A Queuing Analysis of Bandwidth Allocation Schemes for Compressed Video

Saurab Nog and Carl J. Beckmann

Department of Computer Science
Dartmouth College
Hanover, NH 03755-3510

saurab@cs.dartmouth.edu, carl.beckmann@dartmouth.edu

Dartmouth Technical Report PCS-TR96-257
January 14, 1996
Revised March 25, 1996

Abstract

Video and audio compression techniques allow continuous media streams to be transmitted at bit rates that are a function of the delivered quality of service. Digital networks will be increasingly used for the transmission of such continuous media streams. This paper describes an admission control policy in which the quality of service is negotiated at stream initiation, and is a function of both the desired quality of service and the available bandwidth resources. The advantage of this approach is the ability to robustly service large numbers of users, while providing increased quality of service during low usage periods. Several simple algorithms for implementing this policy are described and evaluated using queuing model analysis applied to video-on-demand. The queuing model results are compared with simulation results to validate their accuracy.

Keywords - Admission Control, Quality of Service, Scalable Compression, Video on Demand, Queuing Analysis, State-Space.

Revised - March 25, 1996 to include discrete event simulation results.

1 Introduction

In this paper we consider the problem of bandwidth reservation in multimedia systems with selectable video compression rates/delivered quality. Consider, for instance, a video-on-demand system where digital video is transmitted over a network with limited capacity. The number of users or *streams* is highly variable, with an average value which is a function of the time of day (see Figure 1). Each stream contains audio and video information, and lossy compression can be used to reduce the bandwidth requirement of a stream, although some loss in fidelity will occur. During peak usage (*prime time*), it is desired that the system handle as many users as possible with some loss in fidelity per user if necessary. During off hours, it is desired that each user obtain as much fidelity as possible, given the overall capacity of a network. At all times, it is also desired that users signing on to the system experience as little delay as possible until the start of their stream.

Before a stream is initiated the user's receiver negotiates a fixed bandwidth with the network, and the capacity allocated to this stream is fixed for its duration (the *reservationist* model of ATM networks [And93, Lan94, Mil94]). It should be noted that this general problem applies not only to bandwidth allocation in a transmission network, but is also applicable to disk or I/O channel capacity allocation within a multimedia server [RC95]. The problem we consider here is how to allocate available network bandwidth on stream initiation in order to service as many users as possible, but also provide each one with the highest quality of service. This is distinct from the problem of *enforcement* once a QOS contract has been negotiated [Jul94, SS94].

As an example to give representative numbers, the network might be a fiber-optic ATM network with a data rate of 622 Mbit/s, serving up to 250 video-on-demand users in a fiber-to-the-curb configuration [Bru94]. Each stream is a digitized NTSC video signal with 500 lines of vertical resolution and 300 lines of horizontal resolution, at 30 frames per second, and 8 bits of luminance and 1 bit of color information per pixel, for an uncompressed data rate of 40.5 Mbit/s. Lossless compression can reduce this rate by a factor of three or four, and lossy compression using MPEG encoding [CAGM94, Gal91, PZ94, SS95] can yield data rates of 6Mbit/s down to 1.5 Mbit/s depending on the quality desired. A CD quality audio stream consists of stereo channels sampled at 44.1 kHz each, with 16 bits per sample, for a raw data rate of 1.4112 Mbit/s and with techniques used in digital tape and disk recording devices, lossy compression ratios of 4:1 can be achieved to yield data rates as low as 350 Kbit/s [Yos94, Nol95].

Thus, depending on the quality of service a combined video and audio stream may require anywhere from 1.5 Mbit/s (MPEG compressed VHS-quality audio/video) to 42 Mbit/s (uncompressed broadcast quality video, CD quality audio). If all 250 users request service, lossless fidelity cannot be provided with only 622 Mbit/s, although it does suffice to give all users the lowest quality of service.

To implement selectable bit-rate streams, a video server could store two or three versions of each movie on disk, one for each compression rate ("HDTV quality", "Broadcast quality"

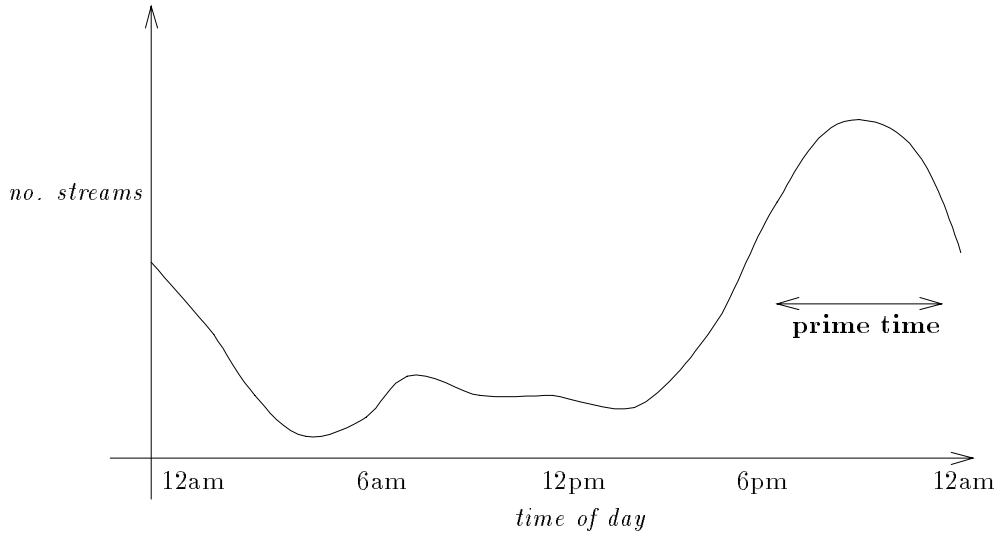

Figure 1: Typical VOD daily usage profile

and "VHS quality", for instance). Note that storage space is always dominated by the highest-quality version. Alternately, a hierarchical compression scheme such as MPEG-2 can be used to provide bandwidth scalability from a single stored media stream [DHH+94, SS95].

2 Static Bandwidth Allocation Algorithms

In this paper, we consider a number of possible algorithms for allocating bandwidth to incoming streams. Bandwidth allocation is assumed to be *static* in the sense that the allocated bandwidth is fixed for the duration of the stream. In *dynamic* bandwidth allocation, each stream would be free to adjust its allocation (or to have it adjusted by the network controller) in response to varying system loads [KL94, CKLV95, PZF94, DHH+94]. This would require more intelligence on the part of the network controller and the stream server(s), and would require the video server to be able to adjust the bit-rate of the transmitted stream in real time. We therefore believe dynamic allocation to be less widely applicable than the static allocation problem, and do not consider it further here.

It is possible that in practice, systems would choose a hybrid approach in which streams would be forced to periodically renegotiate their bandwidth allocations in response to changing systems loads. Even in this case, using a good static allocation algorithm as a starting point could reduce the frequency with which these renegotiations would have to take place.

For video-on-demand, a typical usage profile may look like Figure 1. This plots the average number of streams as a function of the time of day. For VOD, a stream typically represents a single movie, so the duration of a stream might be anywhere from 1 to 3 hours. What Figure 1 really plots is the average number of active streams as a function of the time of

day, assuming each stream may be initiated without delay. (Another way of describing this usage pattern would be to plot the stream initiation probability as a function of the time of day.)

If the usage profile for each day were known exactly (by some oracle), then a simple approach to the bandwidth allocation problem would be to allocate each incoming stream the total available bandwidth divided by the number of streams from the profile. However, since actual usage may deviate significantly from the average (such as when a new movie is released, or even simply due to statistical fluctuations from the mean) and it is impossible to predict this exactly, we are forced to consider algorithms which adapt to the actual applied system load. Such algorithms may incorporate knowledge of the average usage profile, although they need not do so.

The general behavior of a static bandwidth allocation algorithm is assumed to be as follows. A user i makes a request to initiate a stream by providing to the VOD system or network controller, henceforth called the *controller*, a requested maximum bandwidth BW^i_{max} and a minimum acceptable bandwidth BW^i_{min}. The controller either responds with an available bandwidth BW_{avail}, where $BW^i_{min} \leq BW_{avail} \leq BW^i_{max}$, or denies the request if insufficient bandwidth is available. If the request is denied, the user is *queued* until another stream terminates, at which time the request is *retried*. When the request is finally granted, the user responds with an actual allocated bandwidth BW^i_{alloc}, where $BW^i_{min} \leq BW^i_{alloc} \leq BW_{avail}$, since the user may only be able to choose between discrete values of actual bandwidth used (1.5, 3 or 6 Mbit/s MPEG video compression, for example). The maximum number of potential users is denoted N, and the total available bandwidth BW_{total}. This discreteness in actual bandwidth allocated (BW^i_{alloc}) allows VOD to be modeled as a state-space system amenable to queuing analysis.

Also note that this three-stage negotiation process makes it possible to use such an algorithm on a *per link* basis in a multi-hop transmission network, for instance, or in other multiple-resource allocation problems: A user's initial request is concurrently sent to *all* resources r, and the bandwidth allocated BW^i_{alloc} is the *minimum* of all available $BW_{avail}(r)$ and any discrete constraint by the user. If any resource returns with $BW_{avail}(r) = 0$, the user is queued at that resource and the entire request is retried when r becomes available.

For this study, we consider the following algorithms:

MIN Each user is always granted the minimum request, i.e. $BW_{avail} = BW^i_{min}$.

FCFS As long as sufficient bandwidth remains, each user is granted the maximum, $BW_{avail} = BW^i_{max}$. Other users are queued, in first-come-first-served order, until another stream terminates.

DIV Like FCFS, but when a stream terminates, the freed bandwidth is divided evenly among queued users. Each queued user i is granted at least BW^i_{min}, and if all queued users cannot be accommodated, they are served in first-come-first-served order.

MAP Each incoming request is granted a fraction of its maximum BW^i_{max} which de-

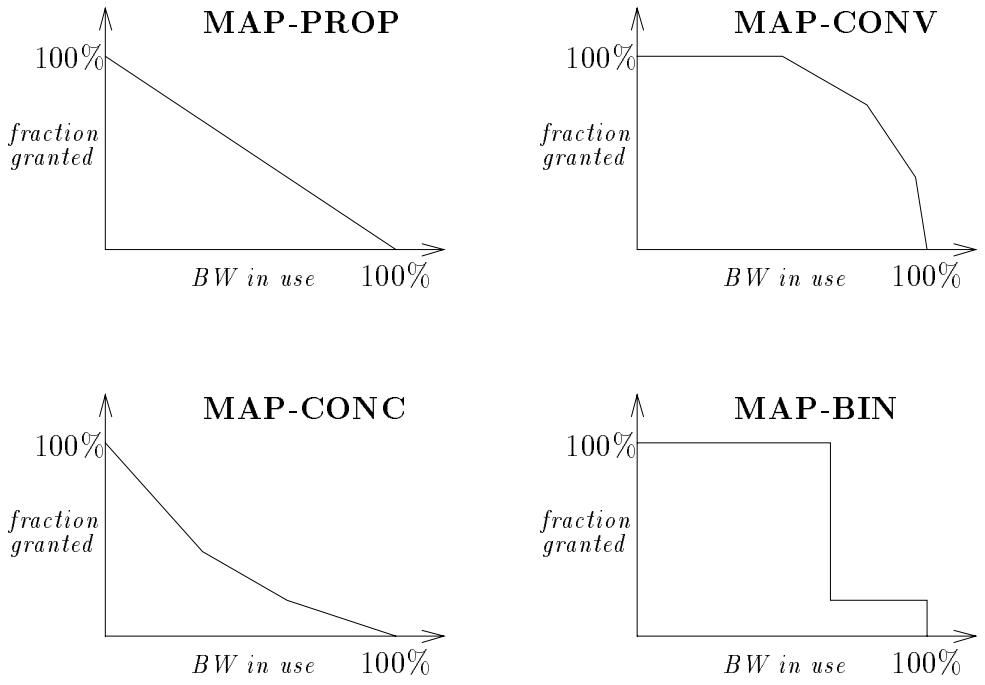

Figure 2: In-use v/s Granted bandwidth maps

pends on the fraction of bandwidth already in use BW_{use}, i.e. $BW_{avail} = \alpha BW^{i}_{max}$, where $\alpha = f(BW_{use})$. There are numerous possibilities for $f()$, such as a linear relation (MAP-PROP), a convex or concave function (MAP-CONV, MAP-CONC), or a binary-valued function (MAP-BIN), as shown in Figure 2. Our queueing analysis considers only discrete versions of DIV and MAP where the the only choices for BW_{alloc} are BW_{min} and BW_{max}, referred to as DIV-BIN and MAP-BIN.

3 Performance Metrics

There are several quantities of interest that will differentiate a good bandwidth allocation algorithm from a bad one. An ideal algorithm would have the following characteristics. During times of high usage, it would divide the available bandwidth evenly among all streams. During times of low usage, each user stream would receive its maximum bandwidth request. In all cases, no user would experience any delay between making a request and initiating a stream.

The latter quantity is the *queuing time* experienced by users. Although it is desired that this be zero at all times, it may be tolerable to allow for a small probability of non-zero queuing time, depending on how stringent the requirements on other metrics are. The fraction of

available bandwidth in use is the *utilization*, denoted v,

$$v = \frac{\sum_{i=1}^{N} BW_{alloc}^{i}}{BW_{total}}, \tag{1}$$

and should be kept as close to 100% as possible during times of high usage in order that each user receive the best fidelity possible. A *time of high usage* is a situation in which the total requested maximum bandwidth exceeds that available, i.e.

$$\sum_{i=1}^{N} BW_{max}^{i} > BW_{total} \tag{2}$$

and *low usage* is defined as the converse. *Saturation* is the situation in which the total of *minimum* bandwidths requested exceeds the total available:

$$\sum_{i=1}^{N} BW_{min}^{i} > BW_{total} \tag{3}$$

(Note that for inactive or *idle* users, $BW_{min}^{i} = BW_{max}^{i} = 0$ by definition.) During low usage, *relative utilization* $\hat{v}$ is defined as the fraction of total *maximum* bandwidth request actually used:

$$\hat{v} = \frac{\sum_{i=1}^{N} BW_{alloc}^{i}}{\sum_{i=1}^{N} BW_{max}^{i}} \tag{4}$$

The *offered load* on the system is a function of the arrival rate of stream requests, and the average duration (or equivalently, the *service rate*) of streams. Let λ denote the *per user* arrival rate, i.e., $1/\lambda$ is the average interval a user spends between making requests; and let $1/\mu$ denote the average duration of a stream. The offered load may be expressed as $\hat{N}$, the mean number of active streams in the absence of queuing. This can be expressed in terms of λ and μ, or equivalently in terms of the dimension less quantity $\rho = \frac{\lambda}{\mu}$ by

$$\hat{N} = N\frac{\frac{1}{\mu}}{\frac{1}{\mu} + \frac{1}{\lambda}} = N\frac{\lambda}{\lambda + \mu} = N\frac{\rho}{1 + \rho} \tag{5}$$

since on average, each of N users spends $\frac{1}{\mu}$ of every $\frac{1}{\mu} + \frac{1}{\lambda}$ cycles with an active stream. Assuming all users make identical BW_{min} and BW_{max} requests, the onset of high usage occurs when $\hat{N}$ exceeds the number of maximal streams the system can support, $M = \left\lfloor \frac{BW_{total}}{BW_{max}} \right\rfloor$.

We will be mostly interested in performance of the algorithms under high and low usage, since it is assumed that systems will be designed to avoid saturation. The next section will show that a variety of simple algorithms can provide the desired characteristics in the *steady state* (i.e. when the average request rate and stream characteristics do not change). We are also interested in their characteristics under *transient* conditions, such as during a change from low usage to high usage and vice versa. A good algorithm should avoid queuing delays and maintain high utilization and fair allocation of bandwidth, even during transient and unexpected load conditions.

4 Steady State Performance of Algorithms

In this section we outline simple arguments to elucidate the behavior of the algorithms in the steady state. These results are derived from [BM95].

Steady state here implies two things:

1. The arrival of stream requests is a random process (e.g., Poisson) with a *stationary* distribution with respect to time, as are the stream durations;
2. The onset of these distributions is sufficiently far in the past that the allocation algorithm itself has reached an equilibrium, statistically speaking, as measured by the distribution of current stream allocations.

4.1 MIN

Except in saturation, simply giving each user his minimum request will guarantee that queuing never occurs, since by (2) there is always available bandwidth. However, this also guarantees that bandwidth is always under-utilized since

$$v = \frac{\sum_{i=1}^{N} BW_{min}^{i}}{BW_{total}} \leq 100\%. \tag{6}$$

Furthermore, assuming that usually $BW_{min}^{i} \ll BW_{max}^{i}$, relative utilization will also be poor during low usage:

$$\hat{v} = \frac{\sum_{i=1}^{N} BW_{min}^{i}}{\sum_{i=1}^{N} BW_{max}^{i}} \ll 100\% \tag{7}$$

However, MIN does guarantee full utilization, fair bandwidth allocation, and minimum queuing delay during saturation.

4.2 FCFS

FCFS improves upon MIN during low usage, since relative utilization is 100%:

$$\hat{v} = \frac{\sum_{i=1}^{N} BW_{max}^{i}}{\sum_{i=1}^{N} BW_{max}^{i}} = 100\% \tag{8}$$

During high usage, however, queuing occurs since by definition (2) not all users can be accommodated. Instead of dividing the bandwidth fairly among all requesting users, FCFS queues some users, while giving others their maximum.

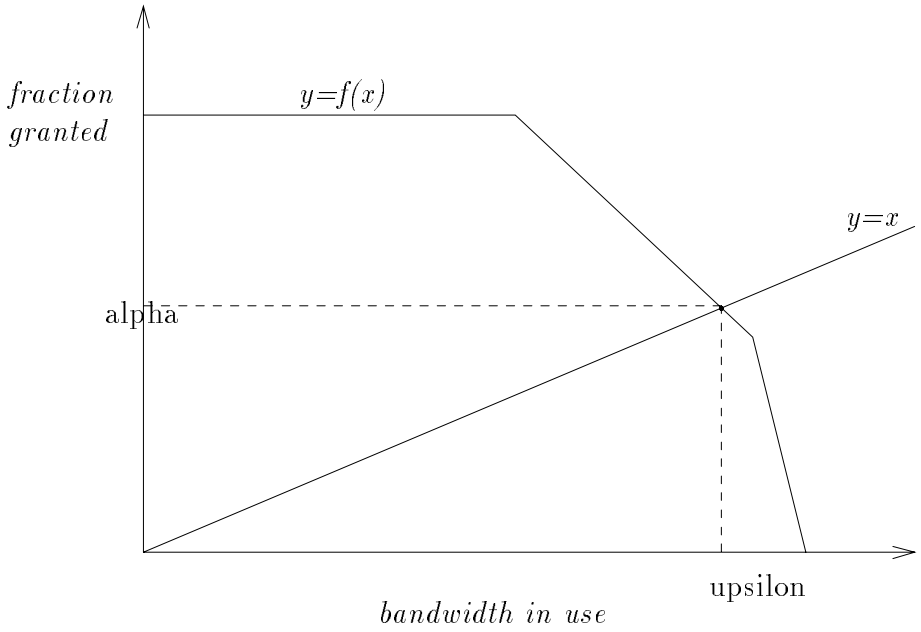

Figure 3: Solution for MAP utilization in steady state

4.3 DIV

DIV solves the fairness problem with FCFS during high usage. Although at the onset of high usage some queuing will occur, eventually some active streams will terminate, freeing their bandwidth to waiting users. Any bandwidth thus freed is divided as fairly as possible among waiting users, and eventually the bandwidth should become perfectly divided among all active users.

DIV still has two problems. Although steady-state bandwidth allocation is fair, some time is required for it to reach steady state, and during this time, significant queuing delays can be experienced by users. The algorithm is essentially *reactive* in that it adjusts its bandwidth allocations only *after* the onset of queuing. But even after steady state is reached, some queuing is likely to occur since the system is only able to accept new streams at the same rate that old streams terminate (on average).

4.4 MAP

The MAP algorithm provides a means of decreasing the bandwidth allocated in response to rising usage *immediately* instead of waiting for the onset of queuing. Consider the simplest form, MAP-PROP, in which the fraction of request granted is directly proportional to the fraction of available bandwidth remaining. During low usage, users get most of what they requested, and as usage increases the allocated bandwidths drop in response. As utilization reaches 100%, incoming streams are only allocated a small fraction of their BW^i_{max}, so the

Algorithm	low usage	high usage		
	$\hat{v}$	v	queuing?	fair?
MIN	$\ll 100\%$	$< 100\%$	no	yes
FCFS	100%	100%	a lot	no
DIV	100%	100%	yes	yes
MAP	$\rightarrow 100\%$	$< 100\%$	little	yes

Table 1: Scheduling Algorithm Characteristics

rate at which the onset of queuing occurs decreases.

If the onset of high usage is sudden, a significant number of streams may still remain with large BW^i_{alloc}, so the bandwidth allocation may be initially unfair since subsequent users will obtain smaller fractions of their requests. However, eventually these streams will terminate and the system should reach a steady state in which the bandwidth allocated to each new stream exactly equals the average bandwidth allocated to *all* streams. If $\hat{N}$ is the number of *active* users being serviced (also called the applied load) and BW_{max} is the same for all of these, the steady state is described by

$$f\left(\frac{\hat{N}\alpha BW_{max}}{BW_{total}}\right) = \alpha, \tag{9}$$

where α is the fraction of a new stream's requested bandwidth actually granted, and the function $f()$ is the mapping function as in Figure 2.

In (9), the numerator in the argument to $f()$ is the amount of bandwidth currently in use, which in turn determines the fraction allocated to incoming requests. The solution to (9) is clearly a *fixed-point* of the mapping function $f()$. (9) can be rewritten as

$$y = f(x) = \frac{BW_{total}}{\hat{N}BW_{max}}x. \tag{10}$$

(10) describes a straight line, and the intersection of y with $f()$ gives the steady-state solution, as depicted in Figure 3.

Note that in general, MAP does not achieve 100% utilization in steady state. The actual v achieved depends on the mapping function $f()$ and the *requested usage factor* $\phi = \hat{N}BW_{max}/BW_{total}$. While theoretically $v = 100\%$ is desirable, it may be useful in practice to leave some "guard bandwidth" to prevent queuing in response to sudden fluctuations in the load at high usage. The amount of guard bandwidth can be selected by choosing $f()$ appropriately.

Table 1 summarizes the characteristics of the various algorithms under low and high usage situations (excluding saturation).

5 Queuing Analysis

Simulation results for simple algorithms presented earlier (Section 2), are well documented [BM95]. Simulation gives a good insight into the performance of a system but is not a good tool for understanding system internals. One of the tools employed frequently for gaining insights into system dynamics is queuing analysis. Queuing theory has its limitations and cannot solve any arbitrary system but it is a very helpful tool in most cases.

For the purpose of queuing analysis, the system is viewed as a discrete state-space model. VOD systems are similar to Markov processes. However certain characteristics of VOD systems are very relevant to their state-space model.

- The total bandwidth to be allocated is fixed (BW_{total}).
- It is a *closed system*. This means that the system is designed to serve a known number of users. This maximum is known beforehand(N) and cannot be changed dynamically. This is different from M/M/1 systems in which there may be arbitrarily large number of users in the system at a given time.
- Each user can have at most 1 outstanding request at any given time.
- The modeling is on a per process basis. The properties of the system as a whole are state dependent. Hence it is no longer true to assume that the system has an arrival rate of λ (M/M/1) but it still makes sense to say that each process generates requests at the rate of λ.
- The service time per process is independent of the allocated bandwidth. VOD servers send compressed data to users having lower bandwidths. Hence a movie stream will terminate after the same interval from its starting time (length of the movie), independent of the quality of the video being delivered.

5.1 Algorithms

In this paper we look at two families of algorithms. Both of them are derived from combinations of simple algorithms discussed earlier. To make them suitable for this kind of analysis, an extra restriction was also applied, namely that requesting process can be allocated bandwidth only at two levels, a maximum bandwidth (BW_{max}) or a minimum bandwidth (BW_{min}). No bandwidth value other than the two may be allocated by the system. Thus their names have a trailing BIN (for binary).

The constraint of binary bandwidths is a reasonable one for VOD systems. As mentioned in the introduction, video servers store two or three versions of each movie on disk using hierarchical compression schemes such as MPEG-2. This allows them to provide selectable bit-rate streams to a new request depending on the state of system resources and the

network. The binary restriction makes the bandwidth allocation problem tractable by reducing an infinite state-space (in the continuous case) to a finite set (discrete bandwidths).

DIV-BIN

DIV-BIN is a combination of DIV (FCFS with fairness) and dual allocation levels. Thus for DIV-BIN

- Requests are serviced in FCFS order.
- The service discipline is similar to "Processor Sharing", with bandwidth being the critical resource.
- The system allocates bandwidth such that it satisfies the maximum number of users, with highest possible quality, as soon as possible. Hence if there are no queued users, each new user receives BW_{max}. If there is queuing, the queued users each receive BW_{min} (in FCFS order, as long as bandwidth remains) from the bandwidth released by a terminating process.

MAP-BIN

MAP-BIN is a family of algorithms which resemble DIV-BIN. It is however more flexible because it has an extra parameter, the cutoff threshold. The threshold is defined in terms of (as a percent of) the total network bandwidth (BW_{total}) and identifies the transition point from max (BW_{max}) to min (BW_{min}) bandwidth allocation.

- Requests are serviced in FCFS order.
- Bandwidth allocation at any time is decided by the level of usage. If the network is lightly loaded ($BW_{use} < Threshold * BW_{total}$) maximum bandwidth ($BW_{max}$) is allocated. Else the requesting process is granted minimum (BW_{min}) bandwidth.

We used 4 instances of MAP-BIN with cutoffs at 50% (MAP50-BIN), 75% (MAP75-BIN), 90% (MAP90-BIN) and 95% (MAP95-BIN) for simulation results.

DIV-BIN and $lim_{XX \to 100\%}$ MAPXX-BIN give very similar results for most cases. However, since MAP-BIN does not try to satisfy the maximum number of users, their state-space differ.

5.2 Queuing Model

If we assume that stream durations are exponentially distributed, a Markov state space of BIN algorithms can be defined by the number of users in various modes (N_{max}, N_{min},

N_{queued}, N_{idle}). Since it is a closed system we have the relation

$$N_{max} + N_{min} + N_{queued} + N_{idle} = N \tag{11}$$

N_{total} is known before designing the system, hence the equation has only 3 free variables. We chose them to be N_{max}, N_{min} and N_{queued}. So the state space is a 3 dimensional surface, the axes being the # users with BW_{max} streams (N_{max}), with BW_{min} streams (N_{min}) and queued (N_{queued}) mode.

Transition Probability Matrix

To get an analytical solution to the model, we need to construct a system of linear equations. This system of simultaneous equations should describe the following :

- Valid (reachable from the initial state) states in the system.
- Possible transitions between these valid states.
- The rates at which these transitions occur.

The probability of change from a state (S_i) is the product of the probability of being in that state ($p(S_i)$) and the rate of change from that state.

Consider a particular valid state S_i. Let the probability of the system being in state S_i be $p(S_i)$. With respect to S_i, three events of significance exist

- System was in S_i and then changed state because of the arrival of a new request or the completion of a current request.

$$\nabla p(S_i) = p(S_i) * (rate\_of\_arrival + rate\_of\_departure)$$

$$\nabla p(S_i) = p(S_i) * (\lambda * N_{idle} + \mu * N_{busy}) \tag{12}$$

- System was in some state S_j and then changed to S_i because of the arrival of a new request.

$$\nabla p(S_i) = \sum_{S_j \xrightarrow{new\_request} S_i} p(S_j) * (rate\_of\_arrival)$$

$$\nabla p(S_i) = \sum_{S_j \xrightarrow{new\_request} S_i} p(S_j) * (\lambda * N_{idle}) \tag{13}$$

- System was in some state S_k and then changed to S_i because of the completion of a current stream.

$$\nabla p(S_i) = \sum_{S_k \xrightarrow{stream\_completes} S_i} p(S_k) * (rate\_of\_departure)$$

$$\nabla p(S_i) = \sum_{S_k \xrightarrow{stream\_completes} S_i} p(S_k) * (\mu * N_{busy}) \tag{14}$$

For each state (S_i) in the state space, net rate of change is the difference between the arrival and departure rates.

$$\nabla p(S_i) = (13) + (14) - (12) \tag{15}$$

Intuitively, this system of equations represents the rate at which the VOD system changes. It is thus the derivative of the probability of the VOD system being in a particular state. This system of linear simultaneous equations can also be written in matrix form as

$$\nabla P = A * P \tag{16}$$

where

A : Transition Rate Matrix (size : n x n)
P : Probability Vector (size : n)
* : Matrix Vector multiplication
n : # valid points in the state-space

The matrix multiplication adds up all incoming and outgoing change probabilities for a particular state, effectively computing the instantaneous rate at which a VOD system in that state would change.

5.3 Steady State Solution

The steady state for our (or for any) model exists when the rate of change (derivative) is 0. So to get the steady state probability distribution for our VOD system we need to solve the following equation

$$\nabla P = \vec{0}$$

which is equivalent to solving

$$A * P = \vec{0}^{1} \tag{17}$$

for P with the additional constraints that:

[1] $\vec{0}$ is a column vector of 0's : size n

- All probabilities are positive ($p_i \geq 0, \forall i = 1$ to n)
- The probabilities sum to 1 ($\sum_{i=1}^{n} p_i = 1$)

In general, A is a sparse matrix and sparse solvers can be used for solving this system of equations efficiently.

6 Results

We modeled the state-space and transition probability matrix (A) for DIV-BIN, MAP50-BIN, MAP75-BIN, MAP90-BIN and MAP95-BIN. We also did a discrete event simulation of these algorithms.

The parameters used for both of them were

- BW_{total} = 622 Mbit/s (ATM Bandwidth)
- BW_{max} = 12 Mbit/s (lossless compression)
- BW_{min} = 1.5 Mbit/s (MPEG compression)
- $N = 250$
- $\hat{N}$ = 10 to 150 in increments of 10
- $\lambda = 1$

•Queuing Model

The system of equations was solved by using *Sparse*[2], a sparse linear equation solver [KSV88]. Figure 4 shows an example of the probability distribution for the system when both the rate of arrival (λ) and rate of departure (μ) are the same (i.e $\hat{N} = \frac{N}{2}$).

The steady state probability distribution produced by the Sparse solver is used to extract various behavior characteristics for the algorithm. Figures 5 through 17 present some of the interesting results.

•Discrete Event Simulation

The PARSIM discrete event simulator [VCBB91] was used to simulate the same system as the queuing system had modeled. We ran the the simulation for 100000 simulated seconds (roughly 1 day). The same simulation results as those for the queuing model solution are presented in figures 6 through 18.

[2]Sparse, Version 1.3a,

- Developed at Department of Electrical Engineering & Computer Sciences , University of California, Berkeley by Kenneth S. Kundert & Alberto Sangiovanni-Vincentelli

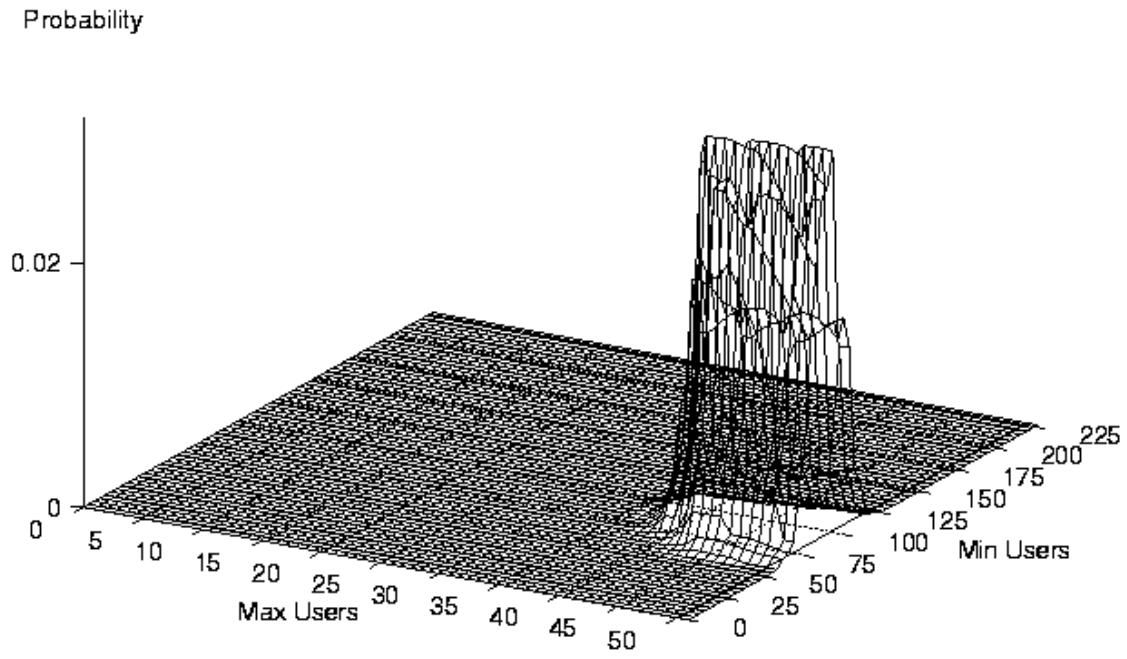

Figure 4: State-Space Probability Distribution ($\lambda = 1, \mu = 1$)

We see that under moderate load steady-state both DIV-BIN and MAP-BIN can keep all requesting users supplied with a stream [Figures 5 and 6]. For all the algorithms, max users (N_{max}) increase linearly in the beginning and then decrease steadily after a threshold is reached [Figures 7 and 8]. Similarly, we have very few min users (N_{min}) initially and they grow linearly later on [Figures 9 and 10]. DIV-BIN has a much worse queuing behavior than the MAP-BIN family [Figures 11 and 12]. There is almost no queuing for any of the MAP-BIN algorithms. Average utilization provides the most interesting feature in this comparison. While DIV-BIN tries to achieve the maximum bandwidth utilization, MAP-BIN algorithms seem to stabilize around their respective cutoff percentages [Figures 13 and 14]. This feature of MAP-BIN is an asset and we will discuss it in detail in the next section.

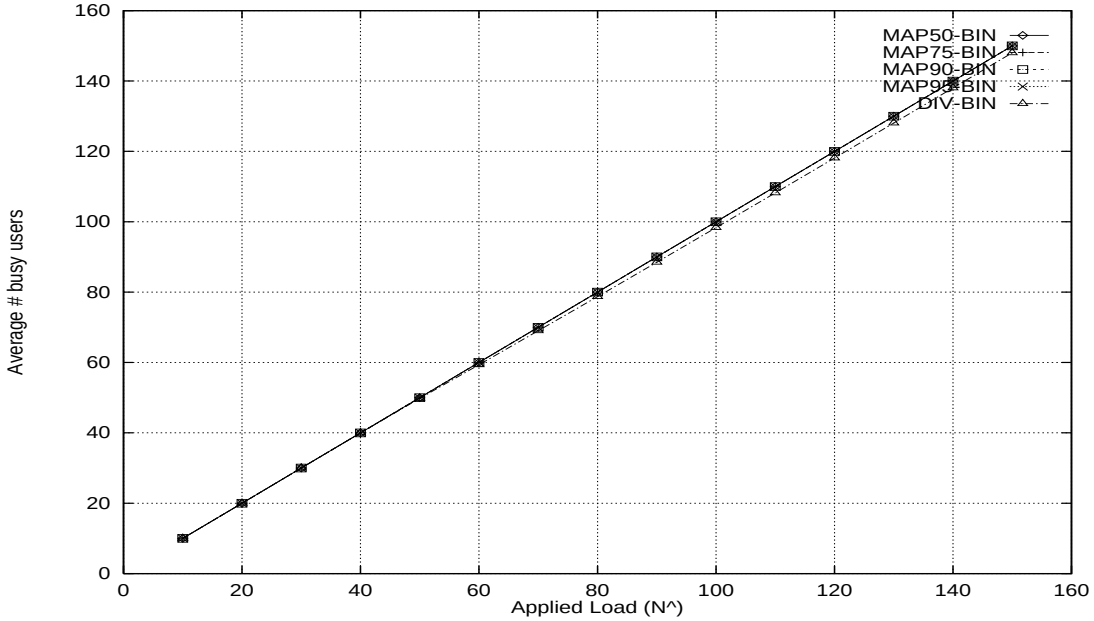

Figure 5: Model: Average Busy Users vs. Applied Load

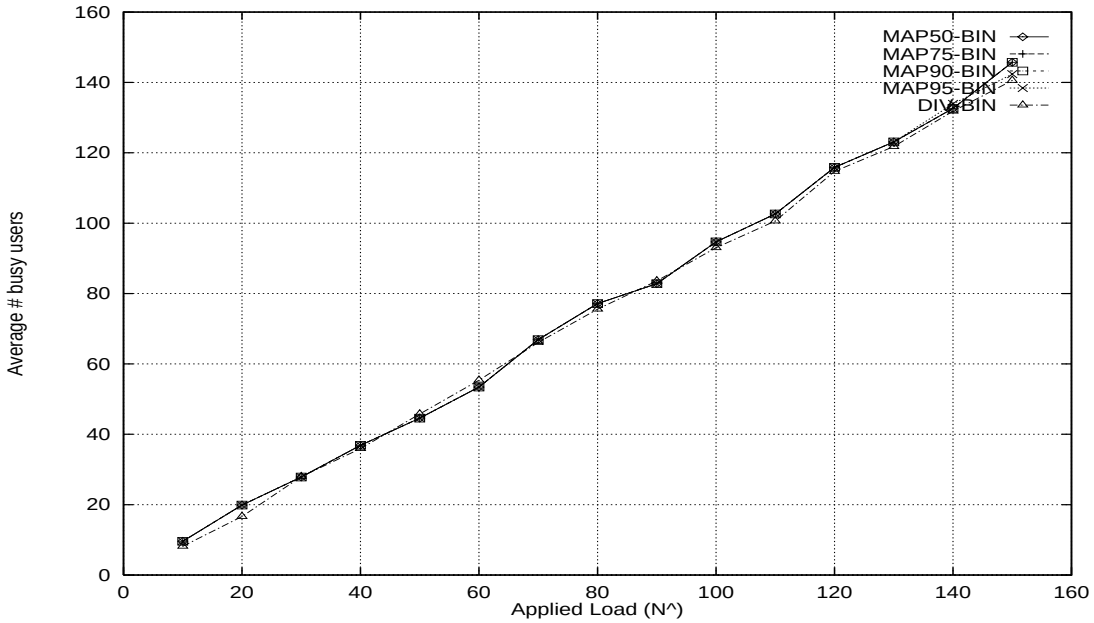

Figure 6: Simulation: Average Busy Users vs. Applied Load

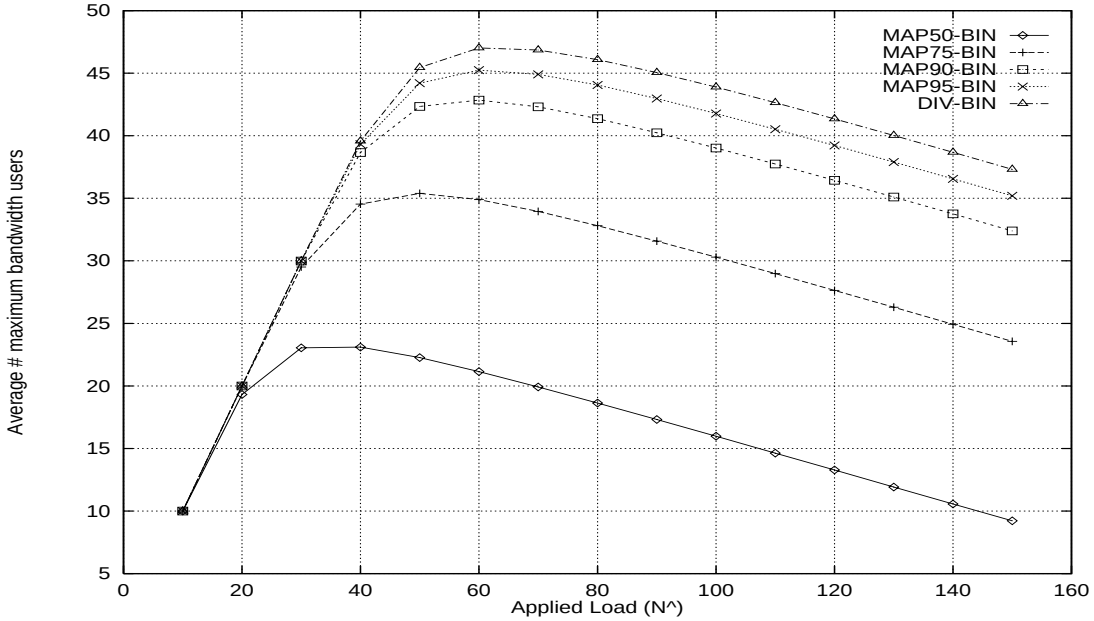

Figure 7: Model: Average Max Users vs. Applied Load

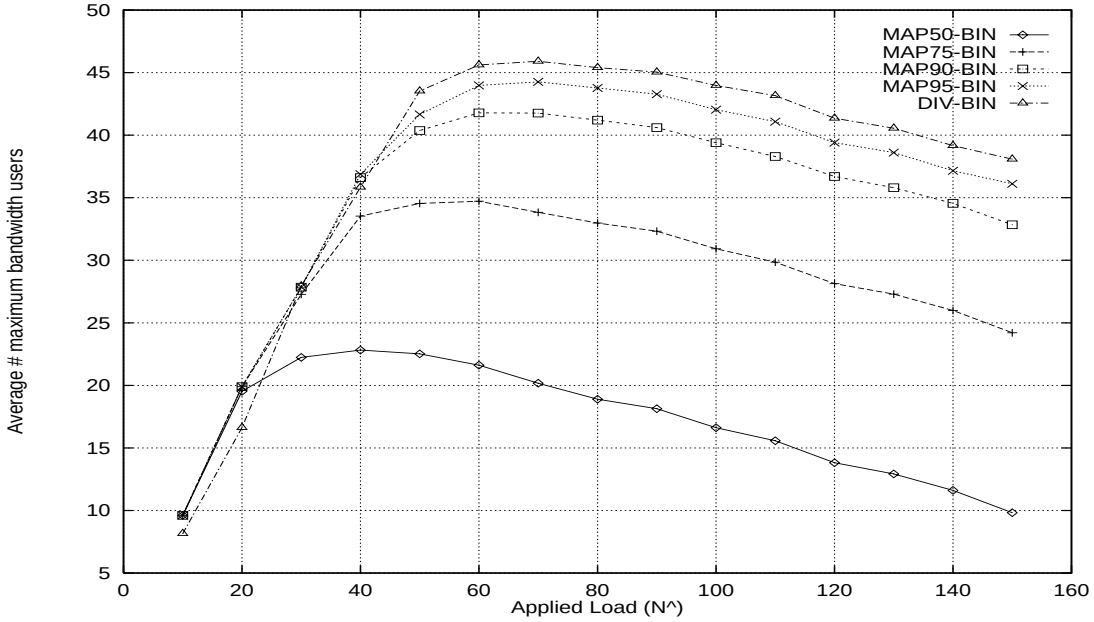

Figure 8: Simulation: Average Max Users vs. Applied Load

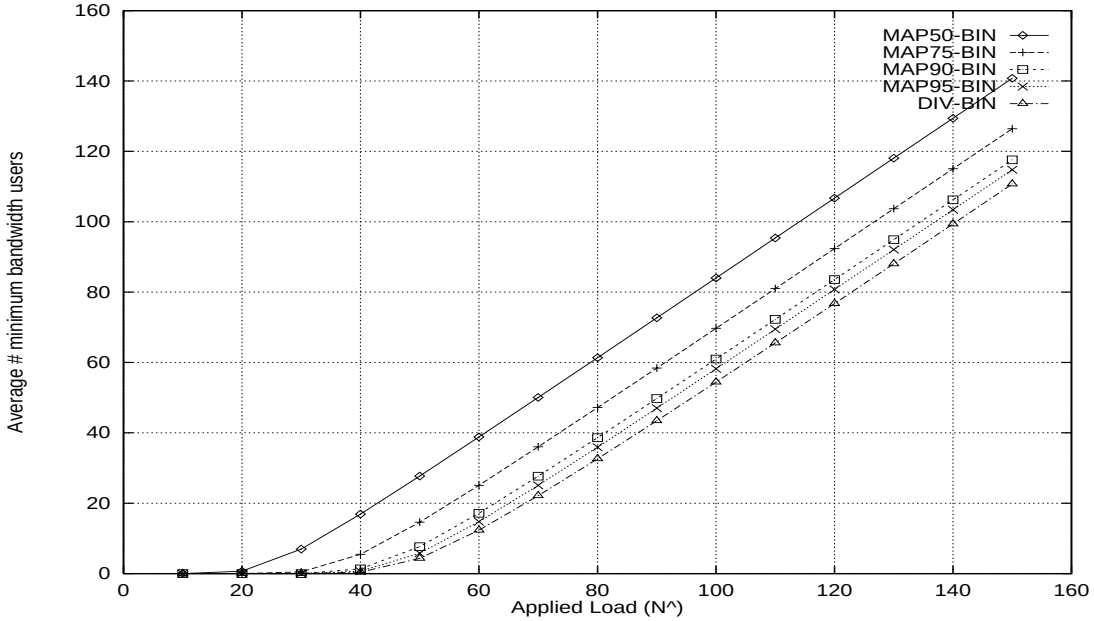

Figure 9: Model: Average Min Users vs. Applied Load

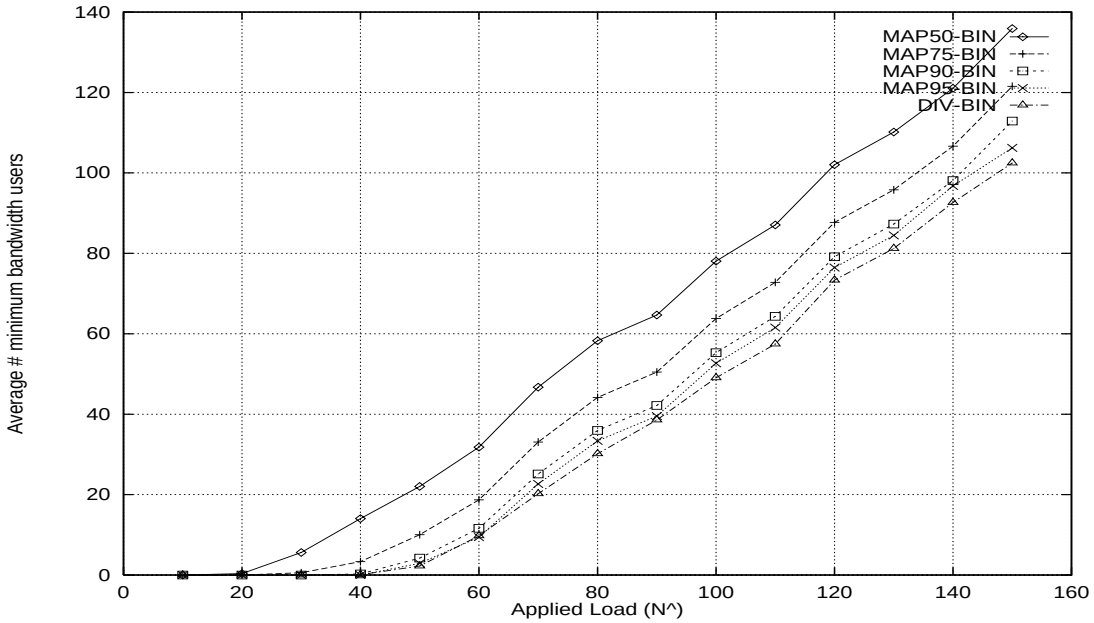

Figure 10: Simulation: Average Min Users vs. Applied Load

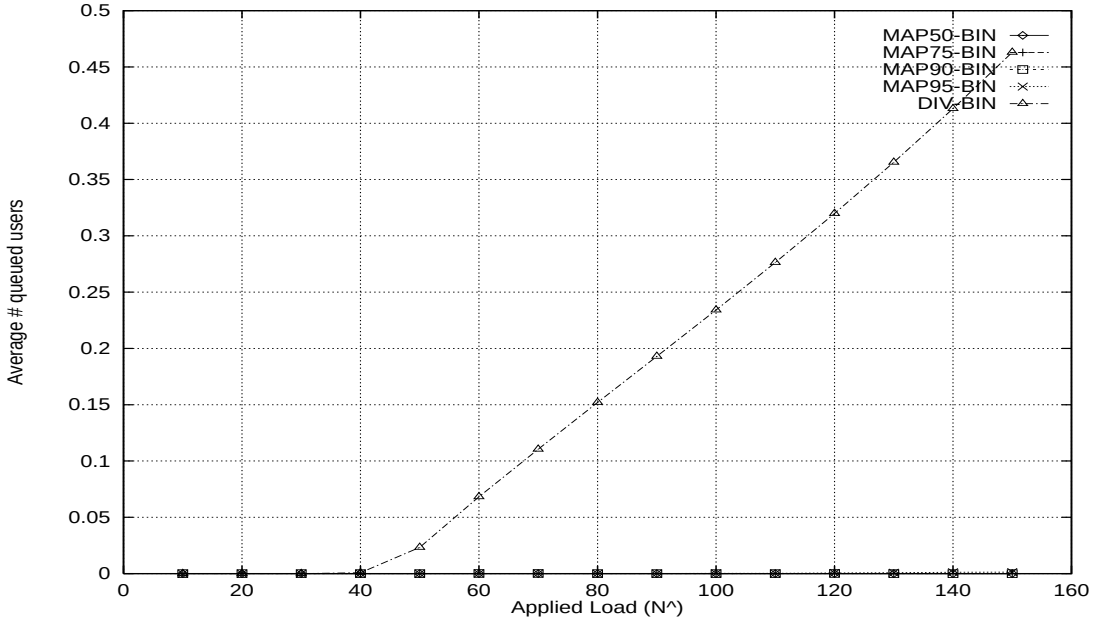

Figure 11: Model: Average Queued Users vs. Applied Load

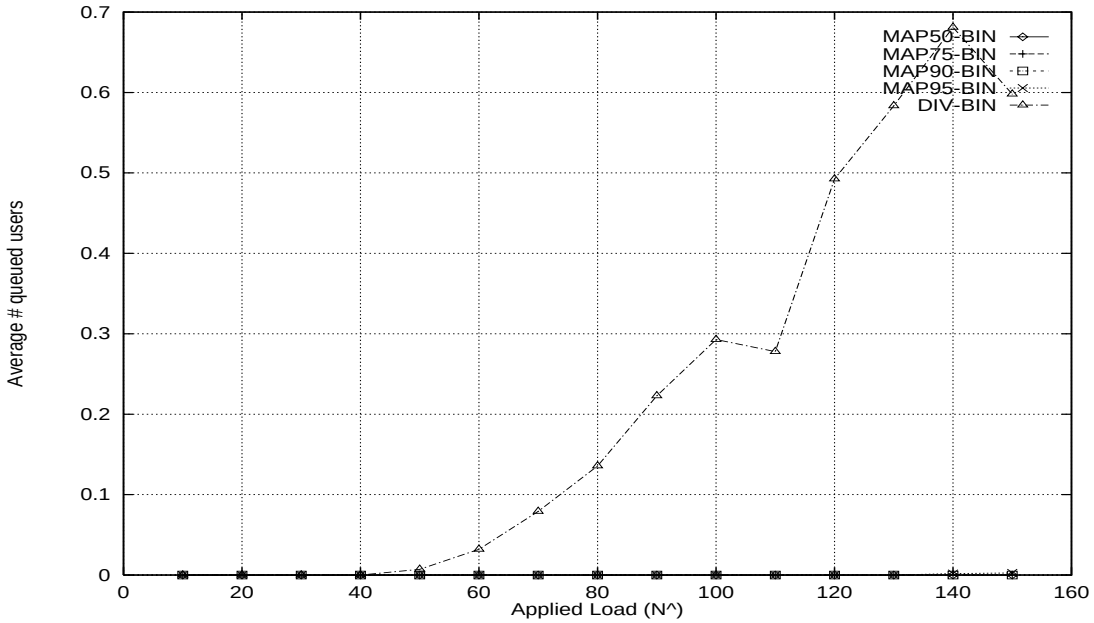

Figure 12: Simulation: Average Queued Users vs. Applied Load

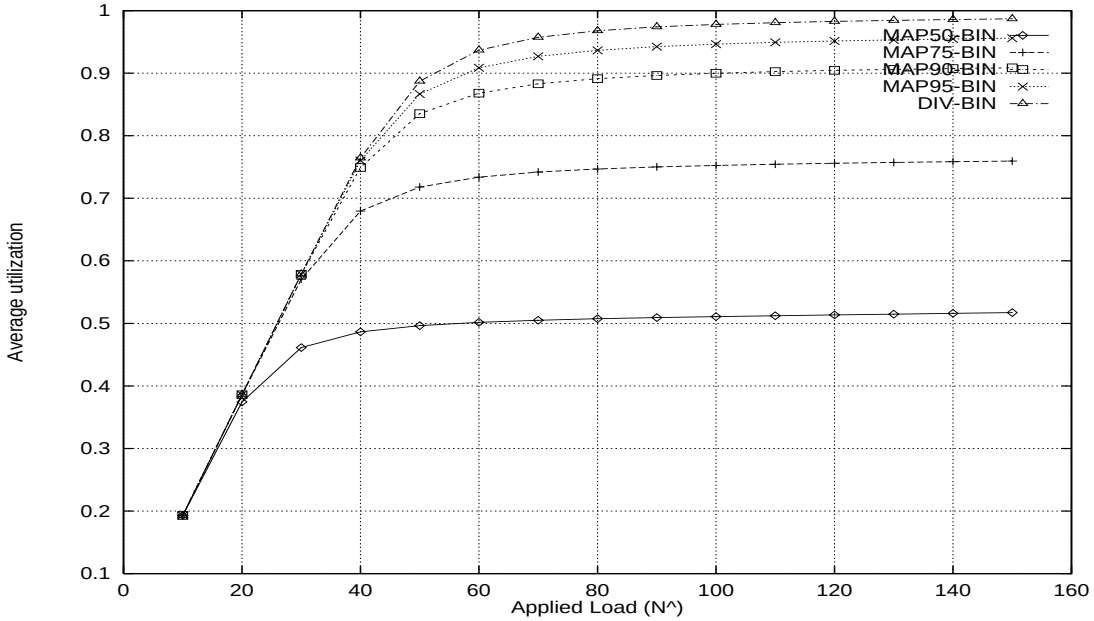

Figure 13: Model: Average Utilization vs. Applied Load

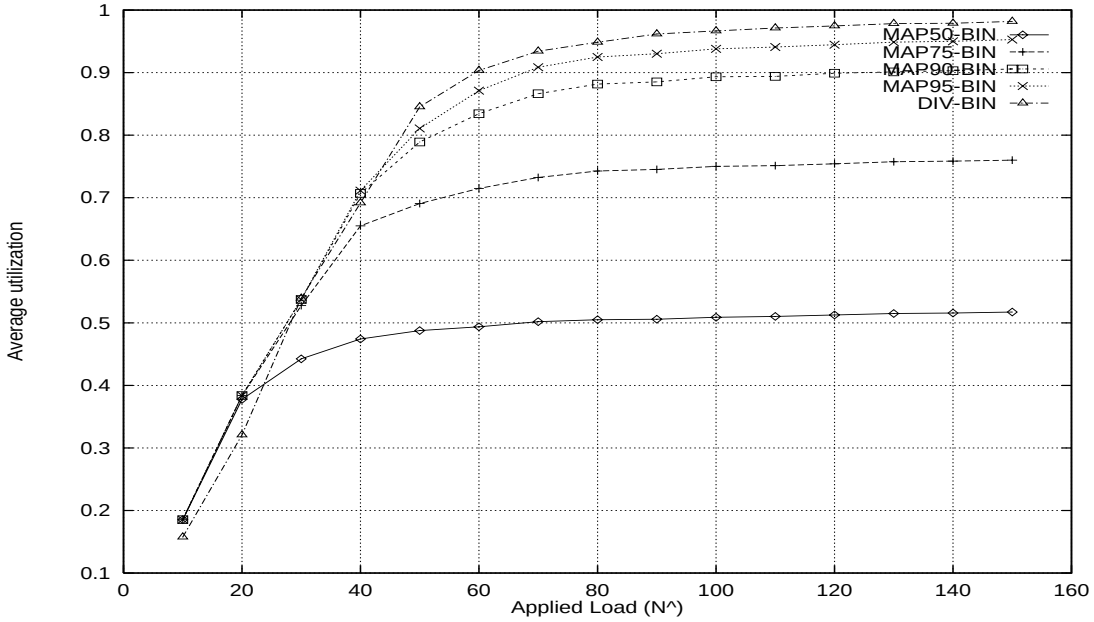

Figure 14: Simulation: Average Utilization vs. Applied Load

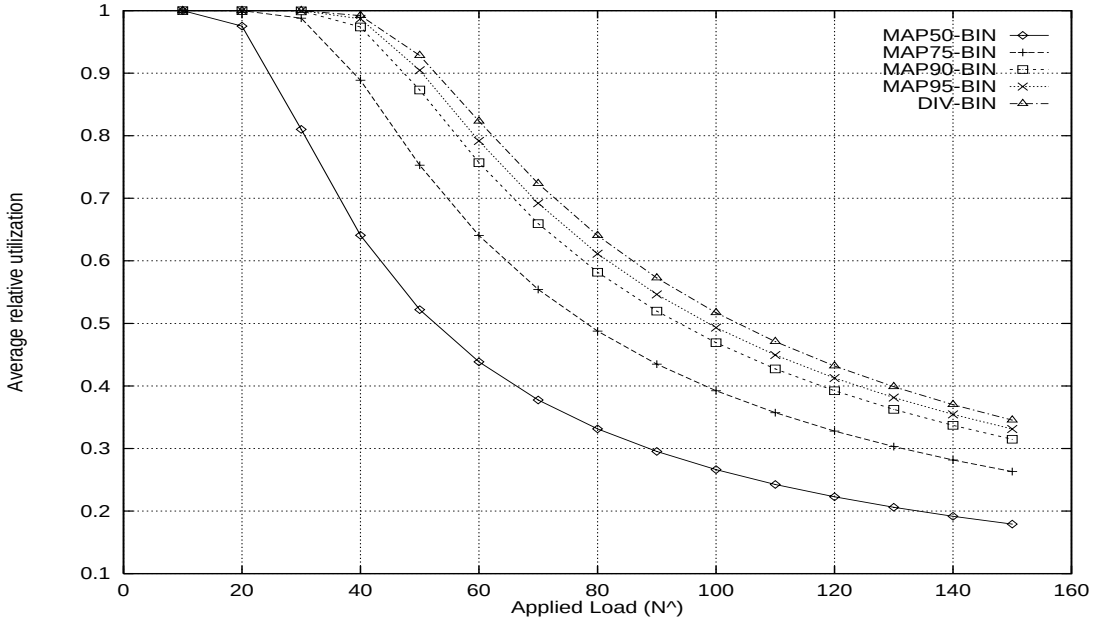

Figure 15: Model: Average Relative Utilization vs. Applied Load

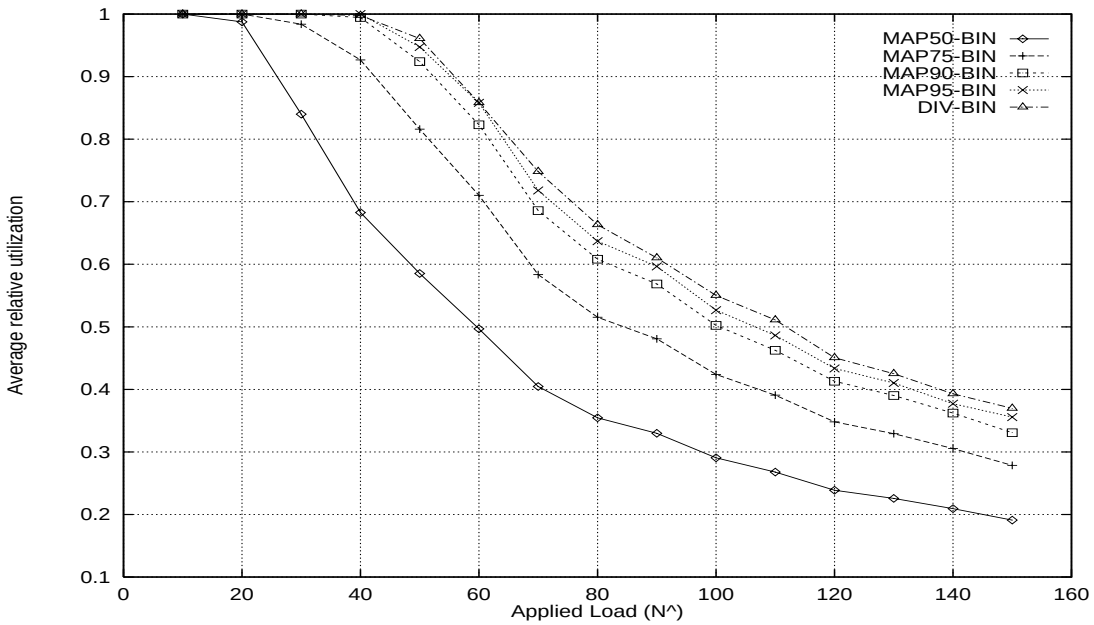

Figure 16: Simulation: Average Relative Utilization vs. Applied Load

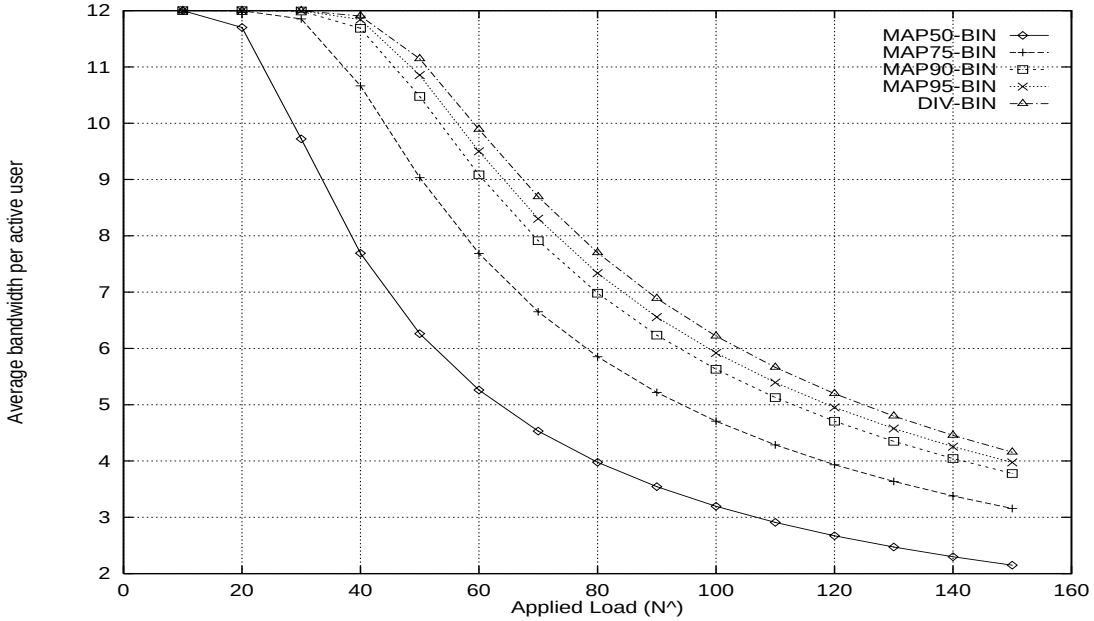

Figure 17: Model: Average Bandwidth per User vs. Applied Load

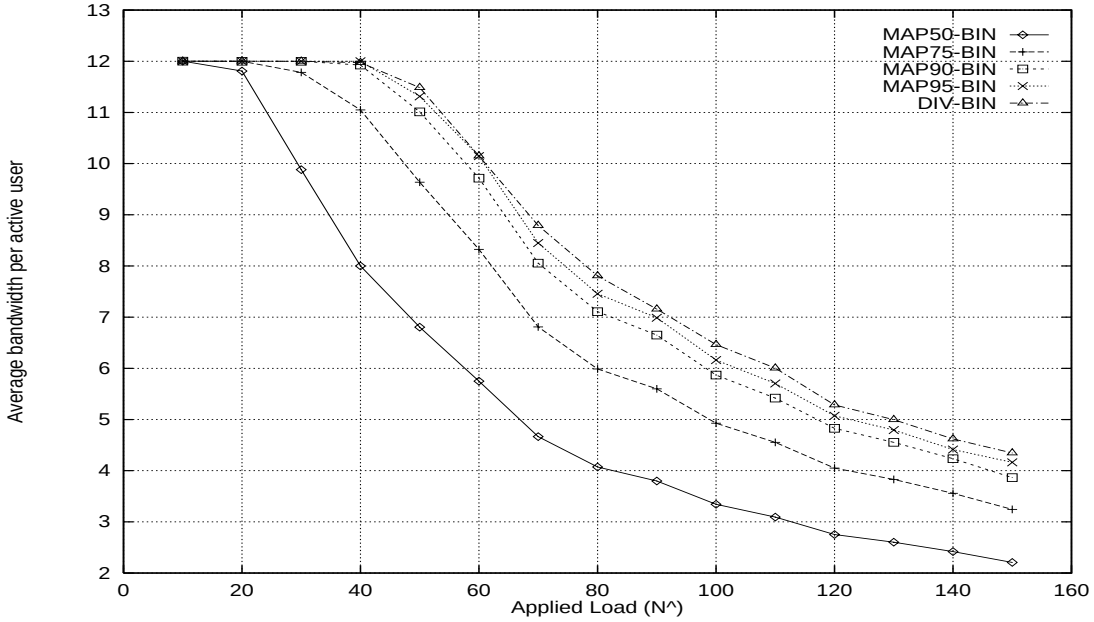

Figure 18: Simulation: Average Bandwidth per User vs. Applied Load

7 Analysis of MAP-BIN

MAP-BIN lends itself to some interesting mathematical analysis.

$$\hat{N} = N\frac{\lambda}{\lambda+\mu} = N_{max} + N_{min} + N_{queued}$$

$$N_{busy} = N_{max} + N_{min}$$

$$\Rightarrow \hat{N} = N_{busy} + N_{queued} \tag{18}$$

for any $\hat{N}$, the average number of busy users at *steady state* will be

$$N_{busy} \leq min\,(\hat{N}\,,\,\frac{BW_{total}}{BW_{min}}) \tag{19}$$

In most of the cases (as in our solution), for reasonable system load $N_{busy} = \hat{N}$ [Figures 5 and 6].

MAP-BIN uses a cutoff bandwidth (BW_{cutoff}) for deciding allocation

$$BW_{cutoff} = BW_{total} * cutoff\_percentage \tag{20}$$

For easier understanding we analyze our VOD separate under three different load conditions

7.1 Low Usage ($\hat{N} \leq \frac{BW_{cutoff}}{BW_{max}}$)

Under light load conditions all processes can be allocated maximum bandwidths. This is borne out by the results because we see the number of max users rising linearly with $\hat{N}$ [Figures 7 and 8]. However as we begin to approach the moderately loaded region ($\hat{N} = \frac{BW_{cutoff}}{BW_{max}}$) we start seeing some deviation. This, we believe, is due to the dynamic nature of the system. Since bandwidth allocation is dependent at the bandwidth usage at time of service, the order in which requests arrive and are serviced changes the allocation patterns. We see a graceful curve leading the number of maximum users into the next phase. In general the performance can be modeled as

- $N_{min} = 0$
- $N_{queued} = 0$
- $N_{max} = N_{busy} = \hat{N}$

7.2 Moderate Usage ($\frac{BW_{cutoff}}{BW_{max}} < \hat{N} < \frac{BW_{total}}{BW_{min}}$)

Since a moderately loaded VOD system can accommodate all users ($\hat{N} < \frac{BW_{total}}{BW_{min}}$), there is no queuing ($N_{queued} = 0$).

$$N_{queued} = 0 \Rightarrow \hat{N} = N_{busy} \Rightarrow \hat{N} = N_{max} + N_{min} \tag{21}$$

Since the VOD system is in steady state, for the MAP-BIN algorithms this implies that

$$BW_{cutoff} \leq BW_{inuse} \leq BW_{cutoff} + BW_{max} \tag{22}$$

If this isn't true, then a terminating maximum or minimum bandwidth stream will not be replaced by a similar one. This violates our assumption of a steady state solution.

Thus

$$BW_{inuse} = N_{max} * BW_{max} + N_{min} * BW_{min} \tag{23}$$

$$\Rightarrow BW_{cutoff} \leq N_{max} * BW_{max} + N_{min} * BW_{min} \leq BW_{cutoff} + BW_{max} \tag{24}$$

$$\Rightarrow BW_{cutoff} \leq N_{max} * BW_{max} + (\hat{N} - N_{max}) * BW_{min} \leq BW_{cutoff} + BW_{max} \tag{25}$$

$$\Rightarrow \frac{BW_{cutoff} - \hat{N} * BW_{min}}{BW_{max} - BW_{min}} \leq N_{max} \leq \frac{BW_{cutoff} + BW_{max} - \hat{N} * BW_{min}}{BW_{max} - BW_{min}} \tag{26}$$

$$\Rightarrow N_{max} \approx \frac{BW_{cutoff} + \frac{BW_{max}}{2} - \hat{N} * BW_{min}}{BW_{max} - BW_{min}} \tag{27}$$

$$\Rightarrow N_{max} \approx \frac{\frac{BW_{cutoff}}{BW_{min}} + \frac{\zeta}{2} - \hat{N}}{\zeta - 1} \tag{28}$$

where $\zeta = \frac{BW_{max}}{BW_{min}}$ i.e the ratio of the binary bandwidth levels.

Since we know N_{max} we pretty much know the steady state performance of the whole VOD system as a function of $\hat{N}$ because

- $N_{min} = \hat{N} - N_{max}$
- $N_{queued} = N_{idle} = 0$

7.3 High Usage ($\hat{N} > \frac{BW_{total}}{BW_{min}}$)

For reasonable parameter values ($BW_{max} \ll BW_{total}$ and cutoff_percent $< 100\%$) for a heavily loaded system

- $N_{max} = 0$

- $N_{min} = N_{busy} = \frac{BW_{total}}{BW_{min}}$
- $N_{queued} = \hat{N} - N_{busy}$

Thus MAP-BIN is accurately modeled by our mathematical model and this has been confirmed by cross-checking with the values available through the sparse solver.

This discussion also leads us to another interesting characteristic of MAP-BIN algorithms. We used this characteristic as a central assumption [equation 22] in our mathematical model and justified it using the steady state argument. All MAP-BIN algorithms, under low and moderate load conditions ($\hat{N} < \frac{BW_{total}}{BW_{min}}$) , limit bandwidth utilization to their cutoff bandwidth (actually $BW_{cutoff} + BW_{max}$). This was also verified by both the queuing model and simulation results [Figures 13 and 14]. This makes MAP-BIN algorithms very useful tools in controlling network usage where VOD might be sharing bandwidth with other applications (like telephones or normal Internet traffic).

8 Conclusions and Future Work

We have presented a framework for the allocation of bandwidth in multimedia networks with binary bit-rate compression. Queuing analysis is a powerful method that was adapted successfully to analyze VOD systems. We have compared two families of simple algorithms for allocating bandwidth to streams, with the objectives of providing the highest quality of service to each stream while minimizing the rejection rate of stream initiation requests. It has been shown that that simple algorithms, easily implemented in low-level network software or firmware, can achieve good steady-state behavior on simulated workloads. The MAP-BIN algorithms can also be used to limit network usage successfully for shared networks.

In future work, we will examine modeling of transient behavior for these algorithms. We will also like to examine the performance of these algorithms in multi-hop networks.

References

[And93] David P. Anderson. Metascheduling for continuous media. *ACM Transactions on Computer Systems*, 11(3):226–252, August 1993.

[BM95] Carl Beckmann and Ahmed Moin. Bandwidth reservation with selectable bit-rate streams. Technical report, Thayer School of Engineering, Dartmouth College, June 1995.

[Bru94] Craig J. Brunet. Hybridizing the local loop. *IEEE Spectrum*, 31(6):28–32, June 1994.

[CAGM94] Navin Chaddha, Avneesh Agrawal, Anoop Gupta, and Teresa H. Y. Meng. Variable compression using JPEG. In *Proceedings of the International Conference on Multimedia Computing and Systems*, pages 562–569, May 1994.

[CKLV95] H. J. Chen, A. Krishnamurty, T. D. C. Little, and D. Venkatesh. A scalable video-on-demand service for the provision of VCR-like functions. In *Proceedings of the International Conference on Multimedia Computing and Systems*, Washington, D.C., May 1995. IEEE Computer Society Press.

[DHH+94] Luca Delgrossi, Christian Halstrick, Dietmar Hehmann, Ralf Guido Herrtwich, Oliver Krone, Jochen Sandvoss, and Carsten Vogt. Media scaling in a multimedia communication system. *Multimedia Systems*, 2:172–180, 1994.

[Gal91] D. Le Gall. MPEG: A video compression standard for multimedia applications. *Communications of the ACM*, 34(4):46–58, April 1991.

[Jul94] Mark Juliano. ATM traffic control. *Byte Magazine*, pages 129–134, December 1994.

[KL94] A. Krishnamurthy and T.D.C. Little. Connection-oriented service renegotiation for scalable video delivery. In *Proceedings of the International Conference on Multimedia Computing and Systems*, pages 502–507, May 1994.

[KSV88] Kenneth S. Kundert and Alberto Sangiovanni-Vincentelli. Sparse user's guide, a sparse linear equation solver, version 1.3a. Department of Electrical Engineering & Computer Sciences, University of California, Berkeley, April 1988.

[Lan94] James Lane. ATM knits voice and data on any net. *IEEE Spectrum*, 31(2):42–45, February 1994.

[Mil94] Arthur Miller. From here to ATM. *IEEE Spectrum*, 31(6):20–24, June 1994.

[Nol95] P. Noll. Digitak audio coding for video communications. *Proceedings of the IEEE*, 83(6):925–943, June 1995.

[PZ94] Pramod Pancha and Magda El Zarki. MPEG coding for variable bit rate video transmission. *IEEE Communications Magazine*, May 1994.

[PZF94] Colin Parris, Hui Zhang, and Domenico Ferrari. Dynamic management of guaranteed-performance multimedia connections. *Multimedia Systems*, 1:267–283, 1994.

[RC95] Reza Rooholamini and Vladimir Cherkassky. ATM-based multimedia servers. *IEEE Multimedia*, pages 39–52, March 1995.

[SS94] Khosrow Sohraby and Moshe Sidi. On the performance of bursty and modulated sources to leaky bucket rate-based access control schemes. *IEEE Transactions on Communications*, 42(2):477–487, February 1994.

[SS95] R. Schafer and T. Sikora. Digital video coding standards and their role in video communications. *Proceedings of the IEEE*, 83(6):907–924, June 1995.

[VCBB91] Alex Veidenbaum, Hoichi Cheong, John Bruner, and Carl Beckmann. Ftp source code for parsim/carl. `ftp://ftp.csrd.uiuc.edu/pub/beckmann/Parsim.tar.Z`, 1991.

[Yos94] Tadao Yoshida. The rewritable MiniDisc system. *Proceedings of the IEEE*, pages 1490–1500, October 1994.