive during a recomputation interval, create new channels and are subjected to the recomputation process at the end of the interval. This repetitive pattern suggested the possibility of looking at the long term system behavior (in terms of the number of channels needed) as a *sum* of the smaller components, i.e., from each batching period or recomputation interval. Clearly, this is analogous to the application of the convolution theorem in finding the system's time response of a linear time-invariant (LTI) system. In particular, the VoD

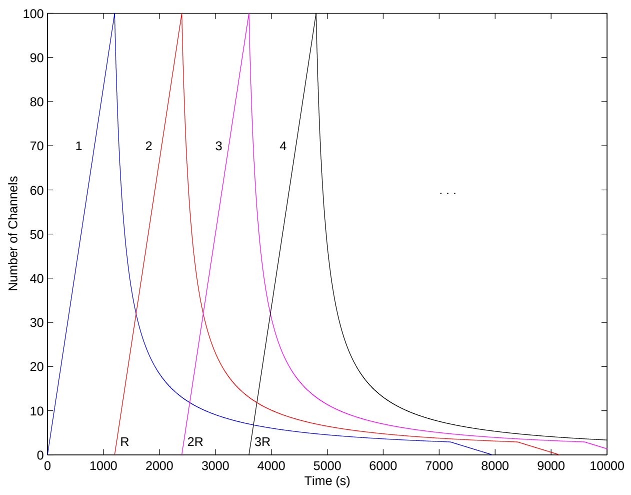


Figure 1: Response of a system to a periodical train of impulses with period R

system behavior can be seen as the response to a periodical train of impulses, with period determined by the batching period W or recomputation interval R. By *impulse* we refer to a function that satisfies the following two properties:

$$\delta(t) = \begin{cases} \infty & , \quad t = 0 \\ 0 & , \quad t \neq 0 \end{cases} \tag{1}$$

$$\int_{-\infty}^{\infty} \delta(t)\, dt = 1 \tag{2}$$

When an impulse is applied as the input signal of a system, the system response is termed, rather obviously, the *impulse response*. For LTI systems, if the impulse response is known, then the system response for any input signal can be found by applying the convolution theorem[4]. In our case we show that the VoD system response modeling problem can be treated *as if* there were a periodical train of impulses applied to the system, and the system response is the infinite sum of the impulse responses of the system.

The idea of infinite sum of impulse responses is illustrated by Fig. 1. Suppose the response of a system to an impulse is the function numbered “1” in Fig. 1. Then suppose the system is excited by an impulse every R units of time. The individual responses will be like the functions numbered 1 to 4 in Fig. 1, and the system response will be the *sum* of the functions 1 to 4 plus any other responses that may happen in the future. This basically is how we propose to model the VoD system. Each individual numbered function in Fig. 1 would be the response of the VoD system, in terms of the number of channels needed, due

[4]There are certain conditions that the input signal must satisfy in order to apply the convolution theorem but which are not relevant to our present discussion.

to a single batching period or recomputation interval. And we can find the average number of channels needed in steady-state by summing up the individual components.

Notice that, unlike real LTI systems, the VoD system *cannot* be characterized by *one* single impulse response, at least not in the Adaptive Piggybacking (Snapshot-RSMA case). When the recomputation interval (R) in the Snapshot-RSMA becomes small in comparison to the client inter-arrival period ($\frac{1}{\lambda_A}$), streams that arrive during different recomputation intervals influence each other, in the sense that streams from different recomputation intervals can be merged together. This means that the impulse responses are being changed according to the input signal. Thus the total system behavior cannot be modeled as the sum of the independent responses from each recomputation interval. In other words, under such circumstances, the superposition principle does not apply.

Our concern then, when we model the two schemes, is in finding the restrictions of our model and check what valid estimates we can draw from the results we obtain. In the next section, we will start by analyzing the Batching scheme.

4 The Batching Scheme

In this section we study a time-out based batching scheme. The first client request that arrives sets a timer that will expire after W units of time. W is the *batching* or *waiting period*. All users who arrive within this time period are said to be *batched* together and will be served by a single multicast channel upon the end of W units of time. We will use the term *batched channel* to refer to this initial channel. After W units of time, the first user that arrives requesting the same movie will set another timer, and this process repeats itself indefinitely. If a client interacts during the video, a new channel will be allocated to that client, which will remain with the client until the end of the video.

4.1 Without Interactions

In the absence of interactions, one batched channel will not influence the other, the superposition principle does apply and the system has a constant number of channels, on the average. Thus with a batching period of W and new client request arrival rate λ_A, the expected interval between two new channel allocations is given by the waiting time W plus the expected interval for a new arrival, which in our case is $\frac{1}{\lambda_A}$. The average new channel allocation

$\lambda_A \backslash W(s)$	300	600	900	1200
1/120	0.9765	0.9737	0.9904	1.0039
1/60	1.0014	1.0010	1.0017	0.9971
1/30	1.0020	0.9974	0.9997	0.9992
1/12	1.0062	1.0010	0.9982	1.0001

Table 1: Ratio of the predicted over the simulated values for the average number of channels in steady state in the Batching scheme with $\lambda_i = 0$

$\lambda_A \backslash W(s)$	1200	900	600	300
1/120	1.0037	0.9959	0.9903	0.9977
1/60	0.9854	0.9948	0.9904	0.9915
1/30	0.9898	0.9907	0.9914	0.9876
1/12	0.9884	0.9918	0.9923	0.9909

Table 2: Ratio of the predicted over the simulated values for the number of clients batched in each channel in the Batching scheme with $\lambda_i = 0$

interval is given by $W + \frac{1}{\lambda_A}$ and the average number of channels C_B in this scenario is roughly given by Eq. 3.

$$C_B = \frac{L}{W + \frac{1}{\lambda_A}} \tag{3}$$

In Table 1 we check the accuracy of Eq. 3 through simulation. We first simulated the Batching scheme for various arrival rates λ_A and batching periods W. For each case we sampled the number of channels in the VoD system during steady state for 20,000 sec. of simulation time and averaged the sampled values. Then we divided the value predicted through mathematical analysis by the value obtained through simulation to check how close they were to each other. This is how we obtained the ratios in Table 1. The same procedure was taken to obtain the values in Tables 2 and 3.

The next value we investigate is the average number of clients that are batched in each channel during the batching period. This value can be obtained by Eq. 4.

$$N_B = 1 + E_{\lambda_A}[W] = 1 + \lambda_A W \tag{4}$$

$E_{\lambda_A}[W]$ is the expected number of arrivals in the waiting period W when the arrival rate is λ_A. To this we add one because with the batching scheme employed here we are guaranteed one client at the beginning of the waiting period. We compare the predicted values of Eq. 4 to simulation results in Table 2.

Because with this scheme more than one client will be served by a single channel, we

$\lambda_A \backslash W(s)$	300	600	900	1200
1/120	0.9796	0.9805	0.9637	0.9509
1/60	0.9751	0.9752	0.9758	0.9763
1/30	0.9828	0.9865	0.9858	0.9845
1/12	0.9892	0.9939	0.9957	0.9944

Table 3: Ratio of the predicted over the simulated values for the average number of clients per channel in steady state in the Batching scheme with $\lambda_i = 0$

define *gain* as the ratio of the average number of clients per channel in steady state over the average number of clients per channel if there are no aggregation schemes (i.e., one). A good estimate of the gain of the system allows the VoD system designer to predict reasonably the capacity needed for the system.

To obtain such estimate, notice that in the Batching scheme, clients who are batched and waiting for a new channel are part of the system but do not occupy any channel. If the clients arrive with rate λ_A, then the average number of clients seen by the system during an interval T[5] is given by $\frac{\lambda_A T}{2}$ and the average number of channels in steady state is given by Eq. 3. Thus the average number of clients per channel, i.e., the steady state gain G_B of the system is:

$$G_B = \frac{\lambda_A W/2}{L/(W + \frac{1}{\lambda_A})} + 1 + \lambda_A W \qquad (5)$$

Table 3 compares Eq. 5 with simulation results.

Tables 1, 2 and 3 show us that the equations deduced in this section fall within 5% of the simulated values, proving themselves to be fairly accurate predictors of the behavior of the Batching system in the absence of interactions.

4.2 With Interactions

If we include interactions, clients that have been batched into one channel may not remain until the end of the video. They may perform interactions and thus generate new channels. In our modeling, a client that interacts is given a new channel until the end of the video.

[5]In mathematical terms, we are computing $\frac{1}{T}\int_0^T \lambda_A t\, dt$.

To model the new channel generation process due to interactions we use non-stationary Poisson arrival process. A batched channel will initially hold N_B clients, as determined by Eq. 4. Each of these clients will interact with rate λ_i. The resultant new channel generation process per batched channel can then be viewed as composed of superpositions of independent Poisson arrival processes, which is in itself a Poisson arrival process. Once a client interacts, it will stay with a channel of its own and further interactions will not generate new channels. This means that the new channel generation rate λ_C is not constant in time and is dependent on the number of clients that are still in the initial batched channel. Thus the new channel generation process can be better modeled as a non-stationary Poisson arrival process. The channel generation rate $\lambda_C(t)$ of this process can be approximated by $\lambda_i \, N_B(t)$, where $N_B(t)$ is the expected number of users that remains in the original batched channel at time t. Since each new channel has one interacting client, the expected number of new channels is exactly equal to the expected number of clients that left the original batched channel. If we express the above paragraph in terms of mathematical equations, we come to Eq. 6 and 7.

$$\begin{aligned} \lambda_C(t) &= N_B(t)\,\lambda_i & (6) \\ N_B(t) &= N_B(0) - \int_0^t \lambda_C(s)\,ds & (7) \end{aligned}$$

If we substitute $N_B(0) = 1 + \lambda_A\, W$ and solve Eq.s 6 and 7, we can see that $\lambda_C(t)$ is:

$$\lambda_C(t) = (1 + \lambda_A\, W)e^{-\lambda_i t}\,\lambda_i \qquad (8)$$

Note that this is the new channel generation rate for a single batched channel. If we integrate this over time, and remembering that at time $t = 0$ there is one channel (the batched channel), we obtain an expression on the number of channels $C_B(t)$ in the system due to one initial batched channel as given by Eq. 9.

$$C_B(t) = 1 + \int_0^t \lambda_C(s)\,ds = 1 + (1 + \lambda_A\, W)(1 - e^{-\lambda_i t}) \qquad (9)$$

Note that the time $t = 0$ is the time when the video stream is *started*, in other words, one batching period after the arrival of the first client. Since there cannot be more channels than clients, we limit the possible expected number of new channels by the expected number of people in the channel. At the end of the video all users exit, and the number of channels drop to 0. Thus an appropriate expression for the number of channels in the system due to the very first batched channel is:

$$C(t) = \begin{cases} 0 & , \quad t \leq \frac{1}{\lambda_A} + W \\ \min(C_B(t - \frac{1}{\lambda_A} + W)), 1 + \lambda_A W) & , \quad \frac{1}{\lambda_A} + W < t < L + \frac{1}{\lambda_A} + W \\ 0 & , \quad t \geq L + \frac{1}{\lambda_A} + W \end{cases} \tag{10}$$

Of course clients that interact will remain in the system longer than L. In our model, we assumed that the length of interaction is negligible in comparison to L.

With this we have a prediction model on one batched channel. This is the impulse response of our modeling process. Notice that since clients who interacted remain in their channels until the end of the video, there is no interaction between clients from two separate batching periods. This allows us to describe the system response as the sum of the responses from each new channel in the system. Assuming that the system parameters remain constant, each new batching period will contribute in the same manner to the system response. Mathematically, this translates to Eq. 11.

$$C_{BT}(t) = \sum_{n=0}^{\infty} C(t - n(1/\lambda_A + W)) \tag{11}$$

To check our proposed equations, we simulate the system with different values of λ_A and W. Due to space restrictions, we illustrate the evolution of the VoD system in time, in terms of the number of users and the number of channels, for $\lambda_A = 1/30$, $\lambda_i = 1/hour$ and $W \in \{300, 600, 900, 1200\}$ (Fig. 2).

Fig. 2 shows us that our prediction equations approximate the system behavior in time well. If we check the steady-state bandwidth requirement, we can obtain the ratio of the average number of channels needed through simulation and through prediction in Tables 4 and 5. The average is taken from samples starting from $t = 10,000$ until $t = 30,000$. The predicted value is obtained by evaluating Eq. 11 numerically at the same points sampled by the simulation and then computing their average.

From Tables 4 and 5 we see that our predictions fall within 5% of the simulated value most of the time, with two instances when the error was more than 5% but still below 10%. The larger error for those two instances are due to the higher interaction rate ($\lambda_i = 1/hour$) and the relatively high ratio between the client inter-arrival time and the batching period. This lower arrival rate means that there will be fewer arrivals in each batching period, and a more accurate characterization would need longer simulation runs. But even with these two instances, our model can track the average number of channels required during steady state fairly well.

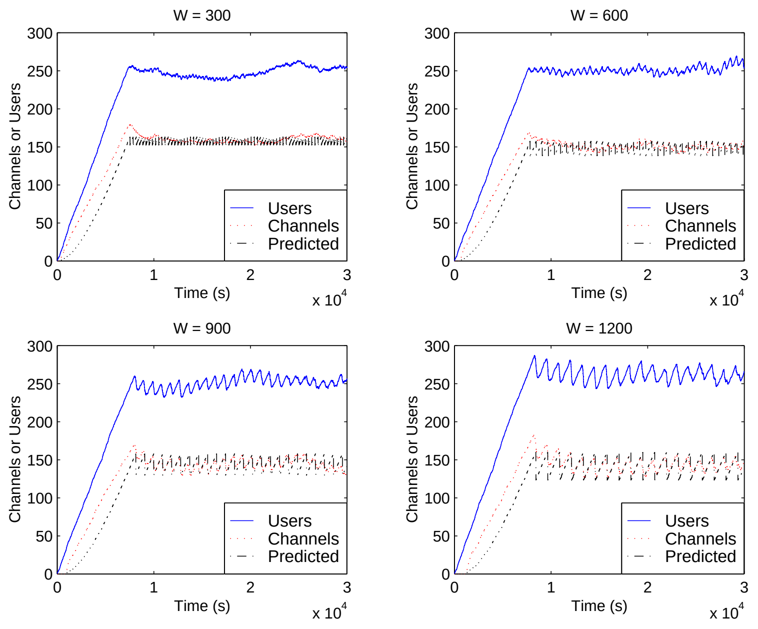


Figure 2: Time response of batching with $\lambda_A = 1/30, \lambda_i = 1/hour$

$\lambda_A \backslash W(s)$	300	600	900	1200
1/120	1.0316	0.9958	1.0265	0.9867
1/60	1.0284	1.0035	0.9797	1.0001
1/30	1.0173	1.0023	0.9932	0.9823
1/12	0.9991	1.0140	0.9836	1.0086

Table 4: Ratio of the predicted over the simulated values for the average number of clients per channel in steady state in the Batching scheme with $\lambda_i = 0.5$/hour

$\lambda_A \backslash W(s)$	300	600	900	1200
1/120	1.0858	1.0369	1.0040	1.0138
1/60	1.0557	1.0160	1.0177	1.0184
1/30	0.9920	0.9819	0.9999	0.9934
1/12	0.9967	0.9833	0.9877	0.9918

Table 5: Ratio of the predicted over the simulated values for the average number of clients per channel in steady state in the Batching scheme with $\lambda_i = 1$/hour

We have thus developed a model for the Batching by time-out scenario with interaction support. Our model makes good estimates on the number of channels required during steady-state even in the presence of high interaction rates.

5 The Snapshot-RSMA Scheme

Batching schemes rely on delaying the response time to client requests. A new idea proposed by Golubchik et al. [1] rely on different content progression rates for video streams while these are being sent across the network. In such a scheme, the start-up latency to a client is negligible. And it is during the time of display that different streams have their progression rates altered and eventually merged together and served collectively. This concept has been termed *Adaptive Piggybacking*. Many algorithms have been proposed on how to choose the display rate of each stream in a VoD system so as to minimize network bandwidth requirements. *Snapshot-RSMA* is one such algorithm, and it is the object of study in this section.

The *Snapshot-RSMA* scheme derives its name from its main two components. *Snapshot* refers to the action of obtaining the positions of non-interactive streams in the system at regular intervals. *RSMA* refers to the mathematical nature of the merging problem, which consists of building a binary tree with minimum path length on a rectilinear grid. The solution to this problem is obtained through a dynamic programming algorithm.

In this scheme, all new client requests are served immediately, i.e., a new channel is allocated, a new stream is initiated and the normal display rate is given to the stream. This display rate is maintained until the time a new recomputation takes place. Recomputations are separated by R units of time, also known as the recomputation interval. At recomputation, the dynamic programming algorithm is applied and display rates are determined for all non-interactive streams. There are two options available for the display rate: the normal display rate ($S_n = 1$) or the accelerated display rate ($S_a = \frac{32}{30}$). The display rate value of 1 means that for each second in the real world, 1 sec. worth of the video is being displayed. The accelerated rate obviously imply that for each second in the real world, more than one second of "video time" is being displayed. The frame rate of the normal display rate is 30 frames per second, while the accelerated rate displays 32 frames per second (thus the rather odd $\frac{32}{30}$ notation for S_a). The different display rates for the streams allows stream merges. Such merges are advantageous for the VoD server since they release channels that can be reused.

While studying this scheme, the first problem we faced was the lack of any analytical formula that can model the merging process. The merging process is determined by the dynamic programming algorithm, and such process is highly dependent on the initial condition of the system, i.e., different stream positions might lead to different results. Since the stochastic nature of the Poisson arrival process makes the modeling of the merging process under Snapshot-RSMA particularly challenging, we will start with a simpler arrival process, namely the uniform arrival process. We chose the uniform arrival process because the behavior of the dynamic programming for uniformly spaced streams is known. Thus it is possible to track the system behavior in time and see how closely our model matches reality.

The optimal merging tree for uniformly spaced streams will simply merge the streams 2 by 2, and the resulting streams also in a 2 by 2 fashion, and so on. Now, two consecutive streams separated by d seconds (in "video time") can meet after $\frac{d}{\Delta S}$, where $\Delta S = S_a - S_n$. To see why, consider that at time $t = 0$, stream i is at position $d > 0$ and given display rate S_n (normal) and stream j is at position 0 with display rate S_a (accelerated), then they will meet when $S_a\, t = d + S_n\, t$, i.e., when $t = \frac{d}{(S_a - S_n)}$.

Suppose that initially we had $C_S(0)$ channels that are equally spaced by $\frac{1}{\lambda_A}$ and their display rates are determined by the RSMA algorithm at time $t = 0$. Since they are equally spaced, the optimal tree will simply merge them in pairs. Thus after $\frac{1}{\lambda_A \Delta S}$ the streams will have merged in pairs and there will be $\frac{1}{2}C_S(0)$ streams left. This is the first merging point. We will at this point give some specific names to the streams to facilitate our explanation. We will consider the leading video stream as the "first" stream (the one closest to the end of the video) and sort the other streams in decreasing stream position order. Thus the "first" merging point is when the first stream meets the second stream, the third meets the fourth, the fifth the sixth, and so on. Since we assumed all streams are equally spaced, the second stream was $\frac{1}{\lambda_A}$ apart from the first stream. Now if we give thought to this process, it is evident that the second merging point will be when the first stream meets the fourth stream. At that point in time the original $C_S(0)$ streams will be merged into $\frac{1}{4}C_S(0)$ streams. This will happen at time $t = \frac{3}{\lambda_A\, \Delta S}$ (notice that the first and the fourth stream are separated by $\frac{3}{\lambda_A}$). In fact, if we carry this on, we will arrive at Eq. 12 for the number of channels after a recomputation.

$$C_S(t) = \frac{C_S(0)}{\lambda_A\, \Delta S\, t + 1}, \quad t = \frac{1}{\lambda_A \Delta S}, \frac{3}{\lambda_A \Delta S}, \frac{7}{\lambda_A \Delta S}, \cdots \tag{12}$$

We illustrate our reasoning with Fig. 3. In it, $C_S(0) = 8$. At time $t = 0$ a recomputation

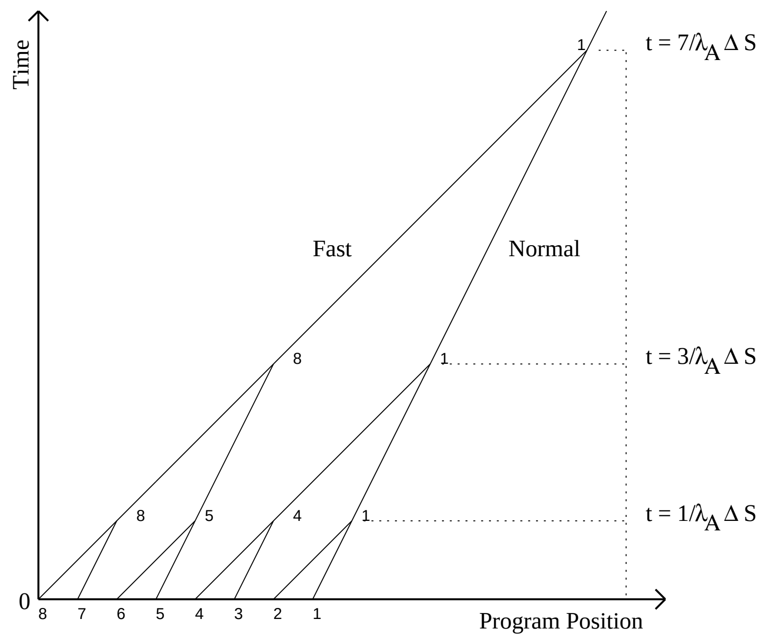


Figure 3: Merging pattern of eight uniformly spaced streams

process took place and streams 1 to 8 are given different display rates. Because they are uniformly spaced, the Snapshot-RSMA algorithm merges them in consecutive pairs. The first merge takes place at time $t = \frac{1}{\lambda_A \Delta S}$. As a result, from the initial eight streams only four are left. From the resulting four streams, streams 1 and 4 will merge, and streams 5 and 8 will merge. They were originally (at time $t = 0$) separated by $\frac{3}{\lambda_A}$, and thus these two pairs will merge when $t = \frac{3}{\lambda_A \Delta S}$. Finally only one stream is left after the merge at time $t = \frac{7}{\lambda_A \Delta S}$, joining the original streams 1 and 8. This last stream may eventually join with other streams in the system or it may reach the end of the video before that.

Notice that in between the merging events the number of streams remains constant. If we define the window function $W(t, \tau)$ as:

$$W(t,\tau) = \begin{cases} 0 & , \quad t < 0 \\ 1 & , \quad 0 \le t < \tau \\ 0 & , \quad t \ge \tau \end{cases} \tag{13}$$

then a more accurate description of $C_S(t)$ would be:

$$C_S(t) = \sum_{n=0}^{\infty} \frac{C_S(0)}{2^n} W\left(t - \frac{2^n - 1}{\lambda_A \Delta S}, \frac{2^n}{\lambda_A \Delta S}\right) \tag{14}$$

Before proceeding, we would like to point out some issues. Note that $C_S(0)$ is divided by 2^n indefinitely, which potentially can make $C_S(t) < 1$. This simply means that if there were other streams besides the initial set $C_S(0)$, then at time t when $C_S(t) < 1$, merges would have happened. This will become relevant as we develop our ideas in the following paragraphs. Also, we need to emphasize that Eq. 14 does not attempt to predict exactly the evolution of the number of channels in time. If $C_S(0)$ is a power of 2 and if there are no interactions, then Eq. 14 is exact, but if that is not the case, merges will not take place at the times predicted, and Eq. 14 becomes only an estimate. The point to be stressed is that we are attempting in this paper to obtain a broad picture on the *general* behavior of the number of channels in time. The time $t = 0$ in Eq. 14 is actually the time immediately after a recomputation process.

With all the above discussion in mind, we will formulate a more accurate equation for the Snapshot-RSMA scheme. If we look at the scheme's evolution in time, initially (from $t = 0$ - this is the *system* time) new clients will arrive, generating new channels at the same rate as the arrival rate λ_A. At time $t = R$ a recomputation will happen and the number of streams will change according to Eq. 14. At time $t = L$ streams will start reaching the end of the video and will exit gradually. To model the client exit behavior we will resort to a linear model. The client that arrived last in the recomputation interval should receive the accelerated display rate *always*, since it is the trailing stream. If we assume, for modeling purposes, that it arrived at $t = R$, the time of its exit will be $R + \frac{S_n}{S_a}L$. Likewise, the *first* client to arrive in the recomputation interval will be given normal display *always*, since it is the leading stream. Considering that this leading client arrived at time $t = 0$, it will exit at time $t = L$. Because of the cost formulation proposed in [2], in the event of a merge between the trailing stream and the leading stream, the leading stream remains[6]. To model this behavior, we determined that if the number of channels at time $t = L$ is less than one, then all streams have merged and all will exit together with the leading stream. However, if the number of channels at time $t = L$ is greater than one, then the exit pattern will be linear from $t = L$ until $t = R + \frac{S_n}{S_a}L$. A more complete formulation for the behavior on the channels that arrived during a single recomputation interval can then be given by Eq. 15.

[6]The cost formulation in [2] involves the number of frames until the end of the video, not the time taken to show it. It is therefore independent of the display rate. This favors the survival of the leading stream, so that it can stay longer and merge with other streams.

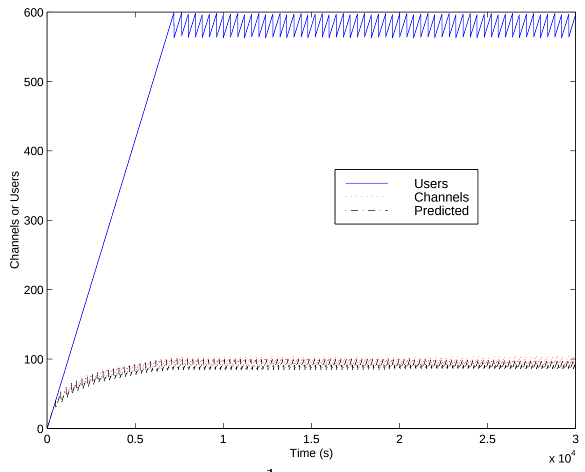
$$C_S(t) = \begin{cases} 0 & , \quad t \leq 0 \\ \lambda_A t & , \quad 0 < t \leq R \\ \sum_{n=0}^{\infty} \frac{C_S(R)}{2^n} W(t - R - \frac{2^n - 1}{\lambda_A \Delta S}, \frac{2^n}{\lambda_A \Delta S}) & , \quad R < t \leq L \\ 0 & , \quad t > L, \text{ if } C_S(L) < 1 \\ C_S(L) - \frac{C_S(L)}{R + \frac{S_n}{S_a} L - L} t - L & , \quad L \leq t \leq \max(L, R + \frac{S_n}{S_a} L), \text{ if } C_S(L) \geq 1 \\ 0 & , \quad t > \max(L, R + \frac{S_n}{S_a} L) \end{cases} \tag{15}$$

Eq. 15 describes the evolution of the number of channels in time due to client arrivals during $0 < t < R$. Note that because the trailing stream is always given an accelerated display rate, it will go further apart from the leading stream from the following recomputation interval. This is a consequence of the Snapshot-RSMA scheme and it leads us to believe that it will favor merging the streams that arrived within the same recomputation interval. However, because the recomputation process takes into account *all* streams watching the same video, if the video length is long enough, clients that arrived in different recomputation intervals can still be joined together. In mathematical terms, this means that the number of streams from that recomputation interval effectively falls below one. This is what we pointed out previously. Note that the natural clustering mechanism favors our impulse response modeling of the system, since each recomputation interval would contribute in an independent manner to the overall system response. However, merges of streams from different recomputation intervals do happen in the long term and needs be addressed in our model. Allowing the number of streams from each recomputation interval to fall below one actually takes into account the merges while still preserving the independent character of each recomputation interval.

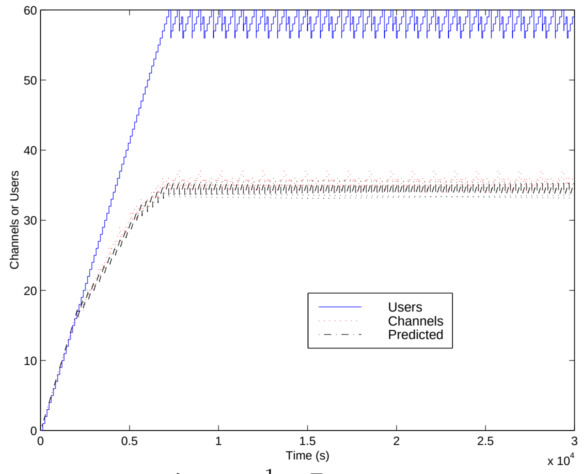
Assuming that there are no significant differences in the responses from each recomputation interval, we sum the contribution of each recomputation interval to obtain the system behavior. The evolution in time of the number of channels in Snapshot-RSMA is then given by Eq. 16. Note that Eq. 16 is exactly the convolution of Eq. 15 with a periodical train of impulses with period R (the first impulse starting at time $t = 0$).

$$C_{ST}(t) = \sum_{n=0}^{\infty} C_S(t - nR) \tag{16}$$

To check Eq. 16 we ran simulations again. We show some time domain results for $\lambda_A = 1/12$ and $\lambda_A = 1/120$ in Fig. 4. From the figures we can see that our model, while not completely successful in its exact accuracy, still closely matches the system behavior.



$\lambda_A = \frac{1}{12}, R = 300$



$\lambda_A = \frac{1}{120}, R = 300$

Figure 4: Simulation vs Prediction for Snapshot-RSMA, Uniform Arrival

$\lambda_A \backslash R(s)$	150	300	600	900	1200
1/120	0.9505	0.9684	0.9570	0.9895	0.9809
1/60	0.9709	0.9981	0.9962	1.0168	1.0063
1/30	0.9621	0.9653	0.9639	0.9174	0.8825
1/12	0.9305	0.9409	0.9265	0.9100	0.9098

Table 6: Ratio of the predicted over the simulated values for the average number of channels in steady state in the Snapshot-RSMA scheme (Uniform Arrival case)

In fact, if we compute the estimate of the average number of channels in the system (for $t > 10,000$) by prediction and by simulation, we see that our model can predict the average number of channels within 10% accuracy most of the time, as shown by Table 6. Again, the predicted value was obtained by evaluating Eq. 16 numerically at the same points sampled in the simulation and computing their average value.

We found two main sources of error. The first being the fact that most part of the time, $C_S(R)$ is not a power of 2. This means that there will be discrepancies in the progression on the number of channels in time. The second kind of error happens when the recomputation interval is too long. Consider that a recomputation took place at time $t = R$, then, after an interval of $\frac{1}{\lambda_A \Delta S}$, according to Eq. 15, $C_S(R + \frac{1}{\lambda_A \Delta S})$ is equal to $\frac{1}{2}C_S(R)$. Now, because to the snapshot nature of the algorithm, if R is greater than $\frac{1}{\lambda_A \Delta S}$, then the $\frac{1}{2}C_S(R)$ streams at $t = R + \frac{1}{\lambda_A \Delta S}$ will not "know" how they should act in order to keep the merging rate predicted by Eq. 15. The result of this loss of knowledge means that the behavior of the impulse response will differ from the one predicted by Eq. 15 and the accuracy of our model will be degraded. Thus a recomputation interval R should be of such length that either it is itself close to (by *close* we mean equal to or a little bit larger than) $\frac{1}{\lambda_A \Delta S}$ or an integer multiple of R is close to $\frac{1}{\lambda_A \Delta S}$.

R (s)	96	150	192	300	384	600	768	900	1200
$\lambda_A = 1/12$	0.9903	0.9305	0.9832	0.9409	0.9540	0.9265	0.8787	0.9100	0.9098

Table 7: Ratio of the predicted over the simulated values for the average number of channels in steady state in the Snapshot-RSMA scheme with $\lambda_A = 1/12$

To support our analysis, we ran more simulations with $\lambda_A = \frac{1}{12}$ and added extra values for $R \in \{96, 192, 384, 768\}$. Notice that for these values of R, $\lambda_A R$ results in powers of 2. We ran simulations and compute the ratios of the predicted over the simulated values for $t > 10,000$ in Table 7.

For $\lambda_A = \frac{1}{12}$, $\frac{1}{\lambda_A \Delta S} = 180$. Thus for values of R which are close to 180 or which have an integer multiple close to 180, i.e., $R \in \{96, 192\}$ it is expected that the prediction will be more accurate, which indeed is what happens, according to Table 7. When R is much greater than 180 ($R \in \{384, 768\}$), errors become more evident, according to the results from the same Table. Also values of R which resulted in $C_S(R)$ as non-power of 2 showed more evident error. This concludes our analysis of the uniform arrival. We will next turn to the Poisson arrival process.

The stochasticity of the Poisson arrival process does not allow us a nice mathematical formulation as the one in Eq. 15. If we attempt to predict the average number of channels needed in steady state by a system with Poisson arrival process by a uniform arrival formulation, we obtain Fig. 5. In Fig. 5 we plot the ratio of the two values (predicted divided by simulated).

Fig. 5 shows us that the uniform arrival overestimates the number of channels needed by the Poisson arrival process in most cases. Indeed, only for $\lambda_A = \frac{1}{12}$ and recomputation interval $R \geq 900$ does the model underestimate the number of channels. The non-horizontal slope indicates that the uniform arrival cannot keep up with the nature of the Snapshot-RSMA applied to a Poisson arrival as the recomputation interval increases. In other words, the type of the arrival process *does* matter when modeling the Snapshot-RSMA algorithm for Adaptive Piggybacking.

Because of the complexity of the mathematical formulation for the Poisson arrival, we propose here a *sampling* technique. Basically we try to sample the VoD system behavior for one recomputation interval. Recall that in our modeling principle, the response for one single recomputation interval is the impulse response of the system. If we have the impulse response, then we can predict the time domain behavior of the VoD system according to Eq. 16, in which $C_S(t)$ is the sampled function. Basically we obtain numerical values for $C_S(t)$

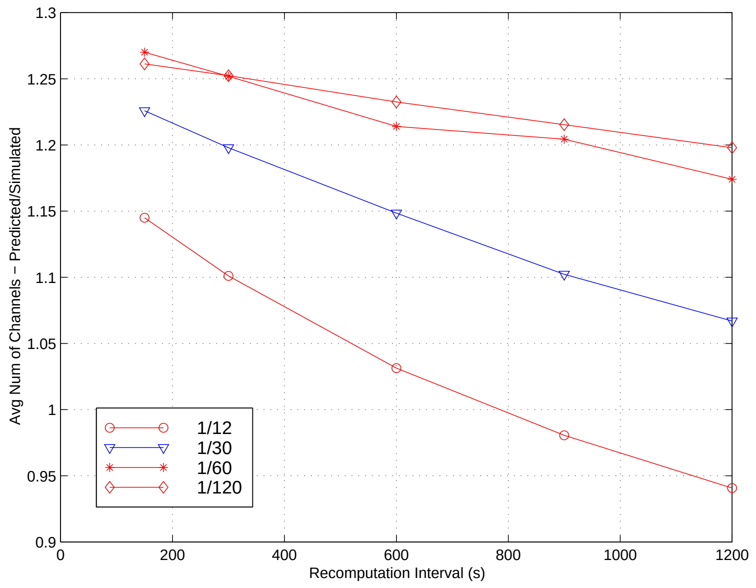


Figure 5: Ratio of prediction using uniform arrival model over simulation using Poisson arrival

(Eq. 15) through simulation and substitute them as $C_S(t)$ in Eq. 16.

To check the accuracy, we plot the ratio of the predicted number of channels vs. results from simulation in Fig. 6. According to Fig. 6, this “impulse response sampling” method works best if the average number of arrivals is high in the recomputation interval. Sampling the system behavior for one recomputation interval cannot reflect any interactions (i.e., mergings) between streams from different recomputation intervals. These interactions are most likely to occur if only a few streams arrived during the recomputation interval, because in this case the Snapshot-RSMA will attempt to merge streams from different recomputation intervals. However, we can see that when the influence of inter-cluster interaction is small, our modeling methodology yields extremely satisfactory results, as seen in Fig. 6.

We can see that in modeling the Snapshot-RSMA case, our principle yields accurate prediction values. For the uniform arrival process, the resultant mathematical model is built entirely through analytical means. When the mathematical formulation is too complex, as in the case of the Poisson arrival, we can resort to sampling the VoD system response to arrivals that occur during a single recomputation interval and use the result of such sampling to build a steady state formulation.

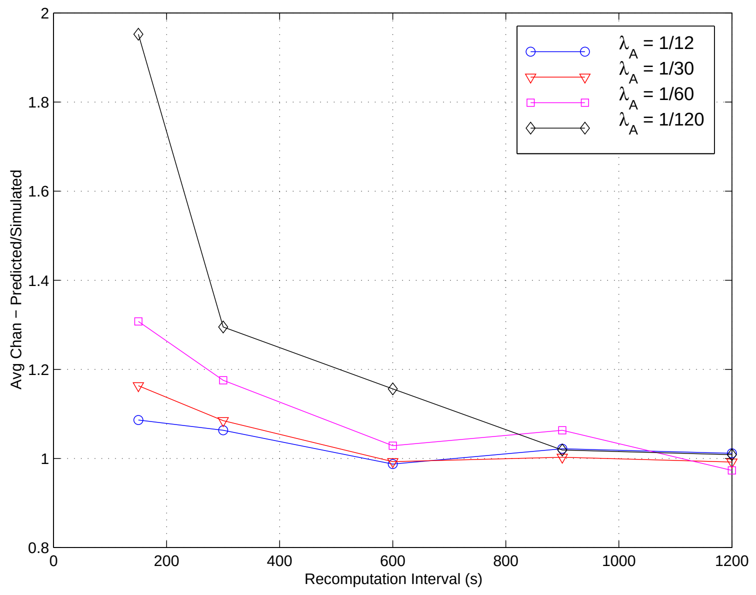


Figure 6: Ratio predicted/simulated using impulse sampling

6 Conclusion

We have proposed in this paper a principle that can be used to model the response in time of some types of VoD aggregation schemes. If such aggregation schemes rely on patterns that repeat periodically, then we propose to model the scheme by obtaining the time response of one period and summing the responses from each period. Clearly this is equivalent to obtaining the impulse response and applying the convolution theorem to find the time response of linear time invariant (LTI) systems. And just like linear time invariant systems, such modeling principle can only be applied under limited conditions.

We have analysed 2 aggregation schemes: Batching by time-out and Adaptive Piggybacking employing Snapshot-RSMA. In both instances we showed that our modeling principle yields highly satisfactory results when the VoD system behaves like an LTI system. In the Batching scheme that means no stream merges from clients that arrived in different batching periods. In the Adaptive Piggybacking case, for the Uniform Arrival process, we have shown that the recomputation interval (or an integer multiple of it) should be close to $\frac{1}{\lambda_A \Delta S}$ to take full advantage of the merging capacity of the dynamic programming algorithm. For the Poisson Arrival process, we proposed sampling the response from one recomputation interval for system modeling. We found that this technique is applicable when the recomputation interval is large with respect to the inter-client request arrival period. If such conditions are met, even in the absence of a closed mathematical formulation, we can still obtain satisfactory time-domain modeling results.

There are still other modeling problems that can be tackled in the future. We are currently investigating the mathematical analysis for the uniform arrival case for Snapshot-RSMA with support for client interactions. We hope to extend our work to Poisson arrivals as well, and to examine any limitations that this added support might bring to the impulse sampling methodology.

References

[1] L. Golubchik, J. C. S. Lui and R. R. Muntz. "Adaptive piggybacking: a novel technique for data sharing in video-on-demand storage servers", *Multimedia Systems*, Vol. 4, No. 3, pp. 140–155, Jun 1996.

[2] C.C. Aggarwal, J.L. Wolf and P.S. Yu. "On Optimal piggyback merging policies for Video-on-Demand Systems", *Proceedings of the 1996 ACM Sigmetrics*, Philadelphia, PA, USA, pp. 200–209.

[3] P. Basu, R. Krishnan and T.D.C. Little. "Optimal Stream Clustering Problems in Video-on-Demand", *Proc. Parallel and Distributed Computing and Systems '98 - Special Session on Distributed Multimedia Computing*, Las Vegas, NV, USA, pp. 220-225, Oct 1998.

[4] T. D. C. Little and D. Venkatesh, " Popularity-Based Assignment of Movies to Storage Devices in a Video-on-Demand System", *Multimedia Systems*, Vol. 2, No. 6, pp. 280–287, Jan 1995.

[5] S. W. Lau, J. C. S. Lui and L. Golubchik. "Merging video streams in a multimedia storage server: complexity and heuristics", *Multimedia Systems*, Vol. 6, No. 1, pp. 29–42, Jan 1998.

[6] A. Dan, P. Shahabuddin, D. Sitaram and D. Towsley. "Channel Allocation under Batching and VCR control in Video-on-Demand Systems", *Journal of Parallel and Distributed Computing*, Vol. 30, No. 2, pp. 168–179, Nov 1995.

[7] A. Dan, D. Sitaram and P. Shahabuddin. "Dynamic Batching Policies for an On-Demand Video Server", *Multimedia Systems*, Vol. 4, No. 3, pp. 112–121, Jun 1996.

[8] C. C. Aggarwal, J. L. Wolf and P. S. Yu. "Optimization Issues in Multimedia Systems", *International Journal of Intelligent Systems*", Vol. 13, No. 12, pp. 1113–1135, Dec 1998.

[9] L. Golubchik, V. S. Subrahmanian, S. Marcus and J. Biskup. "Sync Classes: A Framework for Optimal Scheduling of Requests in Multimedia Storage Servers", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 12, No.1, pp. 60–77 Jan/Feb 2000.

[10] V. Paxson and S. Floyd "Why We Don't Know How To Simulate The Internet", *Proceedings of the 1997 Winter Simulation Conference*, Atlanta, GA, 1997

[11] A. S. Tanenbaum, *Computer Networks*, Third Edition, Prentice Hall PTR, 1996

[12] V. O. K. Li, W. Liao, X. Qiu and E. W. M. Wong, "Performance Model of Interactive Video-on-Demand Systems", *IEEE Journal on Selected Areas in Communicatinos*, Vol. 14, No. 6, Aug 1996