Dynamic Resource Allocation Based on Measured QoS*

Technical Report TR 96-2
Center for Advanced Computing and Communication
North Carolina State University, Raleigh.

Sanjeev Rampal
Network Architecture
IBM
Research Triangle Park, NC 27709
email: srampal@vnet.ibm.com

Douglas S. Reeves
Department of Computer Science
North Carolina State University
Raleigh, NC 27695
email: reeves@csc.ncsu.edu

Ioannis Viniotis
Department of Electrical and Computer Engineering
North Carolina State University
Raleigh, NC 27695
email: candice@eos.ncsu.edu

*This work was supported by the Air Force Office of Scientific Research Under Grant F49620-92-J-0441.

Abstract

Most methods for guaranteeing quality of service (QoS) in packet-switched networks require a static characterization of the user's traffic, and allocate resources accordingly. This process is inconvenient for unsophisticated users, and can result in overallocation of resources. We address the problem of automatically determining the minimal resources necessary to satisfy a specified Quality of Service (QoS) measure using dynamic techniques. Specifically, we study a dynamic control algorithm which determines the minimum bandwidth needed to satisfy a specified average cell loss probability requirement for a given source. This algorithm (referred to as **REQS** for **R**esource-**E**fficient **Q**uality of **S**ervice) measures the actual cell losses, and uses this information to dynamically vary the bandwidth allocation. Experimental results show the algorithm converges quickly to the desired solution. The effect of the measurement frequency, source burstiness, buffer size and loss specification on the convergence time of the algorithm are all investigated. The algorithm is found to be robust, converging quickly and accurately under most conditions. We demonstrate the bandwidth savings attainable by this approach over other techniques such as the equivalent capacity formulas. The applicability of the method for other QoS measures (such as queuing delay percentiles) is also demonstrated. Some of the applications of this technique include control of rate-enforcing servers, traffic shapers and call admission control in ATM networks.

Keywords: Quality of Service, Dynamic Resource Allocation, Equivalent Capacity

1 Introduction

Networks supporting multimedia traffic are expected to satisfy certain Quality-of-service (QoS) performance measures for each user. Additionally, efficient use of network resources is desired for providing a cost-effective service. In the context of ATM networks, typical QoS measures include average cell loss probability (CLP), cell transfer delay (CTD) and cell delay variance (CDV) [1]. Network resources of interest include bandwidth requirements at each physical link, as well as buffer space and processing capability at each switching node. A major issue for the network service provider is how to satisfy the specified QoS requirements using the network resources as efficiently as possible.

The problem of providing QoS guarantees without being excessively conservative in utilizing resources is difficult for variable bit rate (VBR) sources. For such sources, the traffic offered to the network is "bursty" and somewhat unpredictable. As an example, a video codec may generate data at a time-varying rate depending on the (unpredictable) video content. This does not, however, eliminate the need for guarantees on the end-to-end delay and losses when the video is transmitted across the network. Table 1 lists typical data rates and QoS requirements for voice and video traffic [19]. Note that the peak data rate is several times the mean rate which gives some idea of statistical variability of traffic generation rates for such sources.

The conventional approach for dealing with this problem is to have each user declare in advance the characteristics of their traffic. As an example, the ATM Forum has adopted a standard set of 3 traffic descriptors: peak cell rate, sustainable cell rate and maximum burst size [1]. A policing mechanism such as the leaky bucket [1] will ensure that the user's traffic conforms to the declared values for these descriptors. Using this information, an appropriate Call Admission Control (CAC) algorithm determines whether the QoS requirements of a new call can be met, without compromising the QoS of calls that have already been admitted.

There are problems with using this approach, including the following:

- Users have to be able to characterize their traffic generation process reasonably accurately; this is not always feasible. While a video sequence generated off-line can be analyzed very thoroughly, a user initiating a video conference may have very little idea of the expected traffic statistics.
- Even if users have the ability to model the traffic generation process, a small number of traffic

	Peak Rate	Mean Rate	Acceptable Cell Loss probability
Voice	32 KBits/sec	11.2 KBits/sec	10**(-2)
Video	11.6 MBits/sec	3.85 MBits/sec	10**(-5)

(a) Data rates of standard Voice, Video models & Typical acceptable loss probability

CCITT G.114 Delay Recommendations	
One-Way Delay	Characterization of Quality
0 to 150 ms	"acceptable for most user applications"
150 to 400 ms	"may impact some applications"
above 400 ms	"unacceptable for general network planning purposes"

(b) End-to-end delay requirements for multimedia traffic

Table 1: Typical data rates and QoS requirements for multimedia traffic

descriptors may not be sufficient to characterize the traffic generation process well enough to predict the expected QoS accurately. For instance, El-Sayed and Perros [5] showed that two sources with the same peak rate, mean rate and mean burst length could experience very different cell losses because of differences in higher order moments of the distribution of burst lengths.

- Deriving an accurate CAC algorithm which can be executed on-line without excessive numerical computation is difficult even for simple traffic model approximations [8].
- Even if the characteristics of individual traffic sources are known, characterizing aggregations of traffic streams is non-trivial and approximate. This approximation is likely to result in sub-optimal allocation.

A consequence of these problems is that the service provider can either end up over-allocating resources in order to guarantee QoS, thereby sacrificing efficiency, or under-allocating resources and providing poor service quality, thus losing customers.

In this paper, we explore an alternate approach to efficient resource allocation *and* providing strong QoS guarantees. In this approach, resource allocations are adjusted dynamically on the basis of the QoS actually received. The main benefit of this approach is that very little information related to traffic characterization is needed from users. The method uses the *minimum* network resources required to *guarantee* the specified QoS requirements for any source with a stationary traffic generation model. The main drawback of this approach is that for a short initial period the requested QoS may not be provided. Also, some re-allocation of resources will be required during this initial period. However, the effects of these problems can be reduced through engineering solutions. This technique also involves a slight additional cost in implementation since provision must be made for measuring queuing performance on-line.[1]

The algorithm studied here is similar in many ways to that analyzed by Hsu [10]. In [10] the convergence of the dynamic rate to the true steady state effective rate for the algorithm studied there, is proved theoretically (for markov-modulated fluid type sources). However, no performance analysis of the algorithm with real sources is presented and the authors do not address the issue of how to select the algorithm parameters for practical implementation. In contrast, this paper concentrates on performance analysis and practical implementability, demonstrates the robustness of the algorithm under different conditions, and demonstrates the effect of algorithm parameters on the dynamic behavior. Further, as we show in later sections, there are some important differences between the algorithm studied here and that analyzed in [10] which make this algorithm arguably better suited for practical implmentability.

In fact both the algorithm studied here and in [10] borrow from well known ideas in control of queueing systems. These algorithms are generically similar to those studied in several earlier papers including those by Bhattacharya et al [2] and Karlsson et al [15]. All these approaches are based on basic control theoretic ideas of scaling queuing parameters based on errors in observed performance metrics. The theoretical analysis in [10] and the practical studies presented in this paper seek to apply these ideas towards ATM resource allocation.

Several other studies which have explored the use of adaptive resource allocation in broadband networks. Jeon and Viniotis [13] studied algorithms which dynamically vary the scheduling priority of different traffic classes in a multi-class queuing system, in order to meet QoS specifications such as loss probability and average queuing delay. They showed that these algorithms were able to achieve any desired performance vector which was feasible under the class of work-conserving service policies. Clark et al [4] developed the FIFO+ queuing discipline, in which the transmission scheduling priority of a packet at a node is dynamically adjusted. This adjustment is based on the queuing delay seen at the previous node. They showed that this resulted in improved end-to-end delay performance as compared to static allocation of priorities. However, the improved resource

[1] However, performance monitoring modules are already expected to be deployed in ATM switches [8].

utilization was obtained at the cost of not being able to *guarantee* the QoS seen by each user. They claimed that strong guarantees on QoS were not necessary for a type of service referred to as predictive service. Jamin et al [12] extended these notions and tested multimedia applications under predictive service. In [22], Chong et al examine dynamic bandwidth allocation algorithms for MPEG video. They examine two different approaches for predicting the rate required to transmit an MPEG frame based on data from the previous frames (one based on a Recursive Least Squares type prediction and the other using Hopfield Neural networks). They demonstrate the performance improvements over static bandwidth allocation techniques. However, in their study, they do not seek to determine a steady state "effective" bandwidth. Hence bandwidth re-allocation is continually required during the duration of a session. The algorithm developed in this paper determines a steady state effective bandwidth so that once the algorithm has converged, no bandwidth re-allocation is necessary. In [3], an algorithm for rate allocation for the ABR (Available Bit Rate [1]) service is proposed. However, the emphasis there is not to *guarantee* any QoS measures, since applications which require ABR service are typically non-real-time (like file transfers). The target applications for the algorithm studied in this paper are characterized as real-time variable bit rate (VBR) type and require QoS guarantees.

In this paper, we mainly concentrate on bandwidth as the network resource of interest, and cell loss probability (CLP) as the QoS measure of interest. We investigate the rate control of a single queue, where the service rate of this queue is synonymous with the term "bandwidth". In case of flow control policies such as Weighted Round Robin [21], Weighted Fair Queuing [18], Stop & Go [7] and RCSP [26], the end-to-end bandwidth is simply this rate multiplied by the number of links traversed. In addition, in a section towards the end, we show that this model can be mapped to a number of different applications including bandwidth allocation, dynamic control of traffic shaping devices such as leaky buckets and call admission control in broadband networks such as ATM. An algorithm is developed which dynamically obtains the minimum bandwidth necessary to satisfy a specified QoS for a given (arbitrary) source. The effectiveness and robustness of the algorithm is demonstrated by varying the measurement frequency, source burstiness, buffer size and CLP specifications. It is shown that this algorithm can yield significant bandwidth savings over popular off-line approaches, such as the "equivalent capacity" method [6]. This approach can also be used to derive the minimum requirements for other resources (such as buffer space), and for other QoS measures (such as queuing delay percentiles). This approach dynamically determines the "effective" resource requirement to satisfy a given QoS measure for an arbitrary (stationary) source. Thus it may be seen as a generalization of the "effective bandwidth" approach.

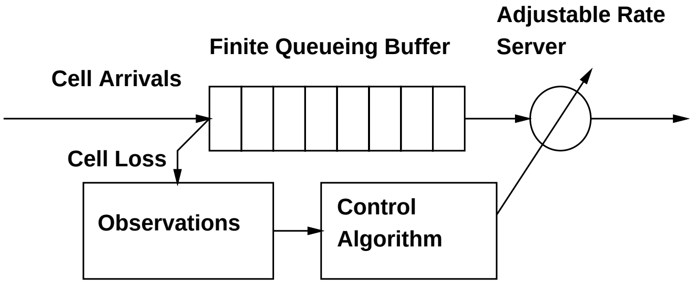

Figure 1: Model of the queuing system under consideration.

Section 2 explains the dynamic algorithm for bandwidth allocation. Section 3 presents simulation experiments, along with an analysis of the results. Section 4 compares the steady state and the dynamic behavior of the algorithm with some alternate approaches for bandwidth allocation. Section 5 then outlines approaches for implementing this technique to perform bandwidth allcation and call admission control in real networks such as ATM. Section 6 summarizes our results and lists issues for future research.

2 Description of the Algorithm

2.1 Queuing Model and Definitions

In order to analyze the algorithm, a simple queueing model will be used for most of the paper. The algorithm studied here, dynamically adjusts the instantaneous service rate $\mu(t)$ of a finite buffer queue in order to meet a specified QoS performance measure Q. In section 5 it is shown how the use of dynamic rate control in this model translates into implementation of appropriate bandwidth allocation and call admission control in ATM networks.

In conformance with the ATM standard, we assume that all customers (hereafter referred to as cells) require the same amount of service. A cell which arrives at a full queue is lost due to buffer overflow. Figure 1 shows a model of the queuing system under consideration.

As shown, observations of the queuing behavior are used by a control algorithm to adjust the service rate of the queue. Let B be the (fixed) buffer size of the system. Decisions about adjustments to the resource allocation are made at discrete instants t_i; the interval between decision points t_n and t_{n+1} is referred to as the nth update interval U_n. The service rate over the nth update interval, denoted μ_n, is assumed to be constant.

For a given sample path of cell arrivals, let $\mathcal{A}_n$ denote the number of arrivals and $\mathcal{L}_n$ denote the number of cells lost due to buffer overflow in the nth interval. The loss probability for each cell in the nth interval is denoted $\mathcal{P}_n = \mathcal{L}_n/\mathcal{A}_n$. The cumulative loss probability for all intervals up to and including the nth interval is denoted $\mathcal{P}_{0..n} = (\sum_{i=0}^{n} \mathcal{L}_i)/(\sum_{i=0}^{n} \mathcal{A}_i)$. When discussing the cell loss probability during the nth interval, we will also use the term "current loss probability". The term "cumulative loss probability" refers to $\mathcal{P}_{0..n}$.

The goal for this method is the following. The instantaneous rate $\mu(t)$ should converge to a steady state rate μ^*, and the cumulative loss rate $\mathcal{P}_{0..n}$ should converge to a steady state value $\mathcal{P}^*$. $\mathcal{P}^*$ should be within an allowable tolerance ϵ of the specified average cell loss probability $\mathcal{Q}_l$, i.e., $\mid \mathcal{P}^* - \mathcal{Q}_l \mid \leq \epsilon$. Further, we would like the steady state rate μ^* to be the minimum rate for which this statement is true. In addition to this goal for the steady-state behavior, the method should converge quickly, and be practical to implement.

2.2 An Algorithm for Resource Allocation Based on QoS Measurement

The result on which the algorithm is based is that loss rate of the queuing system being considered is monotonically related to its service rate. A formal statement of this is the following theorem:

Theorem 1 *Given an arbitrarily fixed sample path of arrivals with constant sized cells, the total number of losses seen in a finite capacity queuing system with deterministic service increases monotonically as the service rate is decreased.*

Proof: The proof is listed in the appendix. □

The algorithm iterates over successive update intervals. The basic step at each iteration is given in equation 1.

$$\mu_{n+1} = \mu_n + K_n E(\mathcal{P}_n, \mathcal{Q}_l) \tag{1}$$

In this equation, $E()$ is an error function which is a measure of how far the current cell loss probability is from the targeted QoS. We chose the error function to be the natural logarithm of the ratio of the current and desired loss probabilities i.e. $E(\mathcal{P}_n, \mathcal{Q}_l) = \log(\mathcal{P}_n/\mathcal{Q}_l)$. This means the rate is increased when the current loss probability is higher than the target, and decreased when it is smaller than the target. By theorem 1, this results in the the drift of the current loss probability and hence also the cumulative loss probability in the right direction. The logarithm function is appropriate because of the wide dynamic range (several orders of magnitude) of losses expected to be measured. In addition, an approximately logarithmic relation between service rate and loss probability has been suggested in the literature. Using this iteration, the instantaneous rate keeps changing as long as the current loss probability ($\mathcal{P}_n$) is not equal to $\mathcal{Q}_l$. Hence convergence of the instantaneous rate implies convergence of the current loss probability.

Figure 2 describes the entire algorithm using pseudocode. The algorithm has two distinct modes of operation. In the first mode, the algorithm tries to quickly converge to the neighborhood of the desired loss probability. In this mode, the lengths of the update intervals are kept fixed, and the scalar K_n is varied. In the second mode, the algorithm tries to gradually converge exactly to the desired loss probability. In mode 2, the scalar K_n is kept fixed, but the size of the update intervals is varied.

We call this the **REQS** (pronounced "rex") algorithm, which stands for **R**esource-**E**fficient **Q**uality of **S**ervice. The algorithm starts in mode 1 with an initial value K_0 for the scalar K_n and an initial update interval U_0. K_n is increased linearly until the sign of the error in the current loss probability changes in successive update intervals. K_n then decreases geometrically at each decision instant until it drops below some value K_∞, at which point the algorithm switches to mode 2 operation. In this mode, K_n stays fixed at K_∞. Each time the error changes sign in successive update intervals, the length of the interval is doubled. The algorithm operates in mode 2 for the remaining duration of the call.

Our motivation for this formulation of REQS follows. The error term which drives the algorithm is based on the loss probability only over the last update interval; that is, the current loss probability is used, not the cumulative loss probability. In mode 1, the algorithm tries to quickly get "reasonably" close to the target. The variation described for K_n first increases the sensitivity of the rate control and then decreases it. The linear increase and geometric decrease in the scalar value was found experimentally to result in the best dynamic behavior of several variations tried. However, the choice of these rules is not critical to the convergence of REQS. The geometric decrease in the scalar value corresponds to a "binary search", since the service rate and loss rate are monotonically related.

An assumption in mode 1 operation is that loss probability measurements over each update interval are statistically significant. If the update interval is too small, however, the loss probability measured over this interval can be considerably far from the average value. On the other hand, a very large update interval may increase convergence time. For this reason, the algorithm starts with a small update interval, and then increases it over time in mode 2. By limiting these increases to

```
/* $K_0$, $K_\infty$ and $U_0$ assumed to be given */
/* initial values: inc_flag = TRUE, mode = 1 */

curr_error ← $\log(\mathcal{P}_n/Q_l)$;
prev_error ← $\log(\mathcal{P}_{n-1}/Q_l)$;

if (mode = 1) {
        $U_n \leftarrow U_{n-1}$;
        if ((curr_error × prev_error > 0) && (inc_flag = TRUE))
                $K_n \leftarrow K_{n-1} + K_0$;
        else {
                inc_flag ← FALSE;
                $K_n \leftarrow K_{n-1}/2$;
                if ($K_n \leq K_\infty$) {
                        $K_n \leftarrow K_\infty$;
                        mode ← 2;
                }
        }
}
else {                  /* mode = 2 */
        $K_n \leftarrow K_{n-1}$;
        if (curr_error × prev_error < 0)
                $U_n \leftarrow 2 \times U_{n-1}$;
        else
                $U_n \leftarrow U_{n-1}$;
}
```

Figure 2: Algorithm for varying K_n and U_n at decision instant t_n

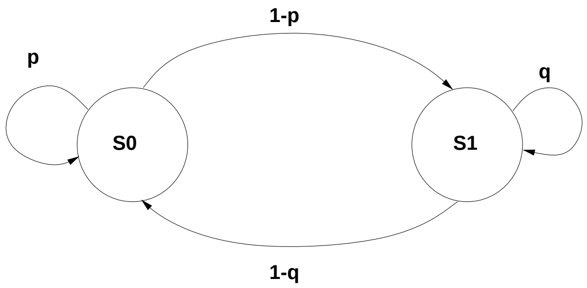

Figure 3: Model of MMBP Source used for simulations

instants when the error changes sign, the update interval is prevented from becoming excessively large. During mode 2 the algorithm operates at minimum sensitivity (since the scalar is at its minimum value). A change of sign in the error is likely to be due to noisy measurements; increasing the update intervals yields better confidence in the loss measurements. Further motivation for the formulation of REQS is presented below in our discussion of experimental results.

3 Numerical Results

Analytical treatment of transient behavior of queues yields closed form solutions in very few cases. For instance, closed form solutions for the time-dependent queue performance measures such as buffer occupancy are available only for simple queues such as the $M/M/1$ queue [17]. Transient analysis of a queue with variable service rate and bursty input traffic is consequently not expected to yield closed form results. Hence we used simulation to analyze the performance of REQS.

3.1 Simulation Model and Procedure

The simulation model was shown earlier in Figure 1. A two-state Markov Modulated Bernoulli Process (MMBP) was used as the traffic source in most experiments. This model has been used extensively in simulation and analytical studies of bursty multimedia traffic [5]. The 2 state MMBP is in either of two states $S0$ or $S1$. In $S0$ cells are generated according to a Bernoulli Process with a mean rate of λ_0 cells/sec, and in $S1$ with a mean rate λ_1 cells/sec. The durations of the two states are geometrically distributed with means γ_0 and γ_1 seconds. A measure of the burstiness of the source is the squared coefficient of variation of the inter-arrival times, which can be determined from the above parameters [5]. For most experiments, values for these parameters were chosen as

$\lambda_0 = 200$ cells/sec, $\lambda_1 = 1000$ cells/sec, $\gamma_0 = .02$ seconds, $\gamma_1 = .01$ seconds. These values reflect typical behavior for voice and video, but scaled down to reduce the simulation time required. In some of the experiments these values are varied to investigate the sensitivity of the algorithm to different traffic conditions. One experiment also uses parameter values based directly on MPEG-compressed videos, to test more realistic conditions.

To determine whether the dynamic algorithm converged, we examined sample paths of the instantaneous rate and the loss probability. If the algorithm converges to the right steady-state cell loss probability, theorem 1 guarantees that the instantaneous rate is the minimum that would achieve that QoS. Experimental evidence is needed, however, to confirm that convergence will occur. The *convergence time* is defined to be the time required to arrive at and stay within 5% of the steady state rate. The steady state rate was determined by running the simulation for long enough (typically for five hours or more of simulated time), to ensure that no further rate changes would occur. This definition of convergence time corresponds to the term 'relaxation time" used in transient analysis of queuing and control systems[23].

The method of batch means was used to determine convergence time. In each batch, a number (typically 3-5) runs were simulated of the queuing system with the dynamic rate control for sufficiently long times to obtain convergence. The error with respect to the steady state rate for each run was averaged over the batch to obtain an averaged error curve. The convergence time obtained from this curve was taken as convergence time of this batch of runs. A number of batches (typically 15-20) were executed for each setting in order to obtain the average convergence time along with the confidence intervals. 90% confidence interval levels were calculated in all cases, and are shown on the graphs. In this work we assumed that the service rate can be set to any real value; there is no quantization effect due to being limited to a discrete set of rates.

In all experiments, the initial service rate μ_0 was set to the mean rate λ_1 of the MMBP in its high state. λ_1 roughly represents the peak rate of this source, so this corresponds to a conservative initial allocation. The values of K_0 and K_∞ were both set equal to about 1% of λ_1 in all cases. Note that K_∞ determines the final sensitivity of the rate control and hence must be small. Larger values of K_∞ were found to result in improved convergence times, at the expense of accuracy in the steady state rate. The value of K_0 was not found to be critical. Since the scalar is multiplied by the error term $\log(\mathcal{P}_n/\mathcal{Q}_l)$, adjustments in the rate are typically much greater than 1% of the peak rate.

We now present the results of simulating this algorithm.

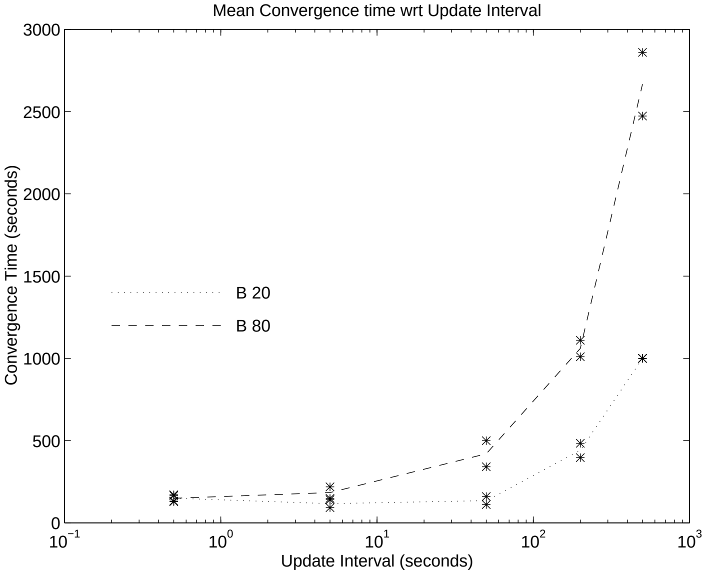

Figure 4: Convergence time with varying initial update interval. Standard source, target CLP 1e-3.

3.2 Variation of Convergence Time with Initial Update Interval

Our first experiment measured the sensitivity of REQS to the choice of the initial update interval, U_0. The initial update interval was varied over the values .5, 5, 50, 200, and 500 seconds. The resulting convergence time is plotted in Figure 4 for buffer sizes of 20 and 80 cells, and a target CLP of 1e-3. These graphs indicate that convergence time is relatively unchanged as long as the initial update interval is small enough.

Since REQS is driven by the error in the loss probability, the convergence time is closely related to the inherent relaxation time of the time averaged loss probability of the given queuing system. The relaxation time of the loss process is quite difficult to predict, and can be significantly greater than the time for the arrival statistics to converge. In other words, the loss measurement cannot be a reliable measure of the true current average loss probability if the measurement interval is small compared to the relaxation time of the loss process. For example, a suggested rule of thumb for the relaxation time of the $M/M/1/K$ queue is $1/(\sqrt{\mu}-\sqrt{\lambda})^2)$ [17] (where λ and μ are the average arrival and service rates), which suggests that even for an arrival process with very low burstiness, the time required for the queue transients to disappear, could rise exponentially as the utilization is increased. Again, the update interval should be short so that we correct the rate more frequently, but not too short that the per-interval measurements are not reliable.

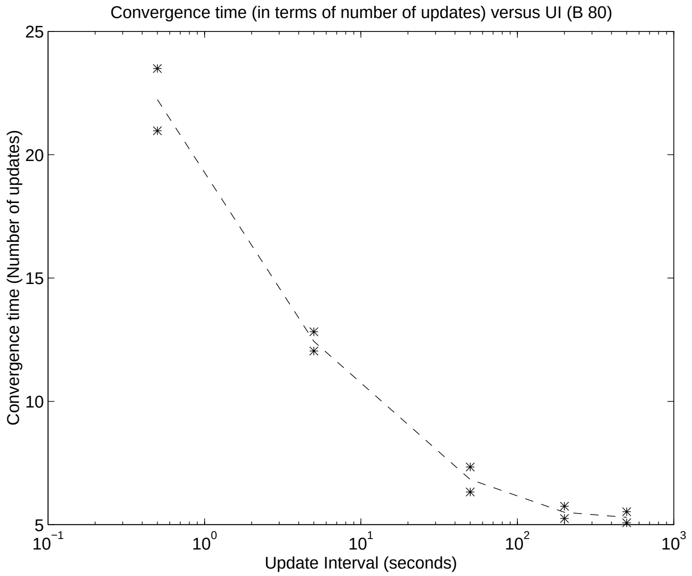

Figure 5: Convergence time (in terms of number of updates) with varying initial update interval. Standard source, target CLP 1e-3, $B = 80$.

The above arguments indicate that selecting a good fixed value for the measurement interval is non-trivial. However, REQS/ works well without requiring a good choice of the measurement interval, which is an important advantage in terms of robustness. This is demonstrated in figure 4. An analysis of the threshold type nature of these curves is interesting. It was found that for initial update values corresponding the flat portion of the curve the algorithm converged in phase 2 while for the rising portion, convergence occurred in phase 1. This can be explained from the working of the algorithm and from the above arguments on queue relaxation times. We do not delve into details here.

Figure 5 presents the convergence times for the same experiments (B=80 only), but measured in terms of the required number of updates. This metric is important because reallocation of resources may be an expensive operation. A small initial update interval requires a larger number of rate changes, while a large initial update interval only requires about 5 updates. Note that the number of updates does not directly scale from the previous graph, since the doubling of the update interval starts at different times for different values of U_0.

There is thus a tradeoff between absolute convergence time and required number of updates. Given an arbitrary source, a smaller U_0 should result in less convergence time, while a larger U_0 should result in fewer number of updates. In these experiments, an update interval of about 50 seconds

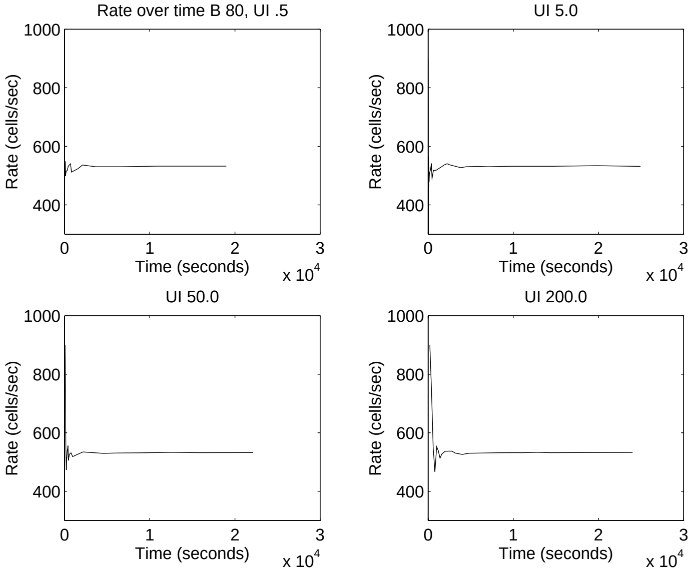

Figure 6: Sample paths of rate for different initial update intervals. Standard source, entire simulated time, target CLP 1e-3, $B = 80$)

represents a good compromise of these two factors. The best "balance point", however, is likely to depend on the exact source characteristics and the related relaxation times, which are assumed to be unknown.

Figure 6 shows sample path results of the instantaneous rate for four values of U_0 ($B = 80$ only). Although we do not provide a formal proof of convergence, data like this strongly supports our assertion that the algorithm converges. In fact, REQS converged in every case in which it was tested, for all experiments listed in this paper. The total simulated time is more than ten times the calculated convergence time, which is a strong indication that the rate has indeed converged. Figure 7 expands the initial portion of these graphs to illustrate the behavior of the method during convergence.

Data on the resulting losses incurred during these same experiments was also collected. Figure 8 shows the cumulative loss probabilities (i.e. $\mathcal{P}_{0..n}$) for the above sample paths,[2] while Figure 9 shows the loss probability in the current update interval (i.e. $\mathcal{P}_n$). These figures demonstrate that the cumulative loss probability is accurately controlled, even though REQS considers only the current loss probability. The accuracy with which the loss probability is controlled supports our claim of

[2] In the plots shown, the cumulative CLP was reset when the algorithm switched from mode 1 to mode 2 to negate the effect of the initial losses. However, this resetting is performed only once.

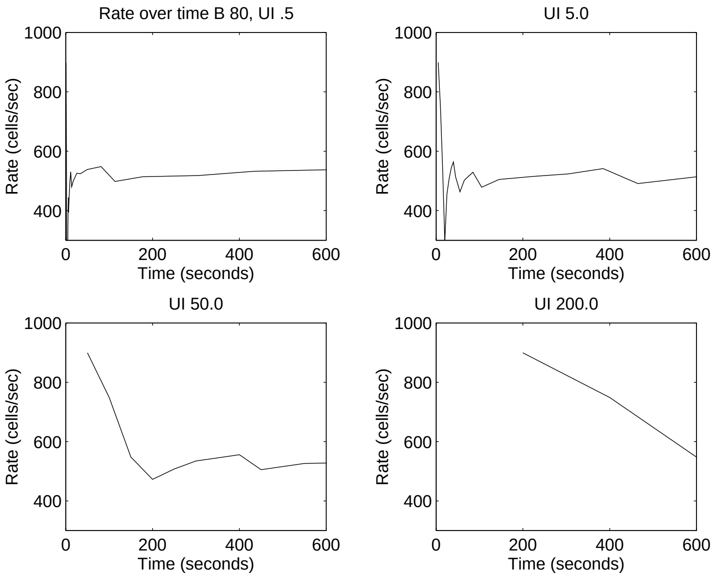

Figure 7: Initial portion of sample paths of rates for different initial update intervals. Standard source, target CLP 1e-3, $B = 80$

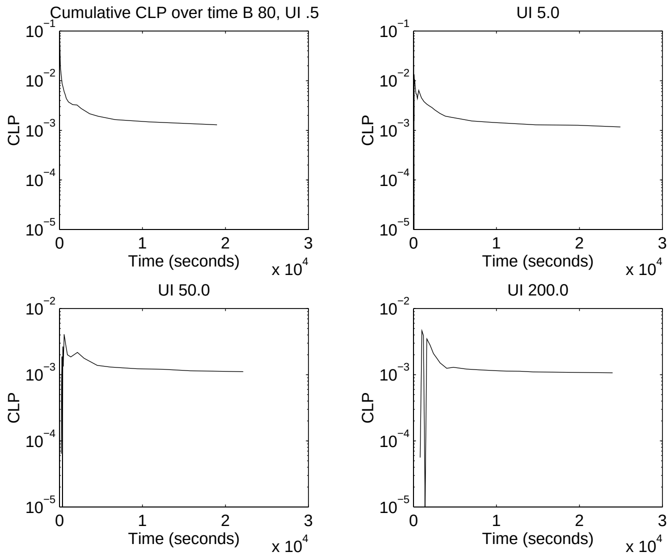

Figure 8: Sample paths of cumulative CLPs for different initial update intervals. Standard source, target CLP 1e-3, $B = 80$.

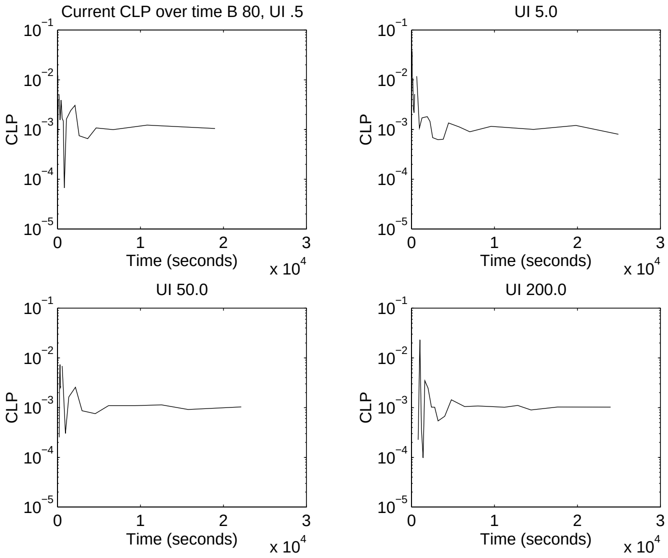

Figure 9: Sample paths of loss probability over current update interval for different initial update intervals. Standard source, target CLP 1e-3, $B = 80$

providing QoS *guarantees.*

Another experiment was conducted, with the same traffic source and buffer size, to determine how close to optimal the proposed algorithm is. In this experiment, the service rate was fixed from the beginning at the rate converged to in the previous experiment. The purpose was to monitor loss behavior of a queue which was initially allocated exactly the right service rate to achieve the desired CLP. The cumulative cell loss probability was then measured for each of 10 different runs, and the error in this CLP (percent difference from the desired CLP of 1e-3) was calculated. Figure 10 plots the error in the CLP for this optimal method. It can be seen that the time taken for the error to drop to less than 10% is on the order of 200 seconds. Note that the convergence time of REQS was about 150 seconds with a small enough initial update interval. This is strong evidence that the convergence time is reasonable, and depends on the inherent relaxation time of the loss probability of the queue.

No closed form solutions are known for describing the transient measures of a queue with bursty arrivals such as MMBP sources. In [17], closed form solutions for the distribution of number of customers in an $M/M/1/K$ system conditioned on the initial number are presented. As a rule of thumb the author suggests a relaxation time of $1/(\sqrt{\mu} - \sqrt{\lambda})^2$ for the distribution of the buffer occupancy of such a system. This formula indicates that relaxation time increases with utilization.

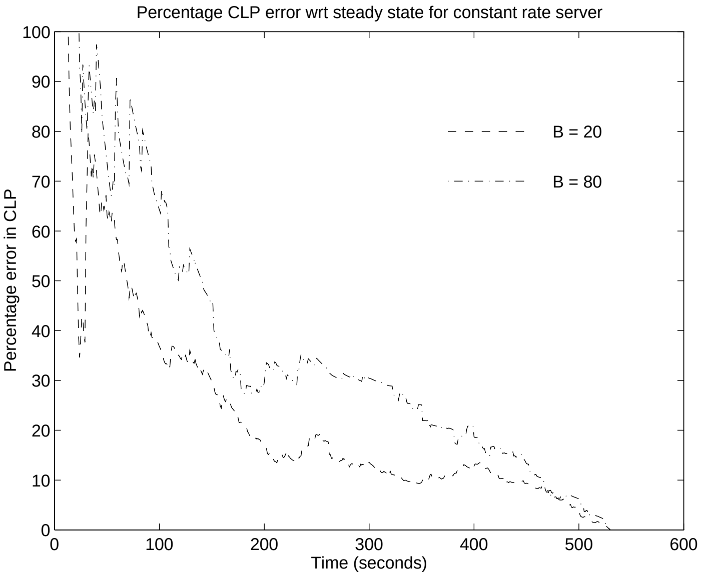

Figure 10: Fractional error of transient CLP with constant rate server. Standard source. Steady state CLP 1e-3.

In this experiment, this is confirmed by the increased convergence time with a buffer of 80 cells versus 20 cells, as shown in Figures 4 and 10. As the buffer size is increased, a lower rate can achieve the same average loss specification, leading to higher utilization.

Figure 10 also indicates that a few dozen seconds is a reasonable choice for the measurement intervals for this queue. Thus, there is not much to gain by choosing U_0s much larger than 50-100 seconds, as was seen in Figure 4.

3.3 Variation of Convergence Time with Buffer Size

The variation of convergence time with buffer size was studied next to determine sensitivity of REQS to this parameter. The size of the queuing buffer was varied over 5, 10, 20, 40 and 80 cells. The same source model and CLP specification (1e-3) were used as before. Figure 11 shows the results for two different values of U_0.

For $U_0 = 5$, convergence time increases very slowly with increasing buffer size. This indicates the desired robustness of REQS to changes in system parameters. It also indicates that convergence times should be reasonable even for very large buffer sizes. For $U_0 = 200$, convergence time is longer, and exhibits a surprising behavior for small buffer sizes. The high convergence time for

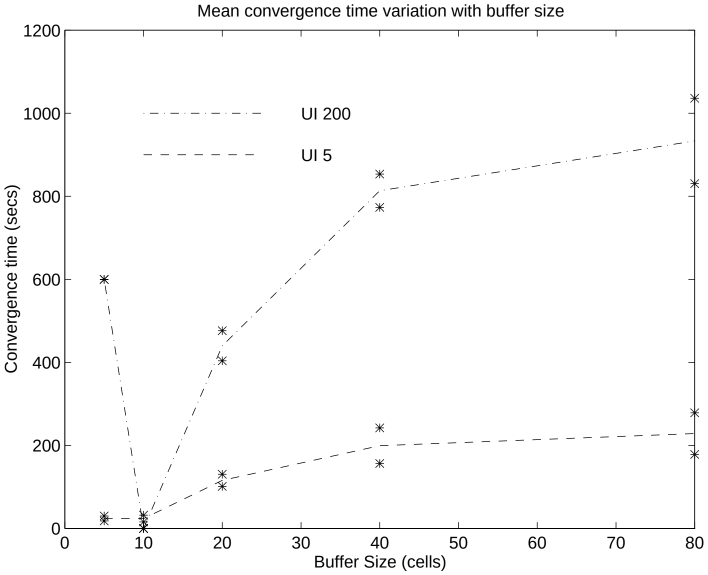

Figure 11: Convergence time versus buffer size. Standard source, target CLP 1e-3.

small buffer size is because the initial rate μ_o is actually below the final (converged) rate μ^*, as seen from the sample paths of the rates in Figure 12. For buffer sizes much smaller than the mean burst data length, it is known (e.g. [8]) that cell scale congestion during a burst is responsible for losses. The average amount of data in a burst for this source was 10 cells which is twice the buffer size in this case. Hence, the effective rate can be greater than 1000 cells/secs. This unfortunate parameter choice demonstrates, however, that REQS converges to the correct value even when the initial rate is less than the final rate. Better convergence times for the case of small buffers can be obtained by setting the initial rate to a higher value.

3.4 Variation of Convergence Time with Burst Length

Another experiment investigated the sensitivity of the algorithm to changes in the characteristics of the traffic source. The burstiness of the source was varied, while keeping the same peak and mean rates. This was accomplished by varying γ_0 and γ_1 (the mean lengths of the two states), but keeping the ratio of the two constant.

The variation in convergence time with the burst length (duration of $S1$) is shown in Figure 13. Again, non-monotonic behavior is seen. Two opposing trends result in this non-monotonic behavior. As burst lengths increases, source burstiness increases (the squared coefficient of variation of the

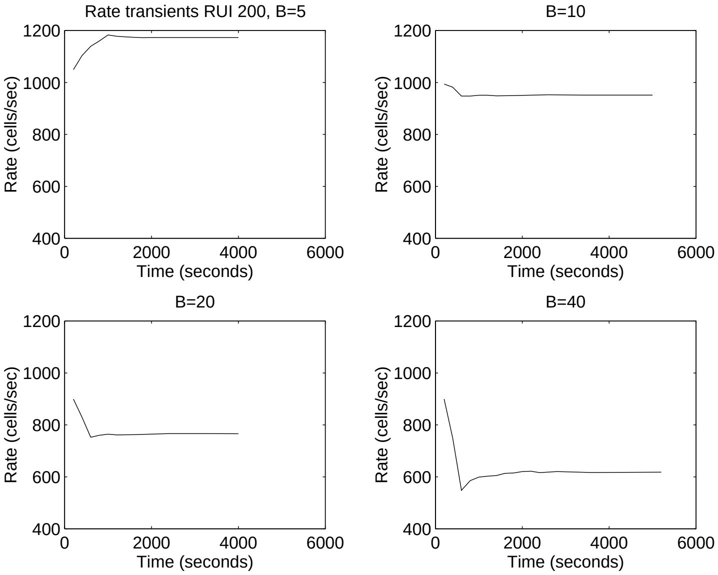

Figure 12: Sample Paths of instantaneous rates for buffer variation. Standard source, target CLP 1e-3, U_0 200s.

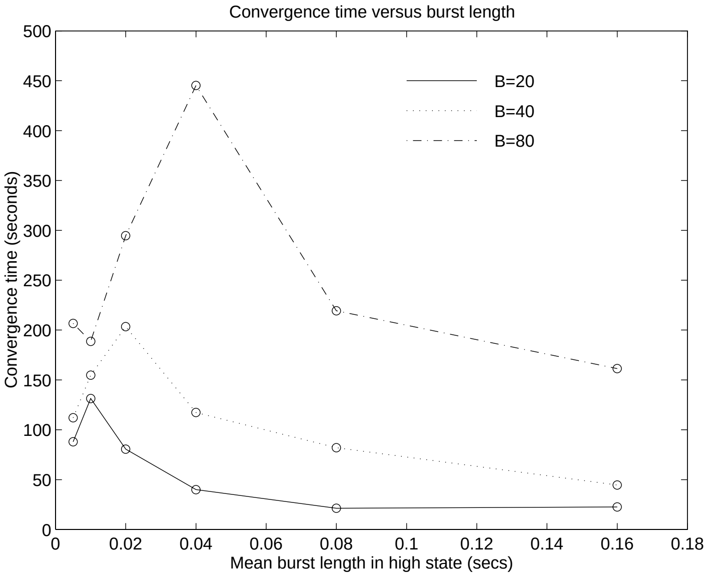

Figure 13: Convergence times for varying burst lengths. U_0 10s, Target CLP 1e-3. Confidence intervals not shown for clarity.

inter-arrival times ([5] increases for instance). The relaxation time of the loss probability can be expected to increase with source burstiness since the loss process will also become more bursty. Such an observation was also made by Wang et al [23] in their study of transient analysis of queues with bursty traffic. As source burstiness increases, queue service rates must increase (meaning lower utilization) to achieve a given loss rate. Lower utilization levels tend to reduce the inherent relaxation times, as described in section 3.2, and hence the convergence time of REQS. For very small burst lengths, the effect of increase in burstiness is dominant, while for larger bursts, the effect of reduced utilization is dominant. This results in the non-monotonic behavior seen. Analogous non-monotonic behavior was also seen in [23], where the maximum transient overshoot first increased, then decreased as the burstiness of the arrival process was increased.

3.5 Variation of Convergence Time with Peak-to-Mean Ratio

In this experiment, the peak-to-mean ratio of the source was varied to see how this affects convergence. This variation was accomplished by varying γ_0 (the mean duration of the low state), while keeping the rates in the two states as well as the mean duration of the high state fixed. The mean duration of the low state was varied over .01, .02, .04, .08 and .16 seconds. U_0 was set to 10 seconds and the buffer size was 20.

The variation of the convergence time is shown in Figure 14. The burstiness of the source increases with the peak-to-mean ratio. The final (converged) rates decrease monotonically as peak to mean ratio is increased, so that utilizations increase with the peak-to-mean ratio. Hence both trends are now in the same direction, and convergence times increase monotonically with the peak-to-mean ratio.

3.6 Variation of Convergence Time with CLP Requirement

We also investigated the sensitivity of REQS to the QoS specification, in this case the target CLP. The target CLP was varied over 5 orders of magnitude (1e-1 through 1e-5), with a U_0 of 10 seconds and $B = 20$. The results are shown in Figure 15. Convergence time increases with lower loss requirement, since the algorithm must estimate the CLP with rarer events.

The sharp rise in convergence time for lower loss probabilities indicates convergence may be slow for very low loss requirements, such as 1e-8 or 1e-9. As discussed elsewhere (e.g. [16]), a steady state loss probability for very low probability events makes sense only over very long time scales. A

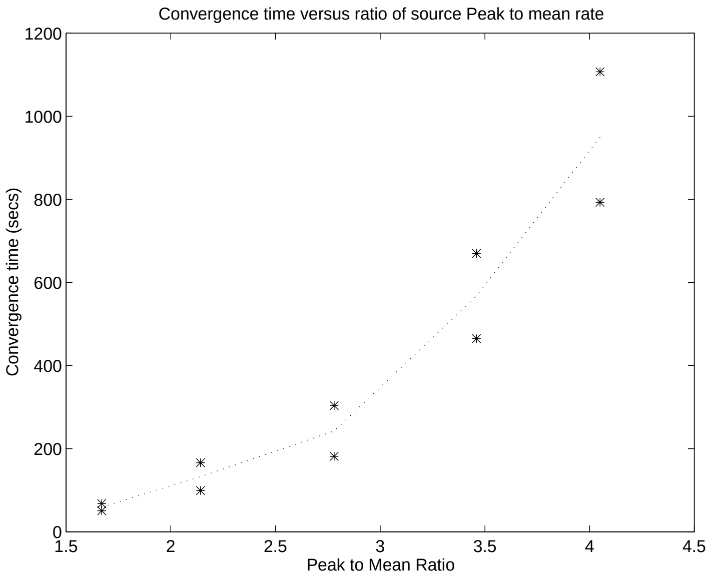

Figure 14: Convergence times for varying peak-to-mean ratio, (mean low state duration varied). Target CLP 1e-3, U_0 10s, B 20 cells

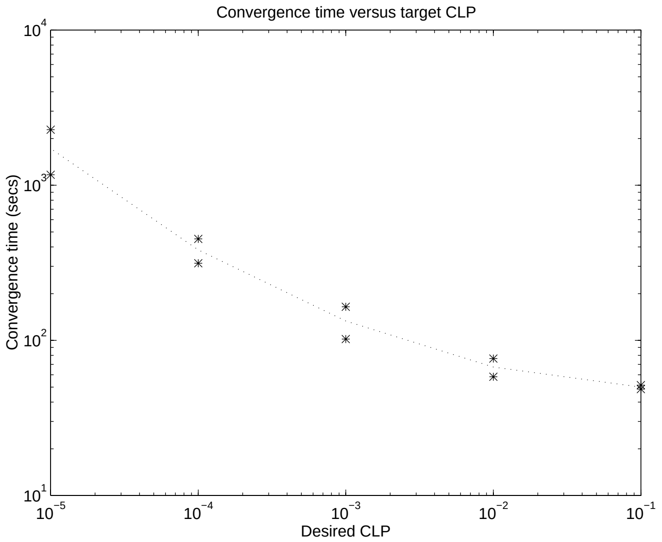

Figure 15: Convergence times for varying target CLP. Standard source, U_0 10s, $B = 20$.

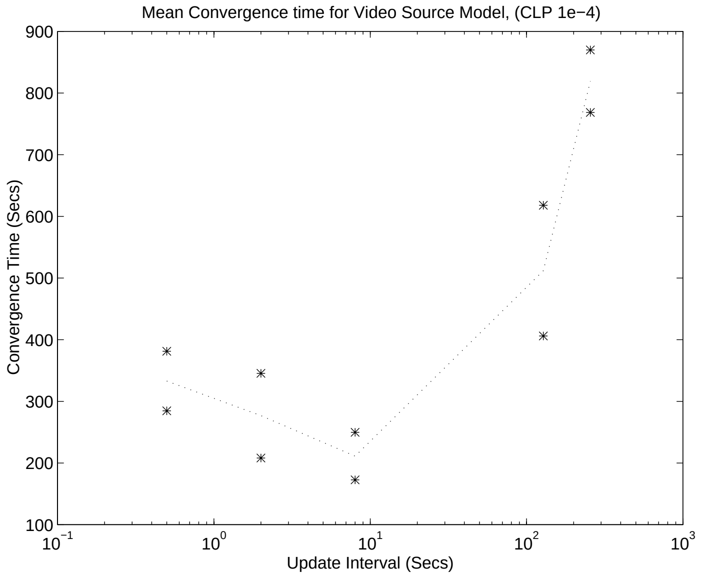

Figure 16: Convergence times for varying U_0. MMBP model at video rates, target CLP 1e-4, $B = 1000$.

user who specifies a very low CLP must understand this limitation. It is possible the convergence time can be reduced using techniques developed for fast simulation of rare events[24].

3.7 Convergence Times at Video Rates

An approximate 2 state MMBP model of a compressed video source was constructed and used as a traffic source for one experiment. This was done to get some idea of the performance and robustness of REQS at typical video rates.

The source is modeled as a 2 state MMBP with high rate of 28000 cells/sec and low rate 5000 cells/sec. Average On and Off times are 30 msec and 60 msec, to approximately model an IBBPBBI-type MPEG-compressed video. The rates in the two states are based on figures for the average amount of data in I and P frames and in B frames, respectively, as presented in [11]. The buffer was set to 1000 cells in order to be larger than the average burst length of 840 cells. The target CLP was set to 1e-4.

Figure 16 shows the behavior as a function of varying initial update interval. Qualitatively the behavior is similar to that in Figure 4. Conclusions about the algorithm also seem to hold true for this more realistic traffic source. The convergence times (about 300 seconds) of REQS appear to

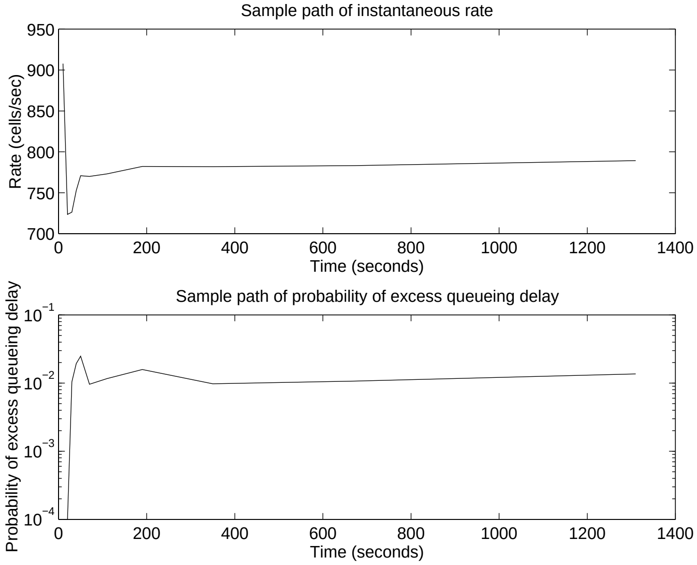

Figure 17: Example of Use of Rate Control for a different QoS measure (percentile of queuing delay). Need fraction of cells with queuing delay greater than 25msecs no more than 1%. Standard source, $B = 100$, U_0 20 seconds.

be reasonable, since a video session is expected to last up to several hours.

3.8 Resource Allocation Algorithms for Other/Multiple QoS Measures

REQS was also used to determine the minimum resources to achieve a different type of quality of service. In this experiment, a bound of 25 msecs was set on the queuing delay, and it was required that no more than 1% of cells experience a queuing delay in excess of this bound. Such a QoS requirement may be specified for an interactive multimedia session in which bounding the fraction of excessively delayed cells is important. Instead of measuring the loss probability, the algorithm measured the fraction of cells which experienced a queuing delay in excess of the given bound. This measurement was used to modify the allocated service rate.

The standard source for the earlier experiments was used, with a buffer size of 100 and an initial update interval of 10 seconds. One sample path of the instantaneous rate and current percentage of cells which experience queuing delay greater than 25 msecs is shown in Figure 17. Clearly, REQS can also be used to control this QoS. This is because this QoS measure also varies monotonically with service rate, as proved in [20].

For the case where multiple QoS metrics are specified, REQS can be modified to obtain the minimum bandwidth at which all specified QoS measures can be met. This can be done by simultaneously tracking all the QoS measures of interest and setting the rate to be the maximum of that required to meet each of the measures independently. Since service rate monotonically controls all the measures, the rate will converge to the value corresponding to the most severe QoS requirement. Experimental results for this case are not presented, due to space limitations.

This concludes the experimental validation of the proposed algorithm. The next section compares REQS with other methods of bandwidth allocation.

4 Comparison with Other Approaches

4.1 Comparison of Steady State Performance with Equivalent Capacity Formulas

The equivalent capacity formula [6] is a widely-cited method for estimating the bandwidth needed to achieve specified loss and queuing delays, for certain types of traffic sources (markov-modulated fluids, in which the state durations are exponentially distributed). We compared experimentally the steady state service rate converged to by REQS with the rate calculated from the "equivalent capacity" formulas.

The same standard source as in earlier experiments was used, with deterministically distributed durations of S_0 and S_1 (the two states of the bursty source). This experiment used deterministic distributions for the state durations to illustrate one problem for the equivalent capacity formulation which does not affect REQS. The queuing system of Figure 1, with REQS as the method of resource allocation, was simulated to find the steady state rate. The equivalent capacity for this same traffic source was calculated using the method of [6], for a target loss rate of 1e-3. These two service rates were then compared. Figure 18 shows how much more bandwidth was required by the equivalent bandwidth formula, than was predicted by REQS (for the same loss rate). This comparison was performed for several buffer sizes.

The bandwidth savings is always non-negligible. For small buffer sizes, the equivalent capacity formulation overallocates the bandwidth by as much as 75%. Guerin et al. [8]. have described a number of situations in which the equivalent capacity formula is either overly optimistic or overly pessimistic. REQS, in contrast, always converges to the minimal rate to achieve a specified loss

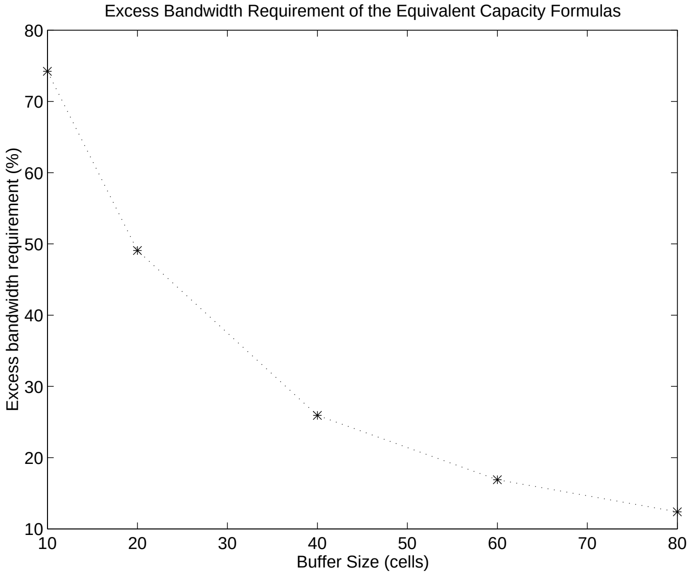

Figure 18: Percentage excess bandwidth predicted by Equivalent capacity formula. Standard source, deterministic distribution of state durations, target CLP 1e-3, $U_0 = 20s$.

probability, for a wide range of traffic characteristics.

The problem with non-exponentially distributed states for the equivalent capacity formulas has been addressed in [9] so this may not be as serious a problem. However, another problem with the equivalent capacity formulas is that these ignore the potential for statistical multiplexing [8]. As a result, these result in overallocation of resources when applied to an aggregation of sources. REQS, on the other hand, can be used to allocate resources for a set of multiplexed sources, to yield an aggregate QoS for the multiplexed sources. In this case, the minimum total rate is found which statisfies the specified aggregate QoS for the multiplexed sources [3]

4.2 Comparison of the Dynamic Behavior with Alternative Approaches

Some alternate approaches were tested for their dynamic performance, i.e., how fast they converged relative to REQS. One alternative is to use a fixed value for the scalar K_n, rather than adapting it over time as described above. Several experiments were run to evaluate this alternative; the resulting plots are omitted due to space limitations. This alternative performed reasonably well

[3] In this case, the QoS of individual sources may be more or less than the specified aggregate QoS. The user gives up some precision in specifying the desired QoS, in return for lower network overhead, and a reduction in the required resources.

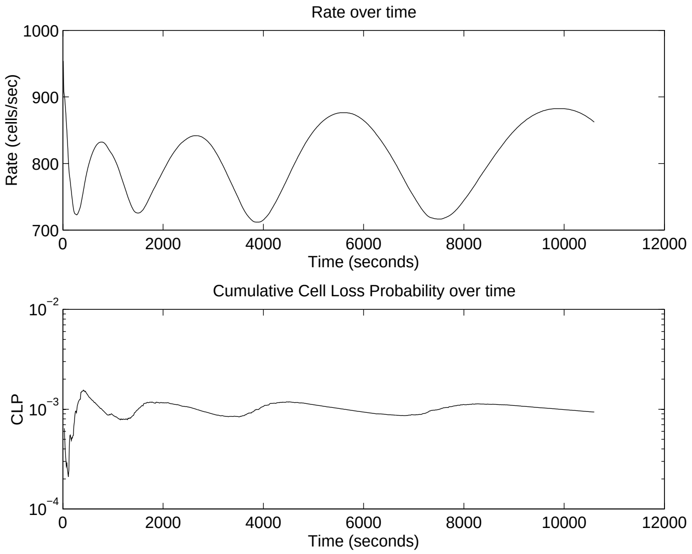

Figure 19: Problems with approachs based directly on error in cumulative QoS measures. Performance measure (CLP) converges but control (rate) exhibits strong oscillations. Standard source, target CLP 1e-3, U_0 10s, $B = 20$.

when the initial update interval was small, but required more time to converge when the initial update interval was large. With a large U_0, the algorithm must achieve convergence during mode 1 of operation; however, the constant scalar version of the algorithm performs poorly in mode 1.

A second alternative substituted measurements of the cumulative loss probability, rather than the current loss probability, in the basic iteration of equation 1. Other dynamic algorithms based on feedback (for example, [13]) have also used the error in the cumulative performance measure to adjust the resource allocation. This alternative was investigated experimentally. It was found that while the CLP converged very well to the desired value, the control parameter (in our case, the service rate) did not always converge to a steady state value. The reason is that over time the cumulative loss probability becomes less and less sensitive to changes in the instantaneous service rate. As a result, large changes in the service rate are needed to adjust the cumulative CLP. Figure 19 shows one pair of sample paths of the rate and cumulative loss probability, using this alternative. While the CLP appears to converge, the service rate oscillates, and in fact the magnitude of the oscillations increases. In [13] the control parameter (the scheduling priority) was not required to converge to a steady state value, so this was not a problem. Our experimental results indicate this approach will have difficulty converging both the QoS measure and the controlled resource. In contrast, REQS is able to control the cumulative loss very well, by measuring and using only the current loss probability. At the same time, the resource allocation converges to a steady state value in every case that was simulated.

A third alternative to REQS is to keep the update interval a constant length, rather than lengthening it as described. In experiments, this alternative also exhibited a tendency to oscillate. The reason is that the loss probability over any finite interval will never converge. An algorithm (such as REQS) which tracks the loss probability over the current update interval must necessarily lengthen the update intervals to ensure convergence.

As mentioned earlier, Hsu and Walrand [10] have independently proposed an iterative algorithm with a similar formulation to ours. Some differences are that Hsu's algorithm uses a fixed update interval, and multiplies the error in the measured CLP by a factor of $1/n$, where n represents the iteration. Hsu was able to prove this algorithm would converge to the desired CLP, under certain assumptions. The speed of convergence was not a primary issue in this work.

For an arbitrary traffic source, the multiplier $1/n$ may decrease much faster than desired. Before the actual QoS converges to the target QoS, the rate adjustment can become too small to make sufficient further progress. This happens if the update interval is chosen too small, resulting in an excessive number of updates. As we have seen from the simulation results, convergence times vary significantly with the choice of update interval. No technique for selecting a good update interval was suggested in [10]. In contrast, the method of adjusting the update interval (during mode 2) proposed in this paper ensures convergence, independent of the source characteristics. We believe our work on the practical behavior of dynamic rate allocation complements the theoretical work of [10].

5 Applications

The queuing model used for analysis of the REQS/ algorithm was a simple queue whose service rate could be controlled. The arrival process was arbitrary but all packets (cells) were of the same length. This can be applied in different ways in broadband networks some of which are outlined here.

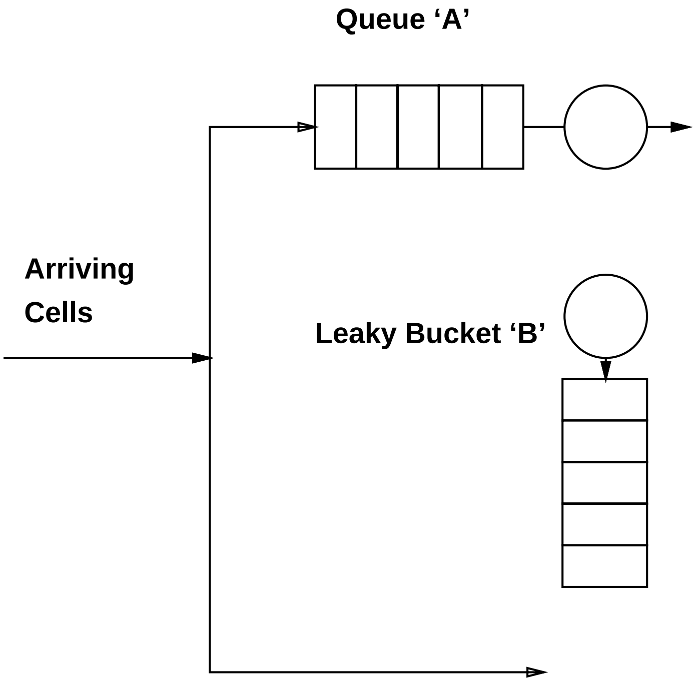

Figure 20: Illustrating duality between a standard queue and a leaky bucket.

5.1 Control of rate enforcing servers

Rate enforcing servers such as WRR [21], Stop&Go [7] and such others, require enforcing a deterministic service rate on each session in order to provide end-to-end QoS guarantees. Using this technique, the optimal rate for each bursty session can be dynamically obtained. The queuing model in Figure 1 can be seen as a single queue of a multi-queue system being served in a round robin manner. Using a technique such as REQS, each queue in such a system can be served at the minimal necessary rate.

5.2 Dynamic Control of Leaky Bucket Rates

A Leaky Bucket mechanism may be seen as a dual of a queue. This is illustrated in Figure 20. The sizes of queue ‘A’ and the token buffer size of Leaky Bucket ‘B’ are the same. Let each cell require one token. Let us assume the token buffer is full initially and the token generation rate of ‘B’ be equal to the service rate of queue ‘A’ in the figure. Consider an arbitrary arrival sequence fed simultaneously to both systems.

It may be seen that an arriving cell will be lost due to overflow at queue 'A' if and only if the same cell does not find an available token at the leaky bucket 'B'. Hence these systems are "duals" of each other. (There are additional modeling assumptions relating to mapping these systems as duals of each other. However, without going into detail, we assume here that these do not change tha applicability of the technique to Leaky Bucket control). Hence, the REQS/ algorithm may be used to dynamically determine the minimum token generation rate necessary to control the average probability of running out of tokens, below a specified threshold. In case of ATM networks, it has been found that setting the 'SCR' traffic parameter for an arbitrary bursty source is a a difficult problem. This technique can be used here to dynamically determine a good value of the 'SCR'.

5.3 Call Admission Control in ATM Networks

As mentioned above, the queuing model used for simulation studies can represent a single queue in a multi-queue system being served in a round robin manner. In fact, the REQStechnique can also be applied for call admission control in an ATM queueing model in which we have no control on the queue service rate. Figure 21 shows a model of the output buffer at an ATM link. The queue service rate is fixed and equals the link bandwidth. Here, the REQSalgorithm can be applied by using a second queuing system to which a "copy" of the traffic destined for the ATM link buffer is sent. The second queuing system is used only to "learn" the minimum bandwidth requirement on the main ATM link. Using the REQSalgorithm, the service rate of the second queue will converge to the minimum rate needed to satisfy the QoS of the aggregate traffic. This rate will represent the minimum bandwidth reservation needed on the actual ATM link.

An example call admission control (CAC) algorithm using this model and a technique such as REQSis as follows. Assume that at some time n virtual connections are sharing the ATM link and that using an algorithm like REQS, the service rate of the second queue has converged to some value C_n which is less than C_l, the (fixed) reservable portion bandwidth of the ATM link [4]. When a virtual connection $n+1$ seeks admission, we use some conservative measure of the bandwidth requirement of this call in isolation (this could be based on peak bandwidth or equivalent capacity [8] for example). Let this bandwidth be c_{n+1}. The CAC rule is now simply that if $C_n + c_{n+1} \leq C_l$, admit the call else reject it. If the call is admitted, once again initiate the dynamic procedure of REQSso that the rate of the second queue now converges to the miminum bandwidth requirement for the $n+1$ virtual connections now using the link. Since C_n is the true minimum requirement

[4] Typically, only upto about 85% of a link's capacity is set aside for bandwidth reservation

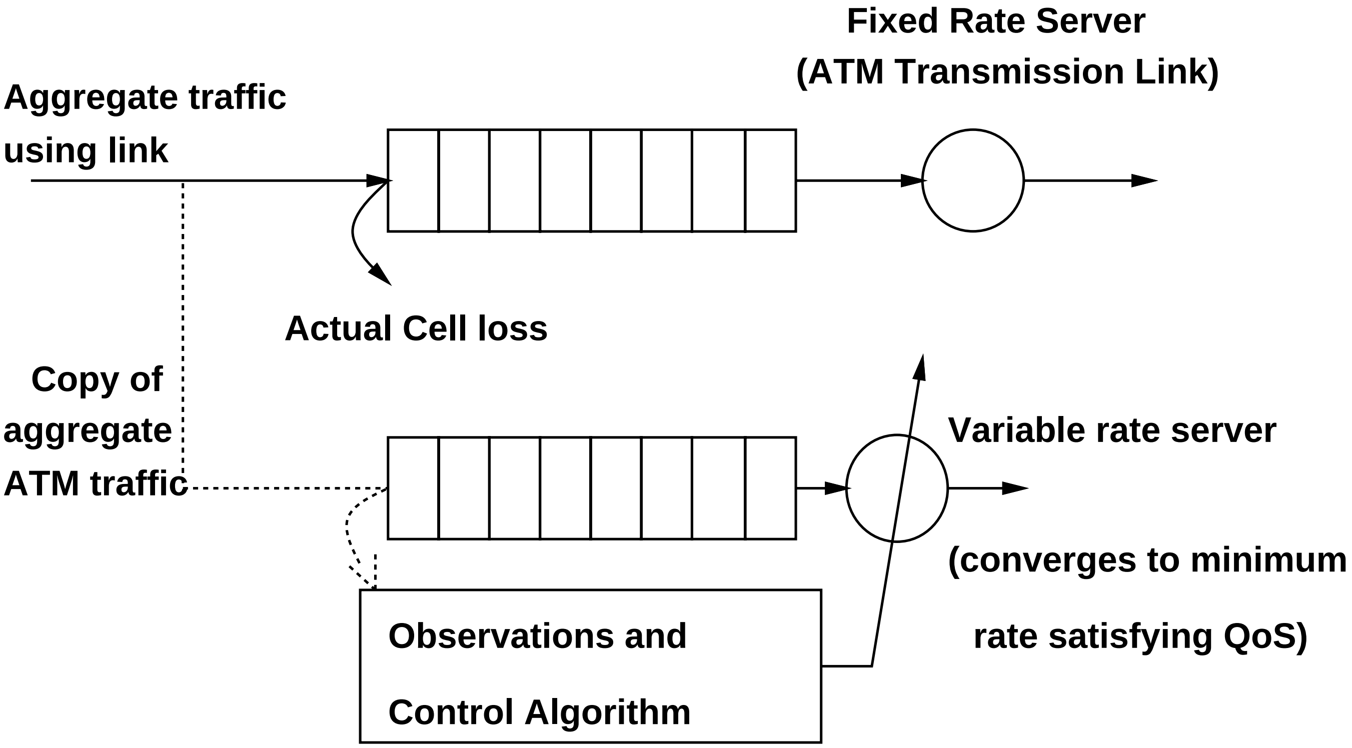

Figure 21: Use of dynamic bandwidth allocation for admission control.

for the first n connections and c_{n+1} is definitely a conservative estimate of the requirement for connection number $n+1$, the minimum bandwidth requirement for the $n+1$ calls will necessarily be less than $C_n + c_{n+1}$. This technique should be applied whenever a new call enters the network, an existing call leaves the network, or existing calls re-negotiate their traffic parameters. One potential problem with this approach to admission control is that the call arrival rates may be very high and the REQSalgorithm may not converge for n calls by the time the $n+1$th call seeks admission. This may be addressed by retaining a conservative allocation for each connection which arrived before the dynamic algorithm converged, until the call arrival drops to a sufficiently low value that the algorithm is able to converge in time. However, this problem needs further study.

6 Conclusions

This paper has studied an algorithm for determining the minimum resource requirements needed to guarantee a specified Quality of Service measure. We focused on bandwidth as the resource of interest, and average cell loss probability as the QoS measure of interest. It was also shown that other combinations of QoS measure (such as queuing delays) and adjustable resource (such as buffer size) can be similarly controlled and optimized.

This algorithm (referred to as REQS) determines the minimum steady state service rate which will satisfy the specified cell loss probability, for a given source. The main advantage of this approach is that essentially no traffic characterization of the source is required. This is important, considering the complexity of modeling and predicting source behavior. In addition, it eliminates the need to approximate (often very crudely) the resources needed to satisfy a QoS requirement. The proposed algorithm determined the minimum bandwidth necessary to satisfy a specified average cell loss probability, under a variety of different conditions. REQS converged quickly to the true "effective" rate in every case. REQS is simple to implement: only cell arrivals and losses need to be counted, and the rate adjustment calculation is very simple. In addition, the number of resource allocations required for convergence appeared to be very reasonable (less than 10 for most cases tested).

Simulation showed that the convergence time of the algorithm increased as the burstiness of the source increased, as the utilization was increased, and as the loss constraint was made more stringent. Convergence time also increased with the size of the queuing buffer, though the rate of increase was fairly slow. A key advantage of REQS is that it is robust; it converged to the correct rate for every one of a wide range of conditions, and with almost no knowledge of the characteristics of the source.

Several important open problems can be mentioned:

- Some users may not be willing to accept that initial (startup) QoS can be worse than requested; they want satisfactory QoS from the start, throughout. In addition, some network providers may not accept that resource allocation can increase during adaptation; there is no guarantee once a call is admitted that additional resources will be available at a later time. One way to address this is to execute the dynamic control algorithm on a "copy" of the traffic (as in Section 5.3) so that the actual bandwidth estimate is lowered only once (after the dynamic algorithm has converged). However, the same problems as outlined in Section 5.3 exist and need further study.

- In [20], we show a monotonic relation exists between the queuing buffer size and cell losses, similar to Theorem 1. Hence, REQS can be used to adapt the queuing buffer size to achieve a desired cell loss probability. Simultaneous control of both service rate and buffer size in order to satisfy multiple QoS constraints should be investigated.

- The performance of REQS on traces of actual source traffic needs to be evaluated. In addition, it will probably be necessary to modify REQS to work equally well for non-stationary traffic sources.

- REQS may need to be refined to improve its convergence time when very low loss probabilities are specified. One way to do this would be to estimate the rate requirement for very low loss probabilities by extrapolation using the rates determined for higher loss rates, or similar approaches based on fast simulation techniques [24].

- Theorem 1 is for a single queue only. In fact, it can be shown [20] that Theorem 1 does not hold for tandem queues. Extensions of this method to tandem queues would be very worthwhile.

- Formal proofs of convergence of REQS should be developed.

The work reported in this paper has shown that a simple dynamic technique based on measurements of actual quality of service can result in accurate control of the QoS, while using network resources efficiently. This technique can be used for a wide variety of traffic sources and QoS specifications.

References

[1] ATM Forum, "ATM User-Network Interface (UNI) Specification, Version 3.0, ", Prentice Hall, September 1993.

[2] P.P. Bhattacharya, L. Georgiadis, P. Tsoucas and I. Viniotis, "Ergodicity and Optimality of an Adaptive Multiobjective Algorithm," *Mathematics of Operations Research*, August, 1993, pp.705-740.

[3] A. Charny, "An Algorithm for Rate Allocation in a Packet-switching Network with Feedback," *Tech. Rep. MIT/TR-601*, MIT, Cambridge, May 1994.

[4] D.D. Clark, S. Shenker and L. Zhang, "Supporting real-time applications in an integrated services packet network: architecture and mechanism," *Proc. ACM SIGCOMM '92*, Aug.1992, pp.14-26.

[5] K.M. Elsayed and H.G. Perros, "The Superposition of Discrete-Time Markov Renewal Processes with an Application to Statistical Multiplexing of Bursty Traffic Sources," *Technical Report TR 94-10*, Dept of Computer Science, North Carolina State University, 1994.

[6] A.I. Elwalid and D. Mitra, "Effective Bandwidth of General Markovian Traffic Sources and Admission Control of High Speed Networks," *IEEE/ACM Transactions on Networking*, Vol.1, No.3, June 1993, pp.329-343.

[7] S.J. Golestaani, "A Framing Strategy for Congestion Management," *IEEE Journal on Selected Areas in Communications*, Vol.9, No. 7, Sept. 1991, pp.1064-1077.

[8] R. Guerin, H. Ahmadi and M. Naghshineh, "Equivalent Capacity and its Application to Bandwidth Allocation in High-Speed Networks," *IEEE Journal on Selected Areas in Communications*, Vol.9, No.7, Sep 1991, pp. 968-981.

[9] L. Gun, "An Approximation Method for Capturing Complex Traffic Behavior in High Speed Networks," *Performance Evaluation*, Vol 19, 1994, pp.5-23.

[10] I. Hsu and J. Walrand, "Dynamic Bandwidth Allocation for ATM Switches," Technical Manuscript, University of California, Berkeley, submitted for publication to the *Journal of Applied Probability*, February 1995.

[11] M. Izquieredo and D.S. Reeves, "Statistical Characterization of MPEG VBR Video at the Slice Level," *Proc of the Conf on Multimedia Computing and Networking*, SPIE, San Jose, Feb 1995.

[12] S. Jamin, P. Danzig, S. Shenker and L. Zhang, "A Measurement-Based Admission Control Algorithm for Integrated Services Packet Networks," in *Proc ACM SIGCOMM '95*, Boston, September 1995.

[13] Y.H. Jeon and I. Viniotis, "Achievable Loss Probabilities and Buffer Allocation Policies in ATM Nodes with Correlated Arrivals," *Proc. IEEE Intl Conference on Communications*, pp.365-369, 1993.

[14] C.R. Kalmanek, H. Kanakia and S. Keshav, "Rate controlled servers for very high-speed networks," *Proc IEEE GLOBECOM '90*, pp.300.3.1-300.3.9, December 1990.

[15] J.M. Karlsson, H.G. Perros and I. Viniotis, "Adaptive Polling Schemes for an ATM Bus with Bursty Arrivals," *Computer Networks and ISDN Systems*, vol.24, 1992, pp.93-103.

[16] R. Nagarajan, J. Kurose and D. Towsley, "Finite-horizon Statistical Quality-of-Service Measures for High-Speed Networks," Technical Manuscript, University of Massachusetts at Amherst, to appear in *Journal of High Speed Networks*.

[17] P.M. Morse, *Queues, Inventories and Maintenance : The Analysis of Operation Systems with Variable Supply and Demand*, Wiley, New York, 1958.

[18] A.K. Parekh and R.G. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks," *Proc, IEEE INFOCOM '93*, Mar. 1993, pp.521-530.

[19] C. Partridge, *Gigabit Networking*, Addison-Wesley, 1993.

[20] S. Rampal, "Routing and End-to-end Quality-of-Service in Multimedia Networks," *PhD Thesis*, North Carolina State University, August 1995.

[21] S. Rampal, D.S. Reeves, and D.P. Agrawal, "End-to-end QoS guarantees with statistical multiplexing in ATM networks," in *Performance Modeling and Evaluation of ATM Networks*, D.D. Kouvatsos ed., Chapman and Hall, 1995.

[22] S. Chong, S.Q. Li, and J. Ghosh, "Predictive Dynamic Bandwidth Allocation for Efficient Transport of Real-Time VBR Video over ATM," *IEEE J. on Selected Areas in Communication*, Vol.13, No.1, Jan 1995, pp. 12–23.

[23] C.-Y. Wang, D. Logothetis, K. Trivedi and I. Viniotis, "Transient Behavior of ATM Networks under Overloads", *Technical Manuscript*, Dept of Computer Science, Duke University, April 1995.

[24] Q. Wang and V. Frost, "Efficient Estimation of Cell Blocking Probability for ATM Systems, *IEEE/ACM Transactions on Networking*, vol.1, April 1993, pp.230-235.

[25] R.W. Wolff, "Stochastic Modeling and the Theory of Queues," Prentice Hall, 1989.

[26] H. Zhang and D. Ferrari, "Rate-controlled Static Priority Queuing," in *Proc IEEE INFOCOM '93*, pp.227-236, March 1993.

Appendix

Monotonicity of Cell Loss with Service Rate

Theorem 1 is proved here. We show that for an arbitrarily fixed sequence of arrivals all of the same length to a finite capacity queue, the total number of losses increases monotonically as the service rate of the queue is increased.

To prove the above statement, two queues Q_1 and Q_2 will be compared, each with a buffer of size B and service rates μ_1 and μ_2 respectively ($\mu_1 < \mu_2$). It will be shown that for any arbitrarily specified sequence of cell arrivals, the total number of losses due to buffer overflow at Q_1 can not be less than at Q_2.

The length of each cell represents the amount of work needed to service (transmit) this cell. Let this fixed amount of work be l. The total amount of remaining work at a queue at some instant is the sum of the lengths of waiting cells and the length of the untransmitted portion of the cell in service at that instant.

The following notation is used.

t_i : The arrival instant of the ith cell in the sequence.

a_i : The inter-arrival time between cell i and cell $i+1$.

W_j^t : The total remaining work at queue j, ($j = 1, 2$) at time t.

Q_j^t : The length of queue j, ($j = 1, 2$), at instant t.

$W_j^{t_i}$ is hence, the total remaining work at queue j seen by cell i upon arrival (not including the work corresponding to the ith cell itself) and $Q_j^{t_i}$ is the number of cells in queue j seen by cell i upon arrival (not including itself).

The total number of losses at each queue will be examined over time, starting with empty queues at $t = 0$. Whenever an arriving cell is lost at one queue but not at the other, the difference in the total losses seen so far at the two queues changes. Such a cell loss will be denoted as a unique cell loss (or UCL). It is shown that the total number of UCLs at Q_1 is at least as much as the number at Q_2 for any sequence of arrivals, thereby proving the result. This will be shown as follows. First it is shown that starting with empty queues at $t = 0$, Q_1 is the first queue to see a UCL. Next it is shown that whenever a UCL at Q_2 does occur, the following UCL must be at Q_1. In contrast after a UCL at Q_1 the next UCL may still be at Q_1. Hence, starting with empty queues at $t = 0$, at no time will the number of UCLs at Q_1 be less than that at Q_2. Thus the total number of losses at Q_1 can not be less than at Q_2.

First a basic result is proven, which shows that if at any arrival instant the remaining work at Q_2 is observed to be no more than that at Q_1, then the next UCL to occur will be at Q_1. First consider a sequence of arrivals which are not lost at either queue. Lindley's recursion for queue evolution [25] can be re-written here as

$$W_j^{t_i+1} = [W_j^{t_i} + l - a_i \mu_j l] \qquad i = 2, \ldots n \qquad j = 1, 2 \tag{2}$$
$$[X] = max(0, X)$$

Let, at the arrival instant of some cell i, the remaining work at Q_1 be at least as much as that at Q_2 (i.e. $W_1^{t_i} \geq W_2^{t_i}$). Since, $\mu_1 < \mu_2$, from the above relation (Eqn 2) we get $W_1^{t_i+1} \geq W_2^{t_i+1}$. The very first cell arrival sees no outstanding work at either queue. Hence $W_1^{t_1} \geq W_2^{t_1}$ is trivially true. Consequently, by induction, $W_1^{t_i} \geq W_2^{t_i}$ for all i as long as no cells are lost. Hence, if at some arrival instant, the remaining work is less at Q_2, then the remaining work at Q_2 continues to be less than at Q_1 for future arrivals as long as no cells are lost. Since $Q_j^{t_i} = \lceil (W_j^{t_i}/l) \rceil, j = 1, 2$, we also have $Q_1^{t_i} \geq Q_2^{t_i}$ for all cells i as long as no cells are lost. This means that starting from any time when the remaining work at the faster queue is no more than at the slower queue, the queue length seen by an arriving cell at the slower queue (Q_1) will always be at least as much as that at the faster queue (Q_2) as long as there are no losses. Hence, an arriving cell which sees a full buffer at Q_2 must also see a full buffer at Q_1 (but not necessarily vice versa). Hence, after any instant at which the remaining work at Q_1 is at least as much as that at Q_2, the first UCL to occur must be at Q_1.

A consequence of this basic result is that starting with both queues empty, the first UCL must occur at Q_1 (the slower queue) (since $W_1^{t_1} \geq W_2^{t_1}$ is trivially true).

We are interested in UCLs at Q_2 since this is the only way that the total number of losses at Q_2 can exceed that at Q_1. We now show that if at all a UCL occurs at Q_2, the next successive UCL must be at Q_1.

If the first UCL (at Q_1) occurs at time t, and at time t^+ the queue length at Q_2 is less than that at Q_1 (i.e. $Q_2^{t^+} \leq Q_1^{t^+} - 1$), then the remaining work at queue Q_2 at time t^+ will also be less than at Q_1. (This is because $W_2^{t^+} \leq l * Q_2^{t^+}$ while $W_1 > l * (Q_1^{t^+} - 1)$. In this case, from the above result, the next UCL will again be at Q_1.

Hence, the only way a UCL at Q_1 can cause a UCL at Q_2 later, is if at t^+, both queues have B cells. At t^+, the remaining work at Q_2 can exceed that at Q_1 by at most the length of one cell i.e. l, (depending on the difference in the amount of service so far received by the cells in service at Q_1 and Q_2). Since Q_2 has a lower service time, at most one cell can be transmitted by Q_1 before a cell leaves Q_2. In other words, at most one cell can be accepted at Q_1, while Q_2 is still full, resulting in a UCL at Q_2. After this UCL however, the next departure from either queue must be from Q_2 (since the service rate of Q_2 is higher, we cannot see two successive departures from Q_1 before a departure from Q_2). After this departure from Q_2, the queue length at Q_2 will become strictly less than that at Q_1 (which will be full). If $Q_2^t < Q_1^t$, we must have $W_2^t < W_1^t$. Hence, the remaining work at Q_2 will again be less than at Q_1. (A consequence of the relation $Q_j^t = \lceil W_j^t / l \rceil$). Since the remaining work decreases faster for the faster queue, even at the arrival instant of the next cell, the remaining work at Q_2 will be less than that at Q_1. From the above arguments hence, after a UCL at Q_2, the next UCL will again be at Q_1.

The above arguments have shown that starting with empty queues, the first UCL to occur will be at Q_1. Further, after any UCL at Q_2, the next UCL must be at Q_1. By induction over time hence, the total number of UCLs at Q_1 is never less than the number at Q_2. Hence, the total number of losses must also be greater at Q_1, the "slower" queue. □