Report Number: WUCS-97-06

1997-01-01

Design and Implementation of a New Connection Admission Control Algorithm Using a Multistate Traffic Source Model

Robert Engel

This report develops a practical method for connection admission control in ATM networks. The method is based on a virtual cell loss probability criterion, is designed to handle heterogeneous traffic types and allows each traffic source to be described by an individual finite-state model with as many states as are needed to describe the source traffic. To make connection admission decisions with respect to individual links, an aggregate finite state model is computed from the individual models and used to estimate the virtual cell loss probabilities. To reduce the computational requirements for maintaining the aggregate traffic model, the aggregate... Read complete abstract on page 2.

Design and Implementation of a New Connection Admission Control Algorithm Using a Multistate Traffic Source Model

Robert Engel

Complete Abstract:

This report develops a practical method for connection admission control in ATM networks. The method is based on a virtual cell loss probability criterion, is designed to handle heterogeneous traffic types and allows each traffic source to be described by an individual finite-state model with as many states as are needed to describe the source traffic. To make connection admission decisions with respect to individual links, an aggregate finite state model is computed from the individual models and used to estimate the virtual cell loss probabilities. To reduce the computational requirements for maintaining the aggregate traffic model, the aggregate model uses quantized data rates and is maintained incrementally using direct numerical convolution. The approximations required by the quantization process can be done in a strictly conservative way.

Design and Implementation of a New Connection Admission Control Algorithm Using a Multistate Traffic Source Model

Robert Engel

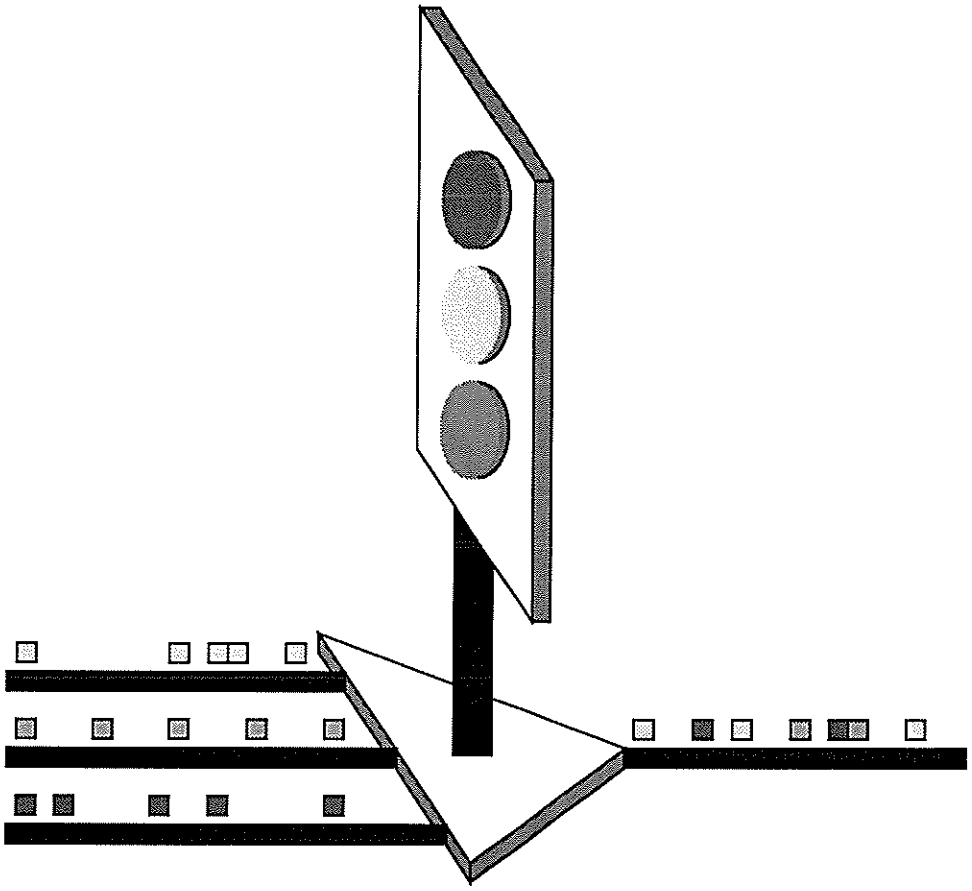
WUCS-97-06

January 1997

Department of Computer Science
Washington University
Campus Box 1045
One Brookings Drive
St. Louis MO 63130

A project presented in partial fulfillment of the degree Master of Science in Computer Science.

Design and Implementation of a New Connection Admission Control Algorithm Using a Multistate Traffic Source Model



Robert Engel

WASHINGTON UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE

Design and Implementation of a New Connection Admission Control Algorithm Using a Multistate Traffic Source Model

by

Robert Engel[1]

Advisor: Dr. Jonathan S. Turner

In partial fulfillment of the requirements for the degree of

MASTER OF COMPUTER SCIENCE

December 1996

St. Louis, Missouri

1. This project was sponsored by the National Science Foundation of Switzerland

Abstract

This report develops a practical method for connection admission control in ATM networks. The method is based on a virtual cell loss probability criterion, is designed to handle heterogeneous traffic types and allows each traffic source to be described by an individual finite-state model with as many states as are needed to describe the source traffic. To make connection admission decisions with respect to individual links, an aggregate finite state model is computed from the individual models and used to estimate the virtual cell loss probabilities. To reduce the computational requirements for maintaining the aggregate traffic model, the aggregate model uses quantized data rates and is maintained incrementally using direct numerical convolution. The approximations required by the quantization process can be done in a strictly conservative way.

Table of Content

Table of Figures

1 Introduction

The Asynchronous Transfer Mode (ATM) is the designated technology with which the Broadband Integrated Services Digital Network (B-ISDN) is going to be realized. One of the advantages of B-ISDN is that video, data, and audio all can be transferred within the same network. Since all applications can have very different requirements for the Quality of Service (QoS) and for the amount of bandwidth they use at any moment, it is crucial to have a mechanism that guarantees that all multiplexed connections on a link get the performance they negotiated with the network.

The Connection Admission Control (CAC) is responsible for this task. The CAC is defined by the ATM Forum [16] as follows (and a similar definition is used by ITU-T [17], the other principal standardization body for ATM): "*The Connection Admission Control function is defined as the set of actions taken by the network at SVC establishment or by Network Management during PVC establishment in order to determine whether a connection can be progressed or should be rejected.... Based on the CAC function a connection request is progressed only when sufficient resources are available at each successive network element to establish the connection through the whole network based on its service category, traffic contract, and QoS and in order to maintain the agreed QoS of existing connections.*"

The ATM Forum currently defines six Quality of Service parameters that correspond to a network objective: The Peak-to-peak cell delay variation (CDV) limits the jitter of arriving cells. The Maximum Cell Transfer Delay (CTD) gives an upper bound for the time cells need to cross a network. The cell loss rate measures the percentage of cells that get lost during transmission. The Cell Error Ratio (CER), Severely Errored Cell Block Ratio (SECBR), and the Cell Misinsertion Rate (CMBR) are other, less frequently used parameters that deal with errors happening during the transmission of ATM cells.

To relate traffic characteristics and QoS requirements to network behavior, the ATM Forum has defined 5 service categories that are divided in real time services and non real time services. In general, functions such as routing, CAC, and resource allocation are structured differently for each service category. The real time services are Constant Bit Rate (CBR) traffic, and real time Variable Bit Rate (rt-VBR). The three non real time traffics are nrt-VBR, Unspecified Bit Rate (UBR), and Available Bit Rate (ABR) traffic.

1.1 Requirements for a Connection Admission Control Algorithm

The central objective of traffic management in ATM networks is to provide virtual circuits that offer consistent performance in the presence of stochastically varying loads on the network. On the other hand, network providers want to utilize their links as much as possible while guaranteeing this performance. Therefore, a CAC has to fulfill the following requirements:

- It should provide consistent performance for applications that require it. Specifically, when a new connection is accepted, it should obtain the required Quality of Service (QoS) while not deteriorating the QoS of already existing connections.
- High network throughputs should be possible even in the presence of bursty traffic streams. To have high throughputs statistical multiplexing is used. However, strict (statistical) limitations on the minimum level of QoS parameters like the cell loss rate must be guaranteed.
- The CAC algorithm should be scalable. This means that independently of the number of connections already multiplexed on a link, the processing time to decide if a new connection can be accepted or not should be constant.
- In order to have the necessary flexibility, the CAC algorithm should not be based upon a fixed set of traffic classes. Rather, traffic characteristics should be described by means of parameters. This guarantees that future applications can also be described without having to change or add traffic classes.
- The CAC algorithm must be simple enough to be implemented in real time. Typically, the decision whether to accept a new connection or not must be made within a very short interval of time (<

1ms). This restriction prohibits long mathematical calculations to be performed for each connection.

- A 'weak' requirement is the tunability of the algorithm. It should be possible to increase/decrease the precision of the algorithm by allocating more/less calculation power. The motivation for this is that on high bandwidth switches the amount of traffic that can not be accepted due to CAC imprecisions could be diminished by using a faster processor. On the other hand, lower bandwidth switches may use less processing power to save costs.

1.2 Related Work

Several CAC algorithms based on the cell loss rate already exist. A conventional approach [5],[6] is to calculate the buffer overflow probability for a given buffer size. With this approach a model has to be selected that describes how cells arrive. Also for each source, the maximum burst length has to be announced, which is normally difficult to estimate. The proposed approach is independent of any arriving model and does not need to know the burst size.

The equivalent capacity method [7],[8] consists in determining a capacity with which a connection can be represented. The capacities of multiplexed connections are simply added and a new connection is rejected if the total (equivalent) capacity is greater than the bandwidth. However, this method overestimates the needed bandwidth when the number of multiplexed connections is large. Therefore, for a large number of connections a complementary, stationary approach is used that assumes that the aggregate bandwidth is Gaussian. The aggregate bandwidth is calculated based on average and mean bandwidth of the individual connections. Especially in the stationary case, the approach presented in this report which describes traffic sources with a multistate bandwidth model is much more precise.

Dynamic CACs [9],[10] measure the effective traffic. The aggregate traffic descriptor is then estimated based on the received parameters and on the effective traffic flow. If the measured traffic is smaller than it would be according to the individual traffic descriptors, more connections are allowed to be multiplexed. This is a good method to achieve a high link utilization when traffic parameters are likely to overestimate the real traffic. However, if a connection sends less than declared, additional connections will be multiplexed. When the connection goes back to its full rate, extensive cell loss is the result.

Recently, neural networks [11],[12] have been proposed for acceptance control. If all connections can be described with a fixed set of classes then the neural network can be trained properly. However, it is unknown how to train a neural network so that it can perform a CAC when connections are described with a set of parameters that can take any value.

1.3 Investigated Approach and Organization of the Report

Instead of using a two state traffic source model characterized by average and maximum cell rate (as e.g. in [1],[4]), we use a general, multistate traffic source model because the traffic of many applications is too complicated to be described precisely enough with a two state (on-off) model. This traffic model is introduced in section 2.

To decide if a new connection can be accepted we have to keep track of the aggregate offered traffic. This aggregate bandwidth is calculated by convolving [2] the declared traffic rates of the multiplexed sources. As convolution is very time consuming we quantize the link rate in a fixed number of bins. In section 3 we develop the equations necessary for updating the aggregate offered traffic when connections are added and dropped.

Among all the QoS parameters that are currently defined the cell loss rate is probably the most important one. Cell delay and cell delay variation are created mainly in the buffers of the switches along the transmission path. Although switches and transmissions links get faster and faster, buffer sizes will not increase that much. Therefore cell delay and cell delay variations are less important issues and it makes sense to base a CAC algorithm on the cell loss rate parameter. However, another algorithm based on a different QoS parameter could certainly be integrated if necessary. In the suggested approach the virtual cell loss probability [3] is used as the CAC parameter. Section 4 shows how to calculate the virtual cell loss probability in constant time.

Since the number of states of a traffic source is critical to the processing time, section 5 shows a method how any many-state source can be approximated by a reduced set of states. This approximation is proved to be strictly conservative and an estimator is given to make the approximation as tight as possible.

In section 6 different problems are presented that arise with the implementation of the CAC algorithm. A way to limit errors due to imprecisions in calculation is shown. This section also presents a way to quantize traffic descriptors so that previous assumptions hold, shows how to limit the array that represents the aggregate traffic, and gives the time performance of the CAC algorithm.

In section 7 the performance of the CAC algorithm is shown for two real traffic descriptors, a bursty World Wide Web traffic and a less bursty JPEG encoded video traffic. A comparison is made with the classical model that describes traffic parameters in terms of peak cell rate, sustainable cell rate, and maximum burst size.

In section 8 a modified leaky bucket scheme is proposed that allows to do Usage Parameter Control (UPC) on multistate traffic descriptors.

Finally, section 9 summarizes the most important aspects of this new CAC algorithm and concludes the thesis.

2 Multistate Traffic Sources

2.1 Motivation for Multistate Traffic Descriptors

Figure 1 presents an example that motivates the introduction of multistate traffic sources.

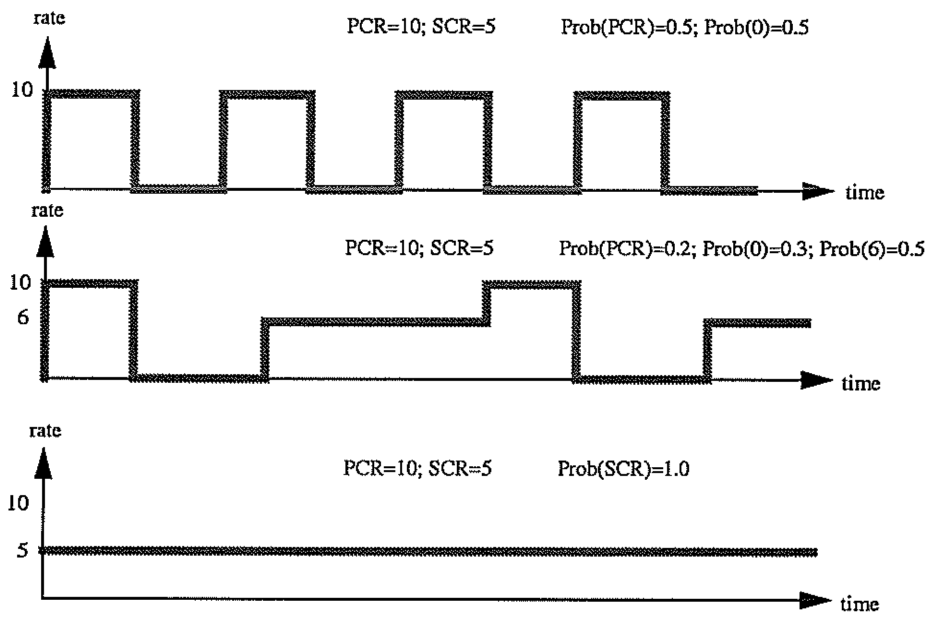


FIGURE 1. Three different traffic examples compliant with the same PCR, SCR values. Top: Most bursty behavior. Middle: A less bursty behavior. Bottom: Least bursty behavior.

It shows three possible concrete traffic streams that are compliant with the declared parameters Peak Cell Rate (PCR) and Sustainable Cell Rate (SCR)[1]. The top traffic characteristic shows the worst (most bursty) behavior of a traffic descriptor while still fulfilling the traffic contract. If we look at the rates in terms of probabilities we see that the probability to send at the peak cell rate is equal to the probability of not sending at all. Although the traffic at the bottom is not really a VBR traffic anymore but a CBR traffic, it certainly does not violate its traffic contract.

So at the top and the bottom we have the two extreme behaviors of traffic that can be described with the classical parameters. Of course there are many levels of 'burstiness' in between of which the middle traffic behavior just gives an example. Obviously, the cell loss due to periodical buffer overflow limits the number of connections we can multiplex on a link. It is very intuitive that the number of medium bursty traffic sources that can be safely multiplexed will exceed the number of burstiest traffic sources in some cases by a substantial amount.

Instead of looking at the traffic characteristics in terms of PCR and SCR let us now look at it in terms of rates and associated probabilities. For the top traffic the probability of sending at PCR (0.5) is equal to the probability of not sending any cells (0.5). For the middle traffic we have a different rate/probability distribution. Both the probability of sending at peak cell rate (0.2) and the probability of not sending (0.3) are smaller and there is a probability (0.5) of sending at a rate a little above the average (SCR) rate. It is enough here to motivate a traffic description with a multistate traffic model rather than the conventional, two state

1. These parameters are defined in [16]. PCR is the maximum cell rate at which the source is allowed to send. SCR is the (average) cell rate a source is allowed to maintain over a long period of time.

model. Therefore, if a traffic source is known to follow a behavior similar to that of the source in the middle it is preferable to represent it with the multistate rate/probability model, because this allows to use the link more efficiently while not exceeding the bound on the cell loss probability. Later sections will show this intuitive explanation with concrete examples.

2.2 Definition of a Multistate Traffic

Assume that every connection is described as a list of pairs that fulfill the following criteria:

$$(\lambda_1, p_1)\ ;\ (\lambda_2, p_2)\ ;\ (\lambda_3, p_3)\ ;\ \dots\ ;\ (\lambda_r, p_r) \qquad \text{where } 0 \le p_i \le 1 \text{ and } \sum_{i=1}^{r} p_i = 1$$

Here, p_i is the probability of a source to send data at the normalized rate λ_i (where λ_i are assumed to be ordered in increasing size). Such a description of a traffic may seem unusual since most of the time a traffic is described with peak rate, average rate and burst size (time or number of cells a connection is allowed to send at peak rate). Since the burst size is very hard to determine/estimate we do not want to take it into account for the new traffic descriptor. The worst case behavior (the most bursty behavior) of such a traffic is the on off source model, that assumes that sources send either at peak rate or not at all. The model presented here is more general in so far as several other rates between not sending and sending at the peak rate are possible, too. For a lot of applications the real behavior is more complex than that of an on-off model, so in general the multistate description is more precise. The proposed model also comprises the classic one: A source that is described by PCR, SCR is represented as the following (two state) multistate traffic:

$$p(0) = 1 - \frac{SCR}{PCR} \text{ and } p(PCR) = \frac{SCR}{PCR}$$

3 Incremental Link Rate Distribution Algorithm

To make decisions whether new connections can be accepted on a link, the aggregate traffic that is already on the link must be known. Basically, the aggregate traffic can be expressed in terms of rates and associated probabilities similar to individual traffic descriptors. To limit the memory space that is consumed by the aggregate traffic descriptor, as well as to allow efficient updating after connections have been added and dropped, the aggregate traffic descriptor is quantized to multiples of some basic rate (e.g. 1% of the link rate). The probabilities associated with the offered rates can then be maintained in an array *C*[*i*] where the index represents the quantized rate and the value of the field the associated probability.

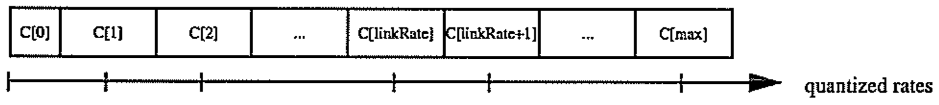


FIGURE 2. Representing the quantized traffic with an array. R=link rate.

Since the offered traffic might exceed the link rate the array has to contain more fields than just the range between 0 and the link rate. As the array has to be bounded the aggregate traffic can only be represented up to a certain maximum quantized rate (*max*). The array representing the aggregate traffic is shown in Figure 2. The effect of limiting the array to a maximum rate is explained in section 6.5

When several connections are multiplexed on a link, for every possible combination of their rates an associated probability can be calculated. An incremental algorithm is used to calculate the aggregate probability distribution. It is a generalization of the two-state model used in [1] to a multistate model.

Below there are two computationally efficient equations to calculate the aggregate traffic both when new connections are added to a link and when existing connections are dropped from a link. They are based on the assumption that not only the aggregate traffic is quantized but that all rates of the individual traffic descriptor are also a multiple of the same basic rate. Section 6.4 shows how a general traffic descriptor can be quantized to any basic rate. From now on all rates are assumed to be quantized. The aggregate traffic descriptor is therefore quantized in the same way. The rate/probability distribution of the quantized aggregate traffic descriptor is represented by $\{(0, C[0]), (1, C[1]), (2, C[2]), \ldots, (max, C[max])\}$ where $C[i]$ is the probability that the aggregate rate offered to the link is *i*.

$$C^*[i] = \sum_{k=1}^{r} C[i-\lambda_k]p_k \qquad C[i]=0 \text{ if } i<0 \qquad (1)$$

$$C[i] = \frac{C[i+\lambda_1] - \sum_{k=2}^{r} C^*[i+\lambda_1-\lambda_k]p_k}{p_1} \qquad C[i]=0 \text{ if } i<0 \qquad (2)$$

The above equations define how to update the aggregate traffic descriptor when a new *r*-state connection is added (1) or dropped (2), where $C[i]$ denotes the aggregate traffic descriptor without the considered connection and $C^*[i]$ the aggregate descriptor with the considered connection.

The new values of the array are calculated by starting at the top of the array (high *i*) for equation (1) and at the low end of the array for (2).

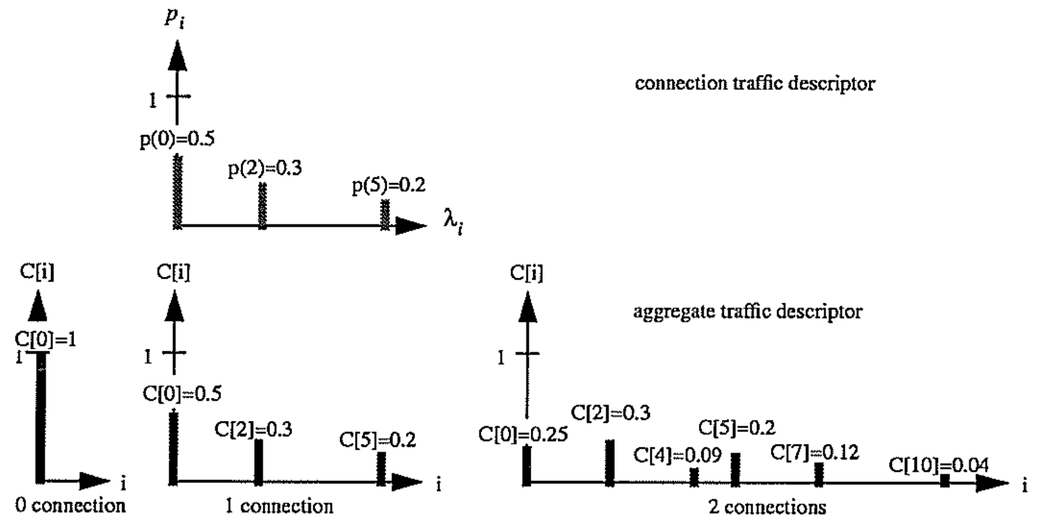


FIGURE 3. Updating the aggregate traffic descriptor when connections are added (connection: p(0)=0.5; p(2)=0.3; p(5)=0.2

In Figure 3 an example is given that illustrates the probability distribution of the aggregate traffic descriptor in its initial state, and after multiplexing a first and a second connection on the link. Note that the aggregate traffic descriptor is initialized with $C[0]=1$.

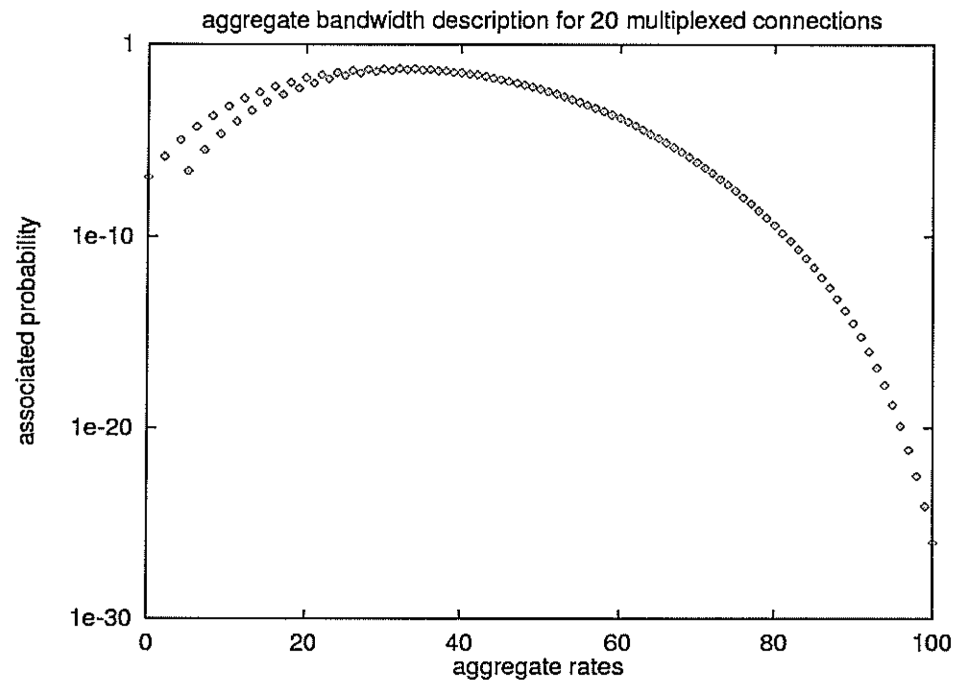


FIGURE 4. Aggregate traffic for 20 multiplexed connections of type p(0)=0.45; p(1)=0.3; p(3)=0.2; p(4)=0.05

Figure 4 gives an example of how the offered aggregate bandwidth distribution might look like after 20 homogeneous connections have been multiplexed on a link. Note that the probabilities for large offered rates can get extremely small.

4 Connection Admission Criteria

Among all the different parameters that define the Quality of Service of a traffic source the cell loss rate is probably the most important criteria. Since it is difficult to estimate the real cell loss directly a virtual cell loss algorithm is presented.

4.1 Virtual Cell Loss Algorithm

To predict the cell loss of the aggregate traffic descriptor and therefore the QoS, a logically bufferless fluid flow model is used that measures the virtual cell loss probability as suggested in [3]. This reference also shows that the virtual cell loss probability provides an upper bound for the real cell loss probability. The virtual cell loss provides a good estimate for the actual cell loss when peak virtual circuit rates are small relative to the link rate or when the actual buffer size is comparable in size or smaller than typical data bursts.

4.1.1 Definition of the Virtual Cell Loss Probability

$$P_v = \frac{totalOverflowBandwidth}{totalOfferedBandwidth} = \frac{\sum_{i>R} C[i]\,(i-R)}{\sum_{i=1}^{\infty} C[i]\,i} \qquad (3)$$

The virtual cell loss probability is defined in (3) where *R* stands for the link rate. As can be seen, direct computation from this equation is time-consuming since the number of terms *C*[*i*] for which *i*>*R* grows with the number of connections that are multiplexed on the link. Some simple mathematical transformations yield

$$P_v = \frac{\sum_{i>R} C[i]\,i - \sum_{i>R} C[i]\,R}{\sum_{i=1}^{\infty} C[i]\,i} = \frac{\mu - \sum_{i\le R} C[i]\,i - R\left(1 - \sum_{i\le R} C[i]\right)}{\mu} \qquad (4)$$

$$\mu = \sum_{k=1}^{N} \mu_k \text{ (total average traffic)} \qquad \mu_k = \sum_{i=1}^{r_k} \lambda_{ki} p_{ki} \text{ (individual average traffic of source } k\text{)} \qquad (5)$$

where μ stands for the total offered average traffic rate of the *N* connections multiplexed on the link. The total offered average traffic can be calculated by taking the sum of the individual average traffic rates as shown in equation (5).

The total processing time when connections are added/dropped is therefore the time needed to update the parameters *C*[*i*] (adding and dropping) plus the time needed to calculate the virtual cell loss probability (only when connections are added). If the maximum number of states of a connection and the maximum number of fields representing the aggregate traffic are limited, then the CAC algorithm runs in constant time. Section 5 shows how to reduce this number of states while still guaranteeing a conservative and precise traffic description while section 6.5 presents a way to keep the number of aggregate traffic bins constant.

4.1.2 Quality of the Method

Figure 5 shows how accurate the theoretical calculation of the virtual cell loss probability is. It depicts the virtual cell loss probability obtained by a simulation and the virtual cell loss probability obtained by equation (4) for different numbers of multiplexed video sources [13].

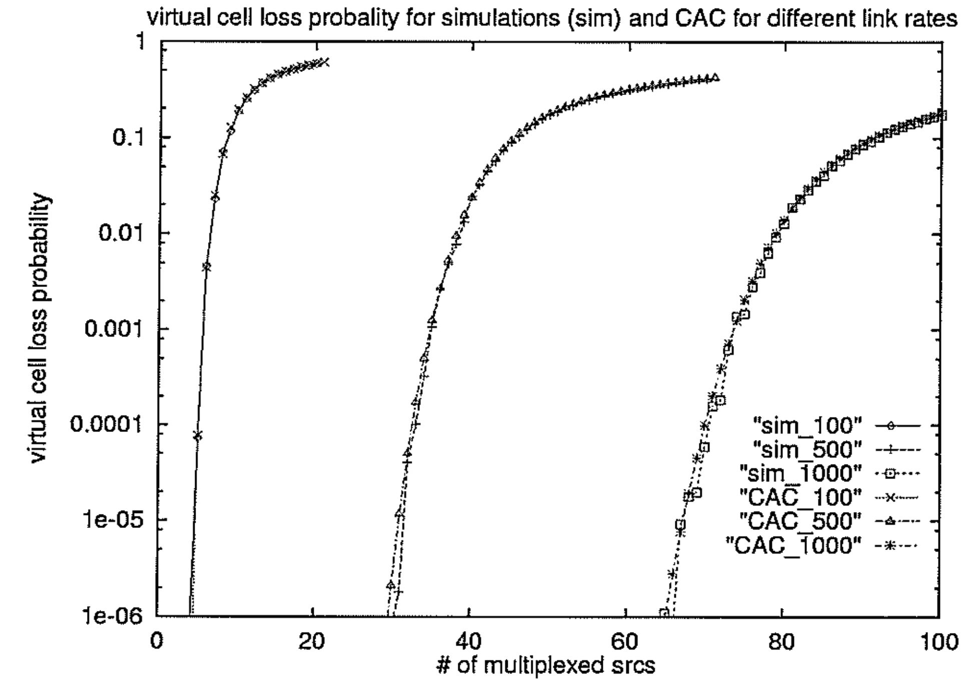


FIGURE 5. Cell loss probability estimated by CAC algorithm and by simulation for link rates 100, 500 and 1000 cells per millisecond. Traffic: JPEG encoded video.

4.2 Individual Virtual Cell Loss Algorithm

The previous section considered the virtual cell loss probability. Although the equations to calculate this probability are valid both for homogenous (all virtual circuit have the same traffic parameters) and for heterogenous traffic (all virtual circuits can have different traffic parameters) the cell loss probability calculated is an 'average' or 'total' cell loss probability. Let us consider an example where two connections A and B with different traffic parameters are multiplexed:

A: p(0)=0.9 p(10)=0.1

B: p(0)=0.5 p(5)=0.5

The following table shows the aggregate bandwidth probability distribution and the different cell loss probabilities.

TABLE 1. Multiplexing two different connections on a link

link rate R	p(0)	p(5)	p(10)	p(15)	(aggregate) cell loss prob	(individual) cell loss prob A	(individual) cell loss prob B
10	0.45	0.45	0.05	0.05	0.071	0.0373	0.1667

Clearly in this case the aggregate cell loss probability overestimates the individual cell loss probabilities. As every user is only interested in its own cell loss probability it might be worth while to look closer at the individual virtual cell loss probabilities.

4.2.1 Definition of the Individual Virtual Cell Loss Probability

The individual cell loss probability for a given connection (λ_i, p_i) with *r* states with respect to the aggregate traffic descriptor *C*[*i*] (does not contain the connection) is given by

$$P_{iv} = \frac{individualOverflowBandwidth}{individualOfferedBandwidth} = \frac{\sum_{i>R}\left(\frac{i-R}{i}\sum_{k=1}^{r} C[i-\lambda_k]p_k\lambda_k\right)}{\sum_{i=1}^{r}\lambda_i p_i} \tag{6}$$

Unfortunately, this is not a practical way to calculate the individual cell loss probability since there can be a potentially large number of fields above the link rate that are non zero. Since the calculation time should be small a better method to calculate the individual cell loss probability is needed.

4.2.2 Usefulness of the Individual Cell Loss Probability

There is an inherent problem in calculating the individual cell loss probability that is not solved just with an efficient algorithm to calculate it. Since on every link there can be potentially *N* connections with different traffic characteristics the individual cell loss probability for every one of them should be calculated when a new connection is added. This is due to the fact that even if a new connection's QoS (in terms of individual cell loss probability) could be satisfied, it might well be that the Individual Cell Loss probability of an already existing connection would not be satisfied once the new connection is multiplexed on the link.

In order to calculate the individual cell loss probability for every existing connection the connection first has to be dropped (so that the aggregate traffic is considered without it) and then the algorithm for the individual virtual cell loss probability has to be executed. Unfortunately, the time spent to update parameters (both for adding and dropping) is much larger than the time to e.g. calculate the virtual cell loss probability. To put this in perspective: It takes about 160 microseconds to calculate the virtual cell loss probability, but around 800 microseconds to update the parameters for a given set of CAC parameters. So, even with a very efficient algorithm the calculation time for the individual cell loss probability can take very long as every connection has to be dropped from the aggregate traffic once.

The only way out of this dilemma is to classify the different traffic parameters. If all traffic types can be distributed in a fixed number of classes then only the individual cell loss probability for every class has to be calculated. However, associating a traffic class with each traffic type is something that was to be avoided according to the specification of the CAC algorithm requirements. And, on the other hand, it is not obvious how traffic descriptors with different numbers of states and different rates and probabilities could be classified in a consistent way.

5 Approximating Traffic Descriptors With Reduced State Models

In the introduction it was stated that among other things a CAC algorithm has to run in real time.As seen in the previous sections, the calculation time of the CAC algorithm depends heavily on the number of states of the traffic descriptor. The calculation time is also inversely proportional to the precision of the CAC algorithm: The smaller the quantization level the finer is the granularity with which the aggregate traffic descriptor can be described, especially if traffic descriptors have small rates. And the more states there are the better the actual 'burstiness' of a traffic descriptor can be represented.

In the following it is shown how to transform a general r+1-state traffic descriptor into a reduced, a+1-state traffic descriptor with (a<r) that is guaranteed not to underestimate the original descriptor. Underestimating here means that the calculated cell loss probability of any aggregate traffic distribution together with the approximated descriptor may not be smaller than the cell loss probability of the same aggregate traffic together with the original descriptor. Thus, the approximated descriptor should be an upper bound on the real descriptor with respect to the burstiness. An estimator is presented to make this upper bound as tight as possible.

5.1 Conservative Approximation

Let us consider the following traffic that we want to approximate. For the sake of simplicity and without losing any generality we assume that for every rate in the range [0..r] a probability is associated (of course some of these probabilities can be equal to zero). We want to approximate this original descriptor $\{(0, p_0), (1, p_1), (2, p_2), \ldots, (r, p_r)\}$ with $\{(\sigma_0, q_0), (\sigma_1, q_1), (\sigma_2, q_2), \ldots, (\sigma_a, q_a)\}$ where (a<r). Figure 6 illustrates the strategy. At first, the range of bandwidth of the real descriptor is divided into a regions, ranging from rate 1 to r. Each rate/probability of the approximated descriptor approximates a region of the original descriptor so that the approximated rate is equal to the top rate in the region and the approximated probability is chosen so that the average original bandwidth of the region is equal to the approximated average bandwidth. A region s covering the range of bandwidth $(k, m]$ is therefore approximated as follows

$$\sigma_s = m \tag{7}$$

$$q_s = \frac{\sum_{j>k}^{m} j p_j}{\sigma_i} \text{ if (s>0)}, \quad q_0 = 1 - \sum_{j=1}^{a} q_j \tag{8}$$

First of all, note that the average bandwidth of the approximated and the original traffic descriptor are the same ($\mu_a = \mu_r$), and that the sum of all approximated probabilities is one.

To be conservative it must be possible to multiplex the approximated descriptor with any aggregate descriptor so that the cell loss probability (or the bandwidth overflow) is at least as big as when the same is done with the original descriptor. Suppose the aggregate traffic without the new descriptor is given by the coefficients $C[i]$. From (4) and (1) the difference between approximated and real cell loss probability, which must be non negative, can be calculated.

$$\hat{P}_v - P_v = \frac{\sum_{l>R} \sum_{i=0}^{a} plus(l + \sigma_i - R)\, C[l]\, q_i}{\mu + \mu_a} - \frac{\sum_{l>R} \sum_{i=0}^{r} plus(l + i - R)\, C[l]\, p_i}{\mu + \mu_r} \quad \text{where } \mu_a = \mu_r \tag{9}$$

$plus(i) = i$ if $i>0$ and 0 otherwise. To prove that (9) is greater or equal to zero note that the denominators are equal. Then split up the sum in terms for the different regions and prove that every tuple (σ_i, p_i) is an overestimation of the region it approximates, independently of $C[i]$.

$$\left(\sum_{i=0}^{a} plus(l+\sigma_i-R)\,C[l]\,q_i\right)-\left(\sum_{i=0}^{r} plus(l+i-R)\,C[l]\,p_i\right) \tag{10}$$

Now drop the coefficient $C[l]$, substitute $j = R-l$ and split up the two sums in the regions used for the approximation. Prove that for every value j and for every region s the approximation error is non-negative and therefore the total approximation error must be, too.

$$E_s^j = \left(\sum_{i=0}^{a} plus(\sigma_i-j)\,q_i\right)-\left(\sum_{i=0}^{r} plus(i-j)\,p_i\right) \tag{11}$$

Therefore equation (11) has to be greater or equal to zero for all values j and s.

$$E_s^j = \left(\sum_{i=0}^{a} (-j)\,q_i+\sum_{i=0}^{a} \sigma_i q_i-\left(\sum_{i=0}^{r} (-j)\,p_i+\sum_{i=0}^{r} ip_i\right)\right) = (-j+\mu_a-(-j+\mu_r)) = 0 \quad \text{for } j\le 0 \tag{12}$$

Equation (12) shows that only the case where j is positive needs to be considered as for negative j the approximation error is zero.

There are three cases to consider: Either j is to the right, to the left, or within the region. The following equations show the three cases.

$$E_s^j = 0 \quad \text{for } j>m \tag{13}$$

$$E_s^j = \mu_{as}-jq_s-\mu_{rs}+\sum_{i>k}^{m} jp_i = j\sum_{i>k}^{l} p_i\left(1-\frac{i}{\sigma_s}\right)\ge 0 \quad \text{for } j\le k \tag{14}$$

$$E_s^j = \sum_{i>k}^{j} ip_i+j\left(\sum_{i>j}^{m} p_i-\sum_{i>k}^{m}\frac{ip_i}{\sigma_s}\right) = \sum_{i>k}^{j} ip_i\left(1-\frac{j}{\sigma_s}\right)+j\left(\sum_{i>j}^{m} p_i\left(1-\frac{i}{\sigma_s}\right)\right)\ge 0 \quad \text{for } k<j\le m \tag{15}$$

Since the approximated bandwidth in a region is always the maximum bandwidth of the region the proposed strategy is clearly strictly conservative.

From equations (14) and (15) it can be seen that the approximation strategy can be improved while still guaranteeing a completely conservative approximation. The new strategy is the same for the top region. Seen from a lower region the top region approximation clearly overestimates the probability distribution (the greater the term in (14) the larger the overestimation). In a lower region this overestimation can be compensated by choosing the approximation bandwidth smaller than the top of the region. It is chosen as small as possible while still guaranteeing that the total approximation error represented by one term of (14) and possibly several terms from (15) is still positive. Figure 6 shows a r state traffic descriptor at the top. In the middle the descriptor is approximated with a regions, and the bandwidth within any region is replaced by a single rate at the top. At the bottom this overestimation of the burstiness is compensated by moving selected traffic rates to the left. The lowest region (with rate zero) is not considered as a regular region. It is only used to assure that the sum of the probabilities is one.

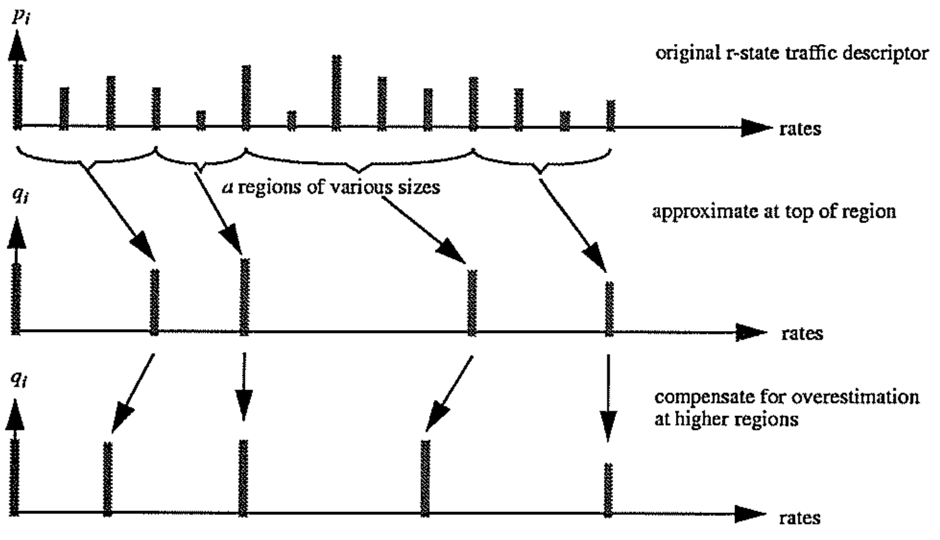

p_i
original r-state traffic descriptor
rates
a regions of various sizes
q_i
approximate at top of region
rates
q_i
compensate for overestimation at higher regions
rates

FIGURE 6. Approximating a real traffic descriptor by choosing a rate/probability for each region

5.2 Determining Optimal Regions

The preceding section does not give any hint how the regions have to be chosen in order to have an optimal approximation. We define the following estimator E of the quality of the approximation

$$E = \sum_{j>0}^{r} \sum_{s=1}^{a} E_s^j \qquad (16)$$

where E_s^j is defined by (13), (14), and (15) for the different cases of j.

E is in so far only an estimator for the quality and not a correct value as the summation is done over all the possible values of j. As soon as different values of j are considered the different values of $C[l]$ must be taken into account, since they are coupled with j. This is not possible, though, because the distribution of $C[l]$ is not known, respectively, the approximation should be independent of the distribution of the aggregate traffic. In our estimator, therefore, the assumption is made that all the $C[l]$ are equal.

In fact, the distribution of the $C[l]$ is generally even more favorable than that. Normally, the CAC operates in a range of $C[l]$ where the following condition is generally true: $C[l] > C[l+1]$ if both are different from zero. In addition to that, $C[l]=0$ if $l>maxRate$ where $maxRate$ is the sum of the largest rates of all descriptors. These two general properties of the distribution mean that the approximation error for high values of j (small values of l) are weighted more strongly. Since for high values of j only the upper regions contribute to an approximation error and overestimation is done in the upper regions and underestimation in the lower regions, this is a safe assumption.

Finding optimal regions consists in calculating E for all possible region combinations. The lowest value of E represents the best region distribution. Of course this calculation has to be done off-line, when the traffic descriptor for a certain application is determined. The CAC will then directly be fed with the approximated traffic descriptor.

5.3 Relaxed Approximation

The estimator of the previous section can be used for an even tighter approximation of a bandwidth descriptor, provided that the assumption about the distribution of the *C*[*l*] is correct.

Instead of guaranteeing that the approximation error is positive for every single value of *j*, equation (11) is relaxed as follows

$$E_s^{j+} = \sum_{i=j}^{r} \sum_{k=s}^{a} E_s^j \geq 0 \quad \forall j \; \forall s \tag{17}$$

Intuitively, the compensation in a lower region for the overestimation in a higher region is now less restrictive. It is defined to be correct if the total error is non negative for a value in the range from the current *j* up to the top region rather than assuring that the error is non negative for every single value of *j*. Again this is based on the assumption that due to the distribution of *C*[*l*] the higher region's (over)estimation error has a larger weight than the lower region's (under)estimation error.

To find out the optimum region distribution this approximation error has to be calculated for all possible region sizes. The region distribution with the lowest approximation error which is still greater or equal than zero is the optimal region.

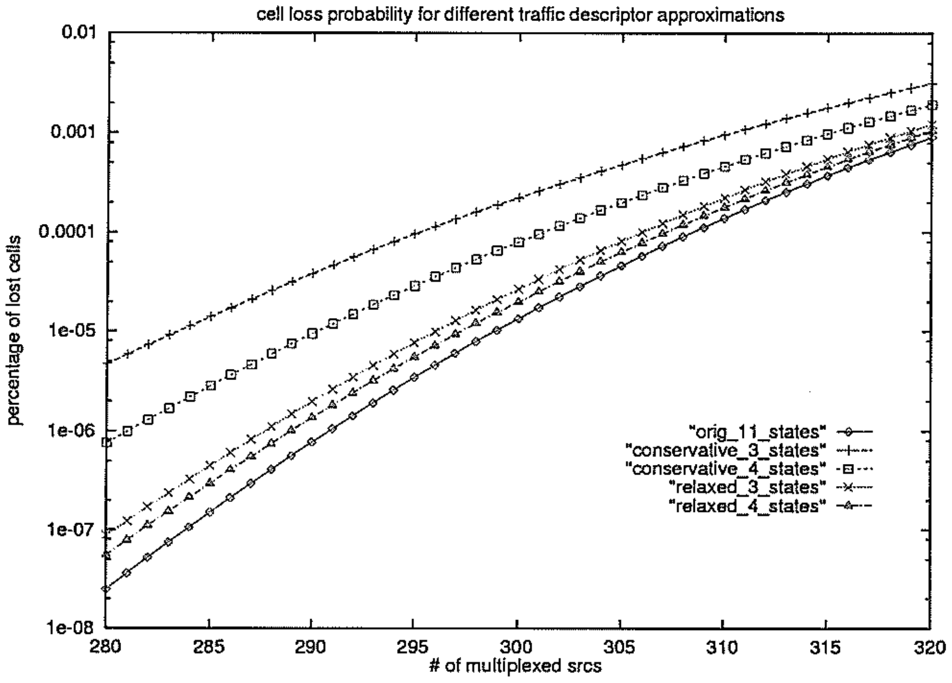


FIGURE 7. Approximating an 11-state traffic descriptor {(0,0.1) (1,0.2) (2,0.3) (3,0.09) (4,0.08) (5,0.07) (6,0.06) (7,0.04) (8,0.03) (9,0.02) (10,0.01)} with different strategies. Link rate = 1000

In Figure 7 we see how big the difference in terms of cell loss probability can be between the different strategies of traffic approximation. Note that even with the conservative four state model, the number of multiplexed sources for which the virtual cell loss probability is 10^{-6} is only about 3% less than the number of sources computed with the 11-state model.

To quantize the already approximated traffic descriptor according to the link rate and the quantization level either strategy can be used. The quantized states can be easily calculated on-line since the number of (already approximated) states is small. For details see section 6.4.

6 Implementation Issues.

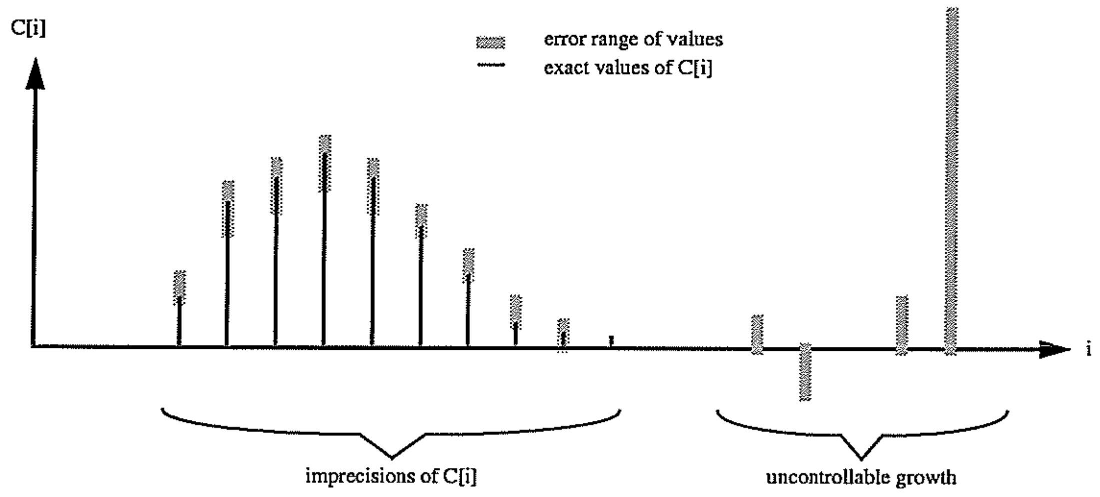


FIGURE 8. Schematic view of errors due to imprecisions caused by dropping of connections

Section 3 gave the equations for updating parameters when new virtual circuits are added or dropped. Unfortunately, there is an inherent computational problem in the calculation of the new parameters *C*[*i*] when a connection is dropped

First of all, note that the values of *C*[*i*] are very small when many connections are multiplexed on the link. This means that the precision of the computer quickly is smaller than the values some *C*[*i*] might have. In addition, probabilities are all in the range from zero to one. In the binary representation of the computer this means that most probabilities can only be approximated since the computer uses only (negative) powers of two. So even when the very first traffic descriptor is multiplexed on an initially empty link, the values *C*[*i*] can already be a little wrong. The following example shows why this can be a problem.

Consider a simple two state (on off) source that sends with probability p and at rate 1. Equation (2) that was used to update the aggregate traffic descriptor when a connection is dropped then simplifies to the following.

$$C[i] = \frac{C^*[i] - C[i-1]p}{1-p} \tag{18}$$

By solving the recursion which bottoms out at (*C*[*i*]=0; *i*<0) the following expression is obtained.

$$C[i] = \frac{1}{\bar{p}} \sum_{j=0}^{i} \left(-\frac{p}{\bar{p}}\right)^j C^*[i-j] \quad \text{where } \bar{p} = 1-p \tag{19}$$

If some value *C*[*i-j*] has an error this error is multiplied by $\left(-\frac{p}{\bar{p}}\right)^j$ every time a connection is dropped. Therefore if $\bar{p} < p$ the error will increase exponentially with j! Note also, that the error is worse at the high end of the array. For this simple example we can impose the restriction that $\bar{p} > p$ which should avoid the exponential increase of the error.

However, possibly the traffic descriptors have more than two states. Actually, the analysis for the multistate traffic descriptor is much more complicated and consists of several summands that contain $-\frac{p_i}{p_1}$ with different exponents. Therefore, even with the mentioned restriction the errors can still grow in an unbounded way.

The iterative computation of *C* does no only lead to imprecisions of coefficients that are non zero but also leads to uncontrollable growth of certain coefficients at the high end of the array that should be zero (but were not exactly zero due to imprecisions) and do now grow with every connection that is dropped. Figure 8 shows schematically the two types of errors. It is even possible to obtain negative values which does not make sense since negative probabilities are not defined.

6.1 Limiting the Error of the Coefficients *C*

A naive assumption would be that the aggregate bandwidth distribution has to follow some general pattern (for example monotonically decrease to the left and right of some 'middle' coefficient). However, Figure 9 shows that this assumption is not correct for the general case. In this example we have a mixture of different traffic sources initially. We then remove successively 1, 11, 21, 31 sources and print what the aggregate traffic descriptor looks like. Here it does not matter how the aggregate traffic is composed originally. The point is just that the aggregate traffic distribution can change from a Gaussian-looking distribution to a distribution with 'holes' in it, depending on what type of connections are left.

To calculate only the congestion probability rather than the virtual cell loss probability approximative equations could be used instead of the exact aggregate probability distribution. For such a case references [14] and [15] present upper bounds on the probability that an aggregate traffic descriptor exceeds a certain value (the link rate). However, the congestion probability is not an upper bound that is tight enough on the real cell loss rate. For the calculation of the virtual cell loss rate we need more information about the distribution of the traffic.

Another idea considers the 'extent' of the probability distribution. Clearly, the maximum index *maxNull* from where on all coefficients are zero (*C*[*i*]=0, *i*>*maxNull*) can be obtained by adding up the maximum rate of each connection. After updating the coefficients when a connection is dropped the coefficients above *maxNull* could be set to zero artificially.

Again, this does not help much, since there are also coefficients in between that should be zero. There is no way to find out where these coefficients are located and to set them to zero artificially.

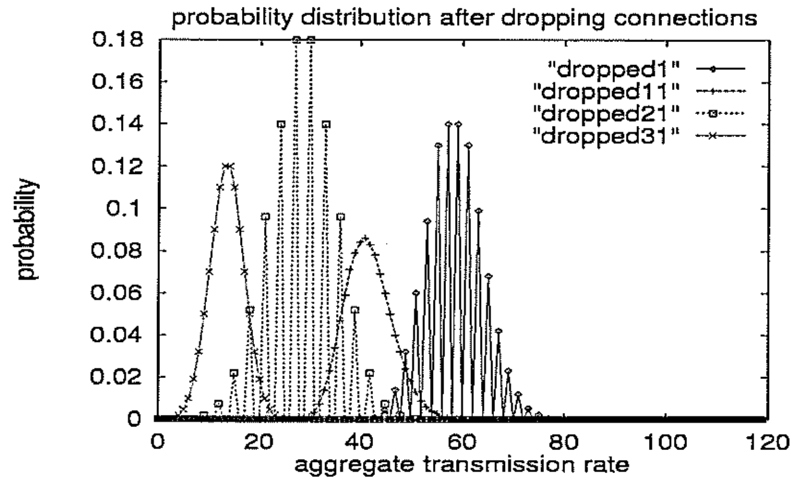


FIGURE 9. Probability distribution after dropping consecutive sets of connections

6.2 Deferring the Calculation of the Probabilities

Another idea is to look to keep track of how every coefficient changes its value when connections are added and dropped. For example, for a given state on the link, where four connections are multiplexed, a value *C*[*i*] might look like the following:

$C[i] = p_{11}p_{21}p_{32}p_{41} + p_{11}p_{23}p_{32}p_{43}$ where p_{ij} is the probability of connection i to be in state j (to send with rate λ_j.

More generally, each coefficient consists of a sum (containing zero or more summands) of a multiplication with N factors, where N is the number of connections multiplexed on the link.
What happens when a connection is added? The well known equation (1) tells that for every state k of the connection the summands that give the value $C[i-\lambda_k]$ have to be considered. Then they are multiplied with the probability p_k. Instead of doing the multiplication right away it is deferred and only the probability is maintained so that the information, on what to multiply, is not lost. This means that p_k is attached to the list of factors of every summand.

When connections are removed it is enough to remove the corresponding probability from the product chain. As the computational problem is generated by the division with a very small probability it can be eliminated completely by keeping for each coefficient C the product chains that contain only the values of the probabilities but are not evaluated.

Adding and dropping connections basically returns to adding and dropping probabilities in different lists. Only for the decision (if a connection to be added is accepted or not) this chain has to be evaluated.

Although this sounds nice, a quick numerical example shows the problem of deferring the actual calculation: Assume that there are 50 connections multiplexed on a link and each connection has 4 distinct bandwidths. This means that the total number of product chains equals to 4^{50}, a number much too large to be maintained. This explosion of the number of lists makes this method not usable.

6.3 Cumulative Probability

The cumulative probability $S[i]$ for a rate i of an aggregate traffic probability distribution is defined to be the probability that the aggregate traffic is smaller or equal than i:

$$S[i] = Prob(offeredRate \le i) = \sum_{j=0}^{i} C[j] \tag{20}$$

It is easy to verify that the updating of the cumulated probability when connections are added and dropped is completely analogous to the updating of the coefficients C.

$$S^*[i] = \sum_{k=1}^{r} S[i-\lambda_k]p_k \qquad \text{(adding)} \tag{21}$$

$$S[i] = \frac{S^*[i-\lambda_1] - \sum_{k=2}^{r} S[i+\lambda_1-\lambda_k]p_k}{p_1} \qquad \text{(dropping)} \tag{22}$$

Since the updating equations have not changed, the computational problem is of course still there. However, the cumulative probability distribution has certain properties that can be explored:

The cumulative probability grows monotonically and is bounded by 0 and 1 on the extreme sides:

$$0 \le S[i] \le S[i+1] \le 1 \tag{23}$$

When a new coefficient is calculated the following relation with respect to the values of the old coefficients holds:

$$S[i] \le \frac{1}{p_k} S^*[i+\lambda_k] \qquad \text{where } k \text{ is any state of the connection to be dropped.} \tag{24}$$

This can be seen easily by solving (21) for any $S[j-\lambda_k]$ and replacing $i = j - \lambda_k$.

Therefore, when a connection is dropped and the new S are calculated now there are upper and lower bounds for the new values. These can be used to modify the updating algorithm, to prevent errors from accumulating.

To reduce the errors further the probability associated with the smallest rate of each traffic descriptor must be greater or equal to one half. This might seem to reduce the flexibility with which traffic descriptors can be described. However, in most of the cases the lowest bandwidth has an associated probability that is normally greater than 0.6. In the other case a traffic descriptor has to be adapted so that this probability is large enough by changing the lower two rates and adapting their probabilities accordingly. To do this in a conservative manner, the same approach as in section 5 can be used.

All in all, there are three ways to reduce the calculation errors. It is recommended, to use all together and to use the default or more restrictive values when indicated.

- Equations (23) and (24) are used to give lower and upper bounds on the newly calculated values for the aggregate traffic distribution probability when a connection is dropped.
- To reduce the effect shown in equation (19) the probability associated with the smallest rate of each traffic descriptor must be greater than a certain value (recommended value: 0.6 or larger).
- The error in equation (19) does not only depend on the ratio of the probabilities but also on size of the exponent. The more bins (fields in the array) there are the larger is the error. Therefore it is recommended to use small arrays to represent the aggregate traffic (recommended value: 100 bins).

Of course, it is always possible to neglect the above recommendations and simply to recalculate the values of the aggregate traffic descriptor from scratch as soon as enough connections have been dropped, before the imprecisions become significant. This recalculation of values could be done by a low priority process in the background so that the CAC still has a good response time.

More details about calculation imprecisions and some concrete tests can be found in the Annex. It is especially interesting to compare the two examples of section 12.3.2. In these examples, for the same CAC parameters the calculation errors are determined for the multiplexing of the traffic descriptors (p(1)=0.6, p(2)=0.1, p(3)=0.3) and (p(0)=0.6, p(1)=0.1, p(2)=0.3). Although both traffic descriptors look very similar (the only difference is clearly in the rate) and exactly the same number of connections were dropped and added in both cases the first traffic descriptor did not produce any noticeable errors whereas the second traffic descriptor did. This shows that it is very hard to give general rules about the behavior of the errors and that many factors determine if the error is big or small.

6.4 Quantization of the Bandwidth

In section 3 the assumption was made that all traffic descriptors used as input for the CAC algorithm specify only rates that are a multiple of a basic rate. Since this basic rate depends on how many bins are used in the algorithm and on the link rate, no general basic rate exists that is the same for all networks. Therefore, when the rates for a certain application are determined it is unknown what the basic rate is and a quantization has to take place before the CAC algorithm can be used. This section shows how to quantize the rates of a connection that wants to be multiplexed on a link, so that the CAC algorithm can work correctly.

The general approach is analogous to the one that was used to reduce many state traffic descriptors to fewer states. The basic idea is shown in Figure 10.The easiest way to quantize rates is to put them always in the bin of the next higher quantized rate. This way, quantization is certain to happen in a strictly conservative manner. But, as explained in section 5, the traffic descriptor is made more bursty than it actually is.

Two special cases can happen, at the quantization:
If two or more declared rates are close together they might be quantized in the same bin. This can only reduce the number of states, therefore increasing the processing speed for this connection.
If all the rates of the descriptor are very small, the usual quantization puts them all in either bin zero or bin one of the aggregate traffic descriptor. Therefore, the quantization leads to an on off representation of the original traffic.

In order to quantize in a more precise way, the same model is used as with the reducing of multistate sources. Again the choice has to be made to choose either the conservative or the relaxed model. To be consistent with the notation (λ_i, p_i) denotes the non-quantized rate/probability and (σ_i, q_i) the values after quantization in the following sections.

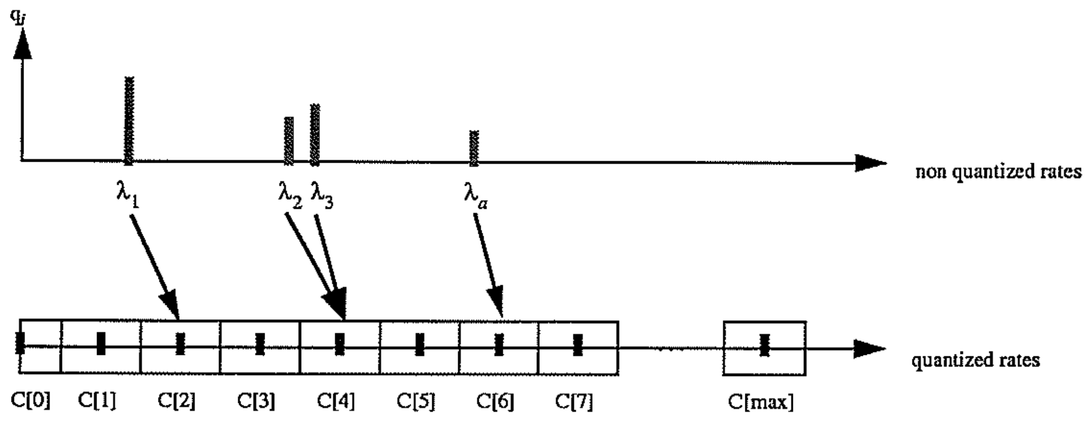


FIGURE 10. Quantization of a Traffic Descriptor by 'rounding up'

6.4.1 Conservative Quantization

For the quantization to be conservative the following equation must be non negative.

$$E_s^j = \left(\sum_{i=0}^{a} plus(\sigma_i - j)\, q_i \right) - \left(\sum_{i=0}^{r} plus(\lambda_i - j)\, p_i \right) \tag{25}$$

As seen, the quantization is certainly conservative if $\lambda_s \le \sigma_s$. Again, to have a better quantization, overestimation in higher regions are compensated by underestimation in lower regions as follows. Look at any region s for which there is a non quantized rate and a quantized rate (the region is here defined simply as the region between the non quantized rate and the next lower non quantized rate. The worst-case underestimation for this region is if $j = \sigma_s$ and if we have $\lambda_s > \sigma_s$.

For the quantization to be conservative the following must hold

$$-(\lambda_s - \sigma_s)\, p_s + \sum_{i>s} \left(1 - \frac{\lambda_i}{\sigma_i} \right) p_i \sigma_s \ge 0 \tag{26}$$

The quantization is therefore done by starting at the topmost region (where the quantization always is the ceiling of the non quantized value). For each lower region the quantized rate is chosen as low as possible without violating (26). In contrast to the state reduction model, the regions here are already given. The only degree of freedom is therefore to quantize as low as possible. This makes the calculation straightforward and efficient and does not take much processing time compared with updating parameters after adding/dropping a connection.

6.4.2 Relaxed Quantization

By modifying (17) the following equation serves as a starting point for this approach.

$$E_s^{j+} = \sum_{i=j}^{\sigma_a}\left(\sum_{k=s}^{a} plus(\sigma_k - i)\, q_i - \sum_{k=s}^{a} plus(\lambda_k - i)\, p_k\right) \geq 0 \tag{27}$$

Again for a region s for which there is a non quantized rate and a quantized rate the worst-case underestimation is if $j = \sigma_s$ and if $\lambda_s > \sigma_s$.

$$E_s^{\sigma_s +} = \sum_{i=\sigma_s}^{\sigma_a}\left(\sum_{k=s}^{a} plus(\sigma_k - i)\, q_i - \sum_{k=s}^{a} plus(\lambda_k - i)\, p_k\right) \geq 0 \tag{28}$$

$$\sum_{i=\sigma_s}^{\sigma_a} plus(x-i)\, y = \sum_{i=\sigma_s}^{\lfloor x \rfloor} (x-j)\, y = (\lfloor x \rfloor - \sigma_s + 1)\, xy - \frac{y(\lfloor x \rfloor + \sigma_s)\,(\lfloor x \rfloor - (\sigma_s + 1))}{2} \tag{29}$$

Using the property of (29) in (28) we obtain the following.

$$E_s^{\sigma_s +} = \sum_{i>s}^{a} (\sigma_i - \sigma_s + 1)\, \sigma_i q_i - \frac{(\sigma_i - \sigma_s)\,(\sigma_i - \sigma_s + 1)\, q_i}{2} - \left((\lfloor \lambda_i \rfloor - \sigma_s + 1)\, \lambda_i p_i - \frac{(\lfloor \lambda_i \rfloor + \sigma_s)\,(\lfloor \lambda_i \rfloor - \sigma_s + 1)\, p_i}{2}\right) \quad \text{since } \lfloor \sigma_i \rfloor = \sigma_i \tag{30}$$

Although equation (30) looks complicated, the CAC algorithm can perform the quantization very efficiently by applying the equation for the relaxed case since the equation is not complicated in terms of processing power and normally the quantized rates are at maximum one or two bins below the bin we would put them in by just rounding up.

Especially if the aggregate traffic descriptor is rather coarse grained and does not use many bins the relaxed quantization can be very helpful to improve the precision of the algorithm.

6.4.3 Quantizing Small Rate Traffic Descriptors With an On Off Model

If all the rates of the non-quantized traffic descriptor are very small there are two choices how to quantize: Either the quantized traffic descriptor is represented with an on-off model (thus reducing it to two states) or it is replaced with a constant bit rate traffic.

Clearly, when a many state descriptor is represented with an on off model its burstiness is increased. Therefore, this quantization is of a conservative nature. The question is if it is really necessary to use such a conservative approach. As with any other traffic descriptor, this quantized traffic descriptor forces the algorithm to update all the bins incrementally, both when it is first added and when it is later dropped. Since this traffic descriptor can have only a small average rate there need to be many connections of this type to use a significant portion of the link. As these sources are all independent they statistically have the tendency to even out. Therefore, a more economical approach (from the point of view of processing time) might be appropriate.

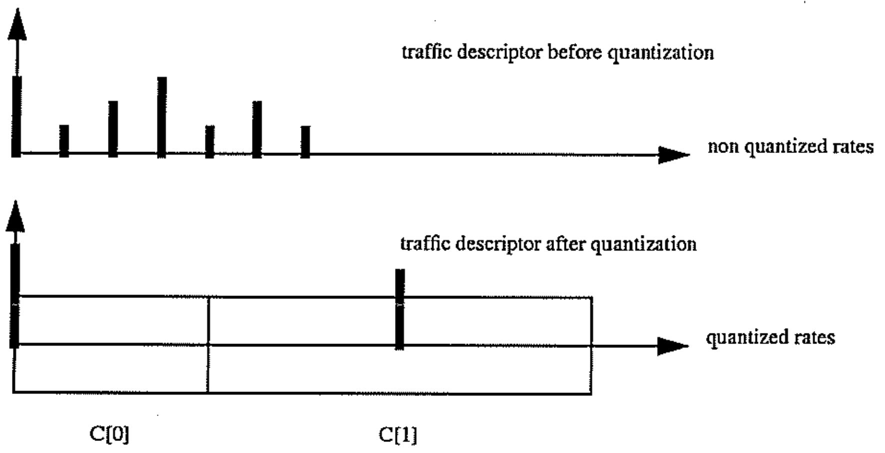

traffic descriptor before quantization
non quantized rates
traffic descriptor after quantization
quantized rates
C[0]
C[1]

FIGURE 11. Quantize a Small Rate Descriptor With an On Off Model

6.4.4 Representing Small Rate Traffic Descriptors with Constant Bit Rate Traffic

The intention of this approach is to reduce the processing time of 'relatively unimportant' small rate traffic descriptors needed to update the aggregate traffic bins when such connections are added or dropped and to avoid the imprecisions of calculations for these operations. Real traffic descriptors are therefore approximated by a CBR traffic descriptor, since there is no calculation involved in adding or dropping a CBR traffic descriptor as the fields in the aggregate traffic descriptor are only shifted left or right.

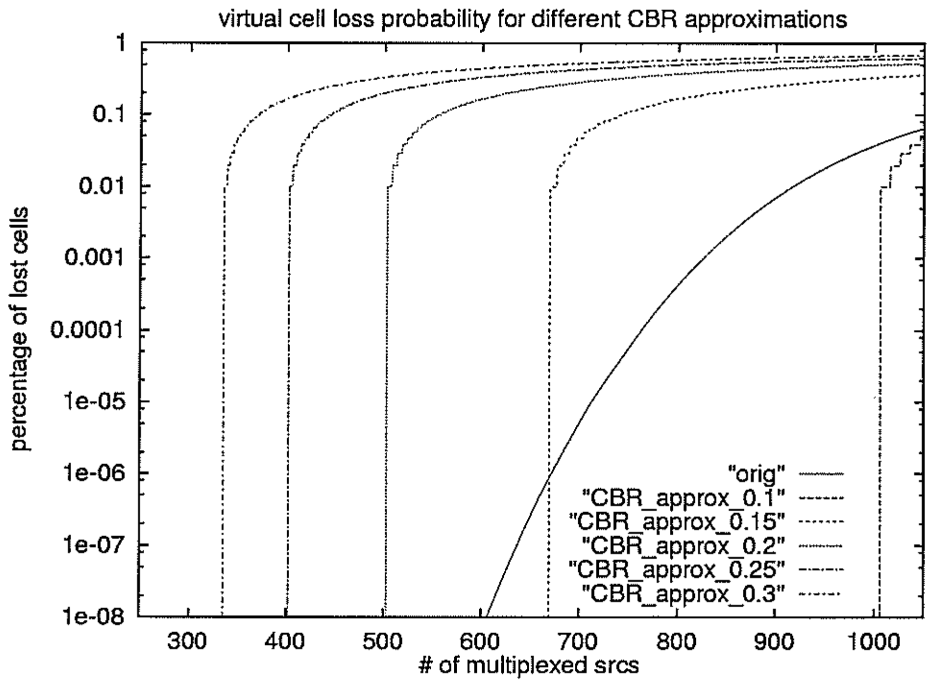


FIGURE 12. Virtual Cell Loss Probability for CBR approximations of different rates (0.1; 0.15; 0.2; 0.25; 0.3) of the following (orig) traffic descriptor: P(0)=0.9; P(1)=0.1. Link rate: 100. The link is loaded 100% if 1000 sources are multiplexed.

While the calculation of the virtual cell loss probability of a two state traffic descriptor has to follow equation (4), the calculation of the virtual cell loss probability of multiplexed CBR traffic descriptors is much easier.

$$P_v = \frac{\mu - R}{\mu} \quad \text{where } \mu \text{ is the total offered traffic} \tag{31}$$

Equation (31) shows how to calculate the virtual cell loss rate for an aggregate traffic consisting only of CBR sources. As long as the sum of all CBR rates do not exceed the link rate there is no cell loss at all. In the case of a CBR rate of 0.1 and a link rate of 100 the virtual cell loss probability is 0 for 1000 connections and jumps to 0.001 when the next connection is multiplexed on the link. In general, for a CBR traffic the cell loss rate is zero for a long time and suddenly jumps to a significant value when one single, additional connection is added. This is in contrast to the virtual cell loss probability for VBR connections that increase more gradually..

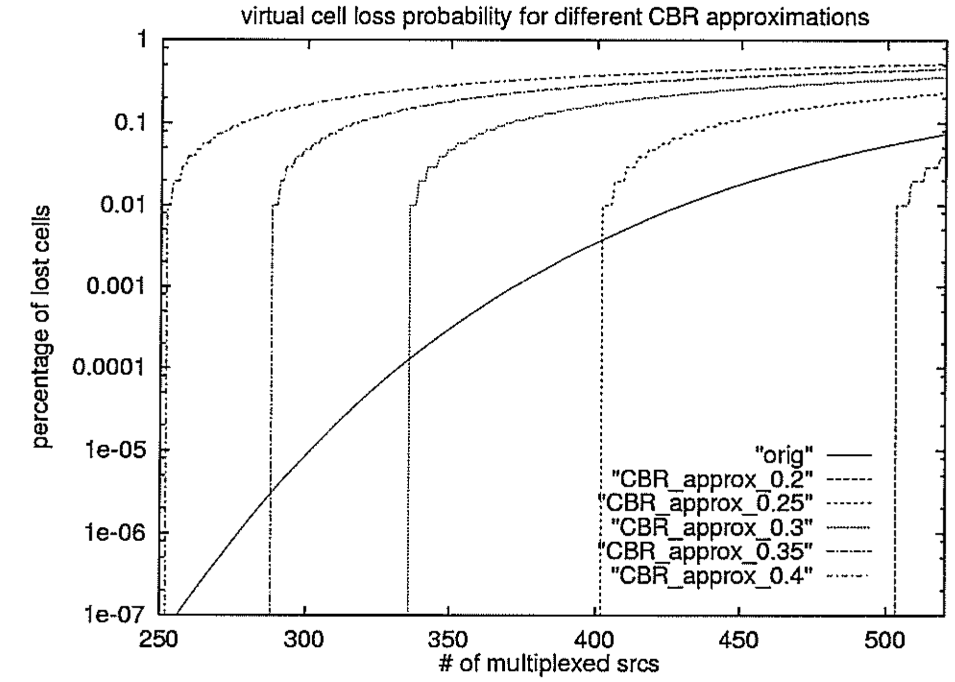


FIGURE 13. Virtual Cell Loss Probability for CBR approximations of different rates (0.2; 0.25; 0.3; 0.35; 0.4) of the following (orig) traffic descriptor: P(0)=0.9; P(2)=0.1. Link rate: 100. The link is loaded 100% if 500 sources are multiplexed

In order to have a good CBR approximation of small rate VBR traffic the above fact is important. In Figure 12 and 13 different approximations of a small rate VBR traffic with maximum rates 1 and 2 respectively are shown. For the mentioned reasons it is impossible to approximate the VBR traffic by a CBR traffic so that the respective cell loss probabilities are identical.

First of all, a CBR approximation has to be chosen whose rate is greater than the average rage of the VBR traffic to be approximated (this is due to the fact that the virtual cell loss rate of a CBR connections is greater than zero only when the offered average traffic is at least equal to the link rate, a case that never occurs in reality).

Secondly, the quality of the approximation depends on the target virtual cell loss rate of the CAC algorithm that is not to be exceeded. It is acceptable to underestimate the virtual cell loss rate of the aggregate traffic as long as it is still well below the target cell loss rate. However, once it gets close to the maximum allowed cell loss rate, the approximation should be an upper bound on the real virtual cell loss rate. For the example of at Figure 12 and a target cell loss rate of 10e-6 the CBR approximation with rate 0.15 intersects the original curve almost exactly at this cell loss rate. This means that as soon as the aggregate loss rate gets close or even above the target cell loss rate, the approximation becomes an upper bound. To have a safety margin a CBR approximation should be chosen that is even a little greater. The same reasoning is true in Figure 13 where a CBR rate of 0.4 would give us an upper bound on the real cell loss probability for the same target virtual cell loss rate.

Note that in the two cases above different CBR approximations are used. In the first case, the average rate of the original VBR traffic descriptor is multiplied by 1.5, whereas in the second case the multiplication factor is 2.0. This is due to the fact that the VBR traffic descriptors in both cases are different.

Since for the CAC algorithm it is not expected that the offered traffic only consists of these worst case traffic descriptors (extremely bursty, with the maximum rate still too small to be quantized in bin 1), but rather that they should be handled in an efficient, economical way the proposed CAC approximates these small rate VBR traffic descriptors with a CBR rate that is equal to the average rate of the VBR times a security factor.

For a more precise approximation the 'burstiness' of a VBR traffic descriptor would have to be considered the factor with which the average rate is multiplied would have to be larger, if the traffic is bursty and smaller if it is less bursty. It is not clear, however, how to map the 'burstiness' to a concrete multiplication factor.

6.5 Limiting the array *S*[*i*]

As stated in section 3 the processing time to update all bins of the aggregate traffic descriptor depends directly on the number of these bins. If the aggregate traffic is represented up to the highest offered rate, many bins above the link rate are needed (the uppermost bin would have to cover the rate of the sum of all the maximum rates of the individual traffic descriptors). To represent all the rates of the aggregate traffic adequately either the size of the bins (thereby reducing the granularity) or the number of bins has to be increased (thereby increasing the calculation time).

A better idea is to clip the representation at a certain maximum rate. For the following reason clipping does not impair the correctness of the algorithm very much. The probabilities of offered rates that are larger than the link rate must be extremely small. This is due to the fact that the congestion probability is an upper bound on the virtual cell loss probability. Since the congestion probability is simply the complement of the probability of the offered traffic at the link rate (1-*S*[*R*]), the remaining probability that can be distributed at rates above the link rate must be smaller than the specified maximum virtual cell loss rate allowed on the link. In addition, since the virtual cell loss probability grows if the aggregate rates are larger and larger, a configuration in which a rate that is high above the link rate has a rather big associated probability is simply not possible because the CAC algorithm would have rejected it.

Consider equation (2) which is used to update the aggregate traffic parameter when connections are dropped. In order to calculate the new value *C*[*i*] the old value *C*[*i*+*x*] needs to be accessed where *x* is the minimum rate of the connection. Here is the problem. In order to update the topmost element of the array (*C*[*max*]) the element *C*[*max*+*x*] needs to be accessed, which is not possible since the array is confined to the range [0..*max*].

Therefore an interpolation is suggested to get hold of these values that are not represented anymore: It is easy to keep track of the topmost rate of the aggregate traffic whose associated cumulative probability is not one yet (this is simply the sum of all maximum quantized rates of the individual sources). Suppose the next higher quantized rate is called *top*. Now try to interpolate between the two values of the cumulative probability *S*[*max*] and *S*[*top*] which are known.

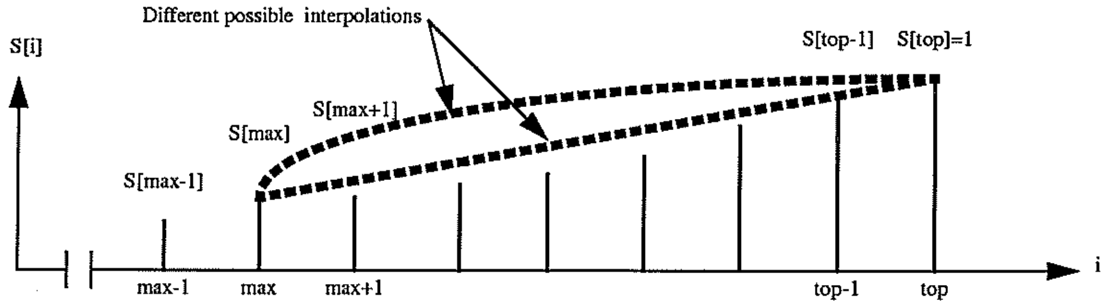


FIGURE 14. Interpolation of array elements between two known values

Although there are different kinds of interpolations possible a linear interpolation is chosen for simplicity. Since the values are very close to one anyway it is not necessary to consider a more complicated type of interpolation. Since the virtual cell loss probability is calculated using only array elements in the range [0..*R*] the interpolated values do not need to be considered as long as at least *R* bins are used in this range. Since normally some additional bins above the link rate are used the values of *S* for which an interpolation is necessary are even closer to one.

6.6 Time Performance

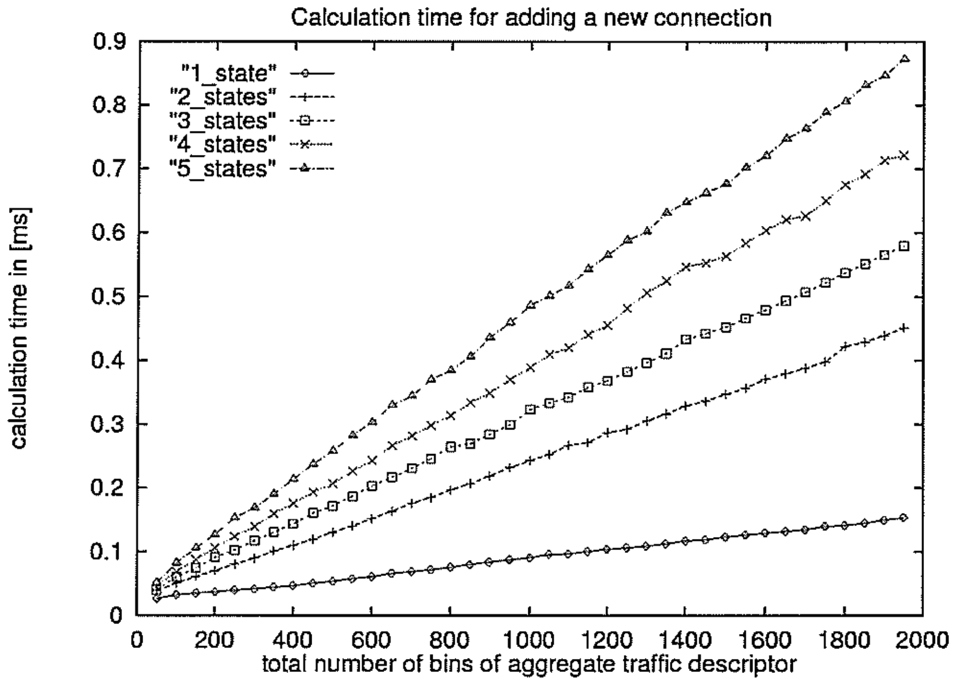


FIGURE 15. Calculation time needed to add a new connection (make a decision + update the parameters) for traffic sources with 1,2,3,4,5 states.

Previous sections pointed out that the processing time to add a new connection or drop an existing connection depends both on the number of states of the connection and of the number of bins used to represent the aggregate traffic. The following two figures plot the actual time needed for the two operations and the influence, the two parameters have. The measurements were made on a SPARCstation 5.

Figure 15 shows the time needed to add a new connection. This time is composed of accepting the new connection (determining if the virtual cell loss probability is not exceeded) and to actually update the parameters of the aggregate traffic descriptor. Note that the case where the traffic source has only one state (CBR traffic) takes significantly less time, since no actual calculations are performed to update the aggregate traffic descriptor. Instead, updating the traffic descriptor consists only in shifting the values of the fields to the right. This makes the suggested approach of section 6.4.4 to approximate small rate traffic descriptors by a CBR traffic all the more attractive. Figure 16 shows the time needed to drop a connection. Here the calculation time consists merely in updating the parameters of the aggregate traffic descriptor. Again, the case where we have a one state CBR traffic source, takes significantly less time. We see, that dropping a connection takes about 1.2 times longer than adding a connection. This is due to the fact that the equation to update the aggregate traffic descriptor in the dropping case (2), involves division, multiplication, and addition, whereas for the adding case only multiplication and addition is done.

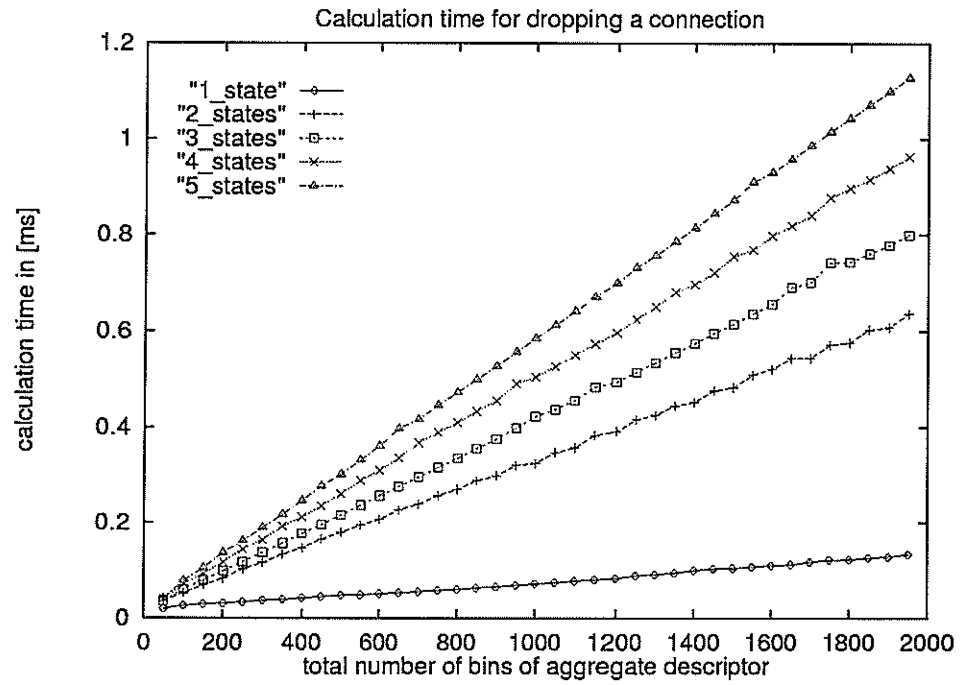


FIGURE 16. Calculation time for dropping a connection (updating the aggregate traffic descriptor) for traffic sources with 1,2,3,4,5 states.

7 Real Traffic Descriptors

A lot of CAC algorithms might work well in theory, but with real traffic examples their performance suddenly drops. This section shows how well the proposed CAC algorithm performs on two real examples of traffic. But first of all, several ways are outlined how to determine the multistate traffic model of a service/ application. This can be done in several ways:

One way is to measure or determine directly the rates at which a specific traffic source sends. For certain applications there might be a theoretical model according to which the traffic source behaves. Together with the possibility that a shaper is used at the border to the ATM network the traffic model for the source can be determined.

Another way is to determine the traffic model statistically. Let the source send for a certain time period and count the number of cells received in an interval. When the number of received cells is divided by the interval length we get the average rate at which the source has sent. From the frequency of the appearing of a certain number of cells in an interval we can conclude the probability of the corresponding rate. The question is of course what the size of the interval should be. If it is very long, only a long time average of the source is measured. If it is too short, never more than a few cells might be counted in the same interval which would give an on off behavior for the source. Again, with some external knowledge about the source it should be feasible to choose an interval that allows to catch enough cells arriving consecutively at the source's maximum speed to gather some meaningful statistics.

7.1 JPEG Encoded Video Traffic

The first traffic to be considered is the traffic generated by a JPEG video source [13]. The original traffic descriptor was obtained by measuring the ATM traffic at the output of an ATM JPEG CODEC using a Q factor of 30.

7.1.1 Traffic Model

To determine the statistical mode of this source we counted the number of ATM cells received in consecutive intervals of 1ms. An excerpt of the behavior of such a source is shown in Figure 17. Then the average rate in each interval was calculated by dividing the number of cells received by the length of the interval. This gave a traffic descriptor with rates in the range from 0 to 21.624 Mbit/s (52 cells/ms) with an average of 11.2568 Mbit/s (26.54905 cells/ms). By evaluating the occurrence of each rate an original, 53 state traffic descriptor with the mentioned rates and their associated probabilities was obtained.

Since a 53 state traffic descriptor has too many states to be processed efficiently by the CAC algorithm it was then reduced to fewer states. The following section shows the influence of the reduced number of states used to approximate the 53 original rates and the influence of the granularity (number of bins) with which the aggregate traffic was represented.

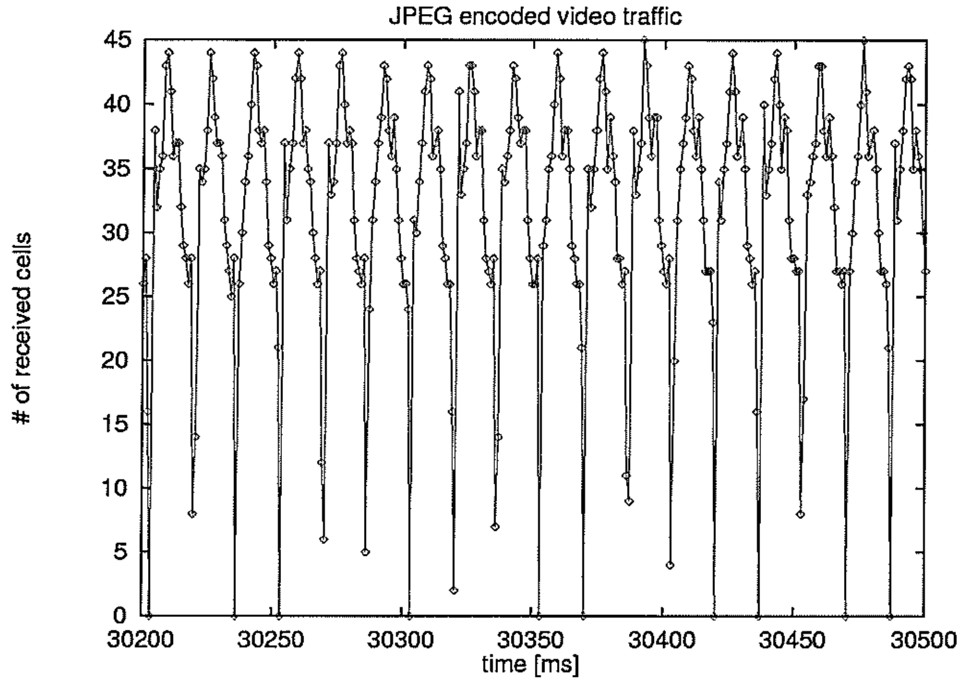


FIGURE 17. Number of ATM cells sent in 1ms intervals by a JPEG encoded video source

7.1.2 Results

The following two figures show how the number of bins used to represent the aggregate traffic descriptor and the number of states with which the original traffic source was approximated influence the CAC algorithm. For all simulations the number of aggregate traffic bins indicated is the number of bins used to represent the rates between zero and the link rate. In both figures a 100% average link load is achieved if 13.3 connections are multiplexed on the link.

Figure 18 shows the influence of the number of states chosen to reduce a many state traffic descriptor to a reduced state model. The more states are used for the approximation, the closer the approximated virtual cell loss rate is to the original cell loss rate, obtained with the original traffic descriptor and without quantization (by just using enough aggregate traffic bins). When the three approximations are compared with the classical model of an on off approximation it can be seen that the suggested approach gives a much better result.

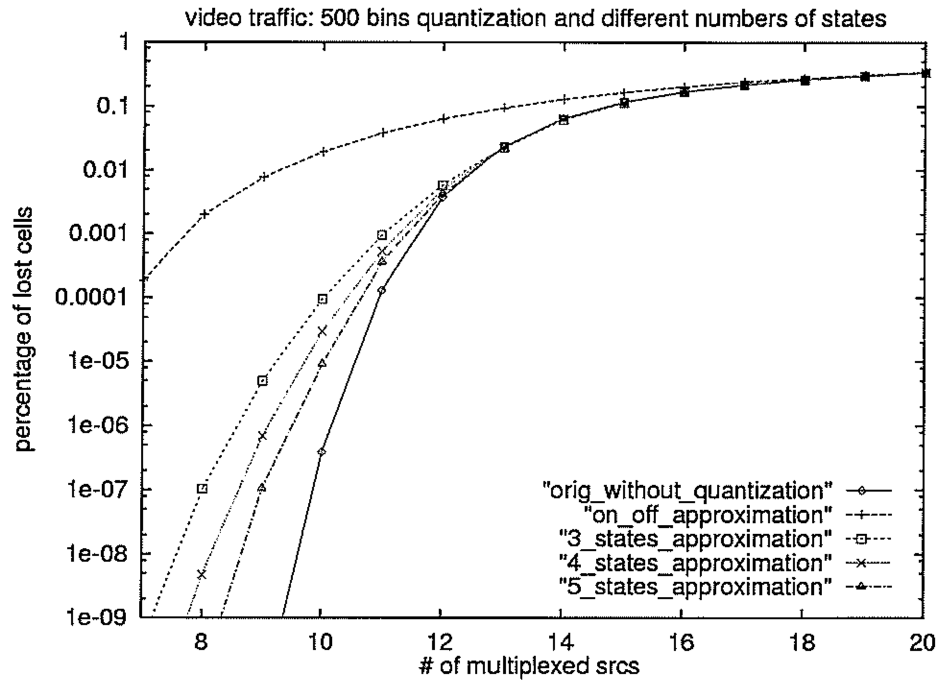


FIGURE 18. Cell loss rates for video traffic multiplexing. Link rate=150Mb/s, the approximations use 500 quantization bins and varying numbers of states

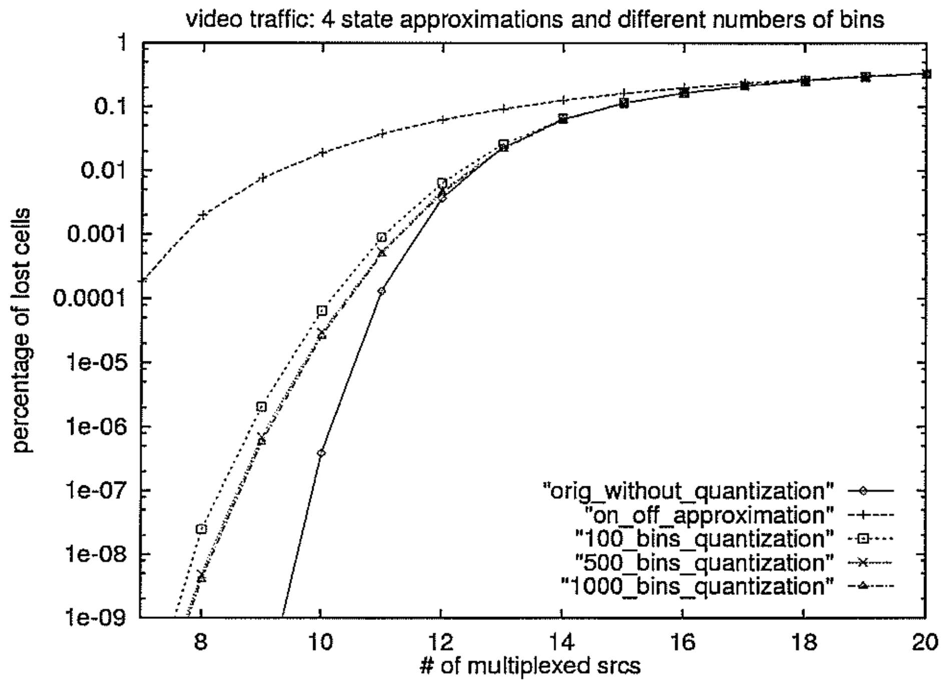


FIGURE 19. Cell loss rates for video traffic multiplexing. Link rate=150Mb/s, the approximations use 4 states and varying numbers of quantization bins

Figure 19 depicts how the number of aggregate traffic bins influences the accuracy of the CAC algorithm. Clearly, 100 bins give a worse performance than 500 and 1000. The difference between 500 and 1000 is not so big, though. Anyway, no matter how many bins are used the suggested approach has still a better performance than the on off approximation of a classic model

7.2 World Wide Web Traffic

The second traffic to be considered is the traffic expected from an 'average' world wide web user being connected to the www-server at Washington University.

7.2.1 Determining the Traffic Parameters

Although browsing the world wide web is an interactive process, the traffic generated is very asymmetric and consists mainly in the direction from the server to the client. As an ATM connection is normally set-up by the client application it is very hard to estimate the traffic, the client will receive from the server as the client has no idea which files it is going to access, what the size of the files is, and how fast the server is able to send the files (might depend on the current load of the server).

The suggested approach to this problem consists in having the server determine the traffic parameters. Since the server has even less knowledge about the client's intentions, and since the traffic contract should not be changed for every file that the server sends, the traffic descriptor is based on a statistical model the server uses.

First of all, what does a www client expect from a server: Since behind the www client there is a human user, the response time to a request is critical. However, a person can not really notice a difference between a response time of 0.1 millisecond or 0.5 millisecond. Asking for a response time that is below the human perception does not make sense in this case. From the perspective of the user, though, it is important the response time is constant, independent of the size of the file he asked for (a user does not really care how long a file is, he just wants to receive it and have it displayed with constant performance). Therefore the transmission rate should be based on the file size as shown in the following equation.

$$TransmissionRate = \frac{FileSize}{SpecifiedResponseTime} \tag{32}$$

Of course, in general, consecutive files sent to the same client have different sizes. Therefore a statistic is generated containing the distribution of all the files and their length that the server provided in a sample interval of time (e.g. during a day, or a week). From the file length the required transmission rate can be calculated and from its distribution the probability that a certain transmission rate will occur. Figure 20 shows the distribution of file lengths sent by the www server in the computer science department of Washington University in the five days from 4/2/96 to 4/6/96. The average file length was 22.8 kilobytes and a total of 46497 files were transferred.

Since the only information available about the transmitted www files was the logging file generated by the www server some additional assumptions had to be made. In a typical www session normally several files are transmitted from the server to the user (a user might download the home page, then make more and more specific inquiries until he has found the file he was looking for). To come up with a www traffic descriptor therefore the interval of time between transmissions has to be taken in consideration. For this the following value was assumed: The average time between transmissions to a www-client can be calculated from the logging file and is 33 seconds if the maximum idle time in a session is not allowed to exceed 10 minutes (if it does, the connection would be dropped due to a time-out in the server or the client in a real ATM connection).

The statistics thus obtained is of course only true for the type of networks and data that are currently available to www clients. Since ATM networks are considered for the CAC algorithm all file lengths were multiplied by a factor of ten. The reason for this is, that as soon as more bandwidth is available it will be used up very quickly by having larger files to transmit (e.g. a small video film, rather than a single picture). Making the assumption that every file has to be transmitted in half a second a www traffic profile was generated with 101 states and rates between 0 and 10 Megabit/s. Since the average is only 60 Kilobit/s this type of traffic is extremely bursty.

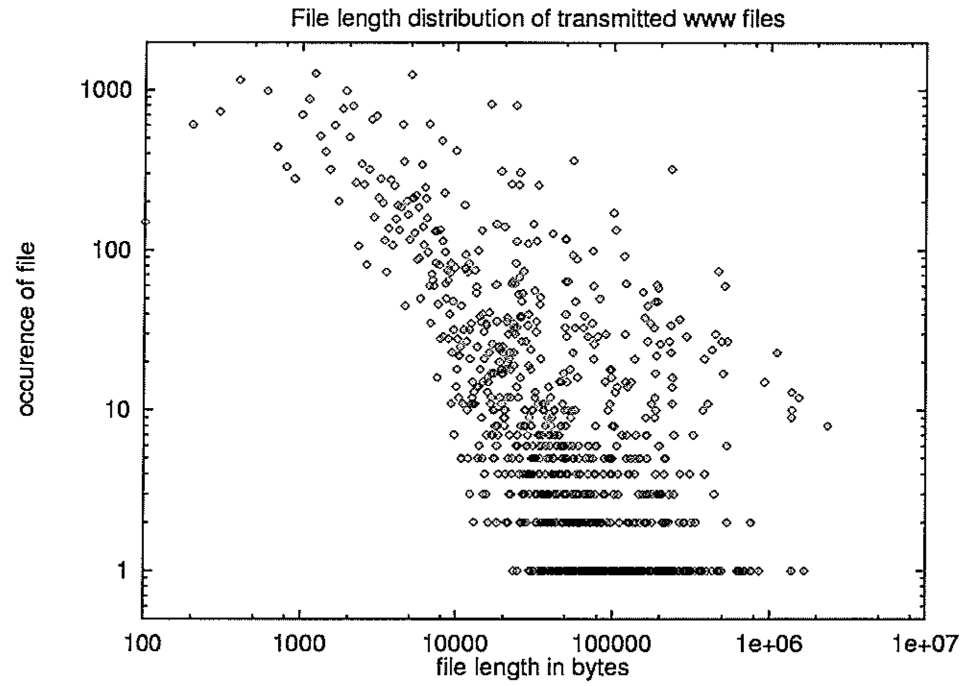


FIGURE 20. Distribution of file sizes transmitted by the www-server in the CS department of Washington University

7.2.2 Results

The following two figures show how the number of bins used to represent the aggregate traffic descriptor and the number of states with which the original traffic source was approximated influence the CAC algorithm. For all simulations the number of aggregate traffic bins indicated is the number of bins used to represent the rates between zero and the link rate. In both figures a 100% average link utilization is achieved if there are 2500 connections multiplexed on the link.

Figure 21 shows the influence of the number of states chosen to reduce a many state traffic descriptor to a reduced state model. The difference from having 3 states to 4 is visible whereas 4 states or 5 states does not really make a big difference. Again the comparison with the classical model of an on off approximation shows that the suggested approach is much more precise.

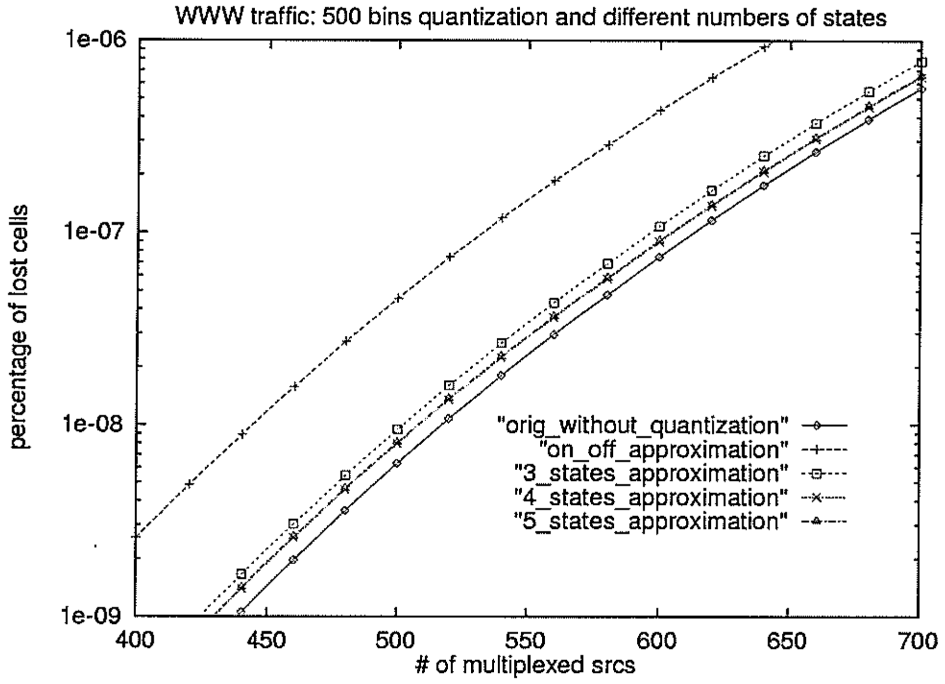


FIGURE 21. Cell loss rates for www traffic multiplexing. Link rate=150Mb/s, the approximations use 500 bins quantization and different numbers of states

Figure 22 depicts how the number of aggregate traffic bins influences the accuracy of the CAC algorithm. The more bins there are, the closer the approximation curve is to the curve obtained with the original descriptor. In this case the difference between 100, 500 and 1000 bins is obvious.

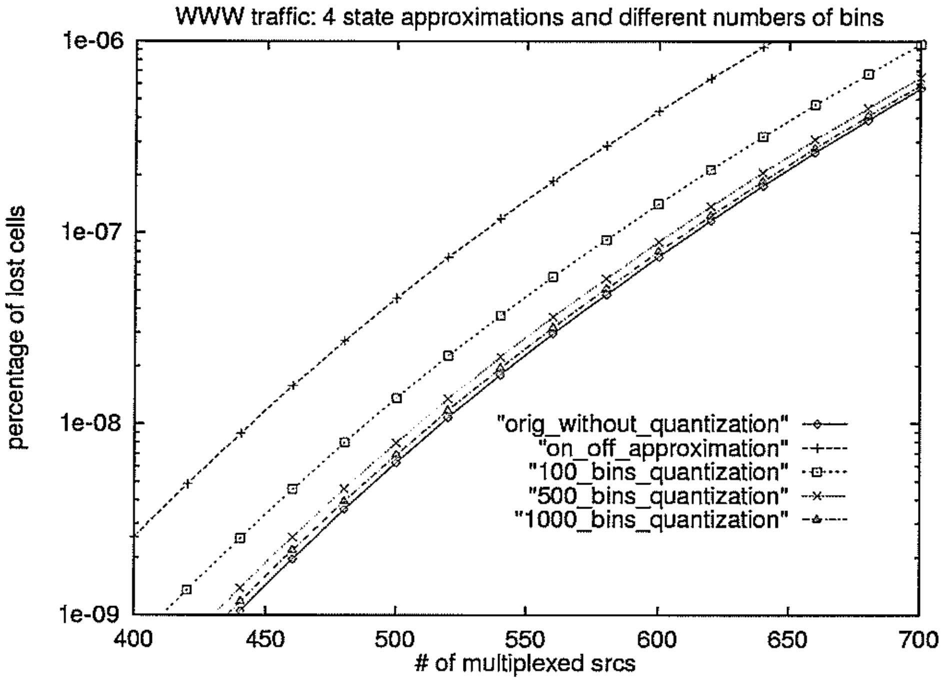


FIGURE 22. Cell loss rates for www traffic multiplexing. Link rate=150Mb/s, the approximations use 4 states and different numbers of quantization bins

7.3 Heterogenous Traffic Multiplexing

Of course the suggested CAC algorithm is not limited to homogeneous traffic multiplexing. In the example of Figure 23 video and www traffic are multiplexed together on a link. The relation between multiplexed video and www sources is 1:19 which explains that there is a big step in the percentage of lost cells every 20 connections (when a new video source is added) whereas the loss rate only increases slightly with every additional www source that is added. This is due to the fact that 19 www sources only generate an average load of 1.14 Mb/s whereas a single video source generates an average load of 11.3 Mb/s. On the other hand, the www sources are much burstier, so adding a new www source generates a noticeable increase in the cell loss probability. To make the differences more obvious, the number of approximated states and used bins was changed simultaneously from plot to plot. From the less precise 3 states and 100 bins to the more precise 5 states and 1000 bin approximation approximated cell loss clearly gets closer and closer to the cell loss of the original descriptor with the increase in precision. Again, even with the lowest degree of precision shown the performance is much better than the classical one of an on off approximation.

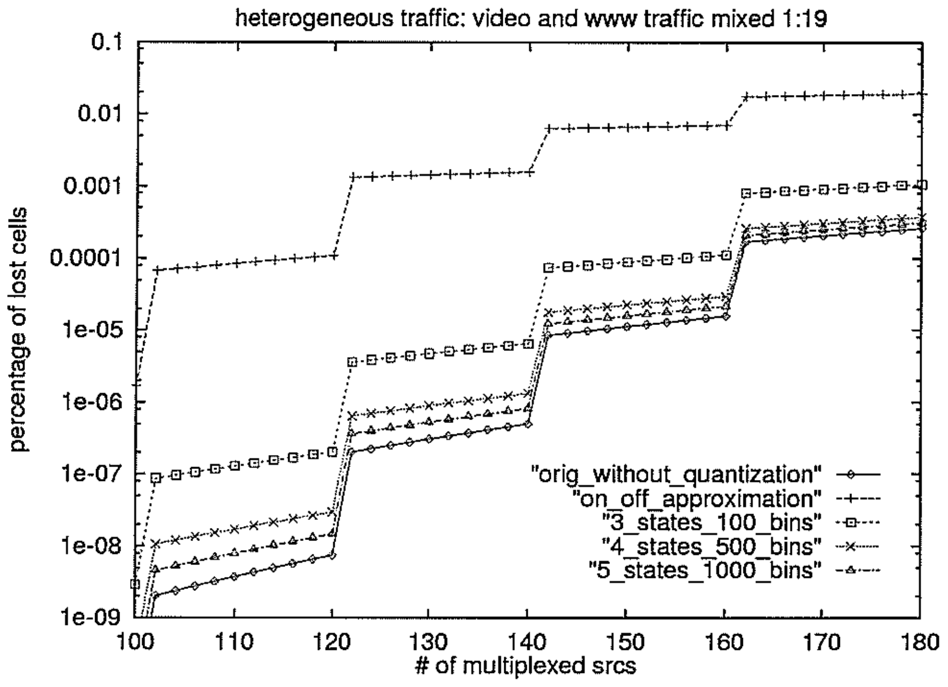
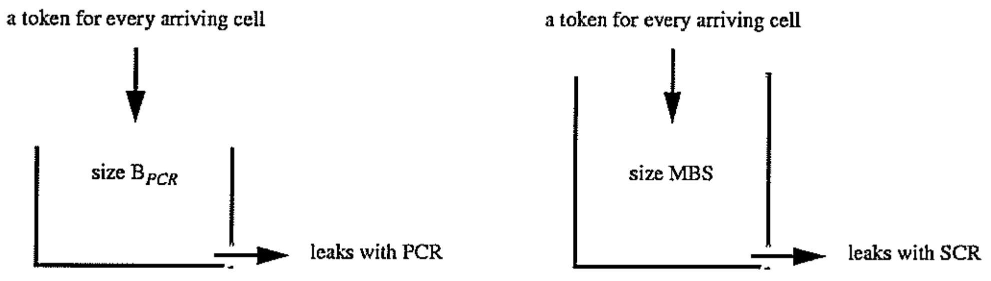


FIGURE 23. Cell loss rates for mixed traffic. Video and www traffic connections are mixed 1:19. Both traffic types use 3,4,5 in case of the approximation.

8 Usage Parameter Control (UPC)

Especially in a WAN environment carriers want to make sure that every connection sends according to its traffic contract. The term generally used is usage parameter control (UPC) at the user-network boundary or network parameter control (NPC) between two networks [16]. In this section only UPC is considered.

The general scheme used for UPC is the leaky bucket algorithm. For the classical case where a traffic descriptor is given by MBS, PCR, and SCR, two leaky buckets are used as shown in Figure 24. The leaky bucket to the left ensures that the source cannot send at a rate greater than the peak cell rate. The leaky bucket to the right ensures that the sustainable cell rate (the average) is not exceeded in a more 'long-term' perspective. Typically the left bucket has a size of only a few token whereas the right bucket's size corresponds to the Maximum Burst Size.

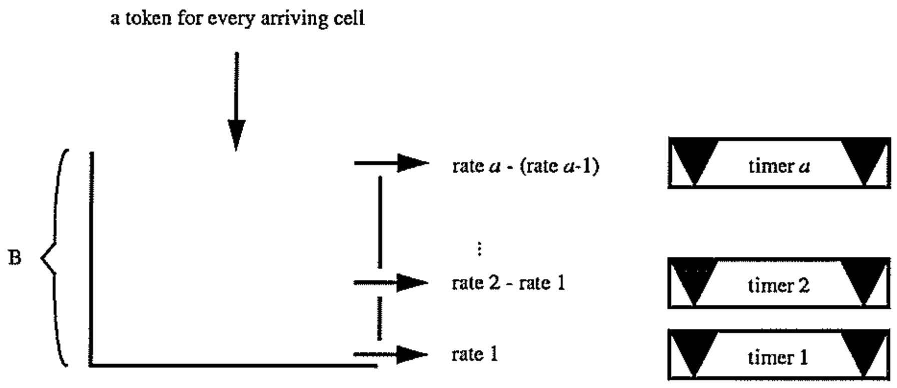

a token for every arriving cell
size B_{PCR}
leaks with PCR
a token for every arriving cell
size MBS
leaks with SCR

Algorithm: For every arriving cell put a token in each container. If either overflows cell is not conform.

FIGURE 24. Leaky bucket scheme

Figure 24 shows how the leaky bucket scheme works. For each arriving cell a token is inserted in each of the buckets which 'leak' tokens with the specified rates. A cell is only conforming if neither bucket is full when the token is inserted.

The leaky bucket scheme is relatively easy to implement. To adapt it to multistate traffic descriptors it is changed as shown in Figure 25.



Algorithm: For every arriving cell put a token in bucket. The highest timer below current level runs. If a timer is expired, the next higher, not expired timer is allowed to run. If no timer is able to run the cell is not conform.

FIGURE 25. Modified leaky bucket scheme usable for multistate traffic

The modified leaky bucket scheme uses one bucket for every multistate traffic descriptor. The bucket has a leak for every state/rate of the traffic descriptor. Since the leaks are on different levels leaking is only possible if the current level of tokens inside the buffer is above the leak. The leaks have different sizes so that the rate at which cells leave the buffer are different. For leak *i* (counted from the bottom) the leaking rate is:

$$leakRate(i) = \lambda_i - \lambda_{i-1} \qquad \text{where } leakRate(1) = \lambda_1 \tag{33}$$

This scheme alone does not yet guarantee that a traffic source has to send according to its traffic contract. To make sure the statistical behavior of the traffic source is correct, a timer is associated with each of the rates. Only one timer (the timer associated with the currently highest leaking leak) can run. Each timer is initially set with a value that is proportional to the traffic source's probability it represents. The initial timer value of timer *i* is defined as

$$InitialTimerValue(i) = \lceil Bp_i \rceil \tag{34}$$

Therefore, the sum of all initial timer values is equal to B (except of some rounding errors). Assigning initial values to the timers guarantees that over the time period B every rate has been active at most Bp_i and therefore the statistical behavior was the same as predicted in the traffic contract. In order to guarantee that the maximum traffic rate is not exceeded a second buffer similar to the left bucket in Figure 24 is still needed.

The choice of B determines the interval over the traffic descriptor is supposed to be statistically correct. Of course the timers need to be reset periodically otherwise they expire and no cell would be considered to be compliant anymore. To avoid boundary effects, the timers are reset before the end of the interval B by adding the initial value to their current value.

Another way to look at this UPC algorithm without the leaky bucket scheme is to determine for every cell at which rate it was sent. This rate can be determined by measuring the time intervals between the cells. The measured rate is then associated with the (next higher) rate of the traffic contract. Now again, for a time interval B, every rate of the traffic contract is allowed to send a predetermined number of cells that corresponds to the probability of the rate.

9 Conclusions

The cell loss rate is one of the most important parameters to describe the QoS of an ATM connection. For the suggested CAC algorithm the virtual cell loss probability is used which gives an upper bound on the real cell loss rate of a connection.

Current approaches that describe a traffic source in terms of PCR, SCR and MBS are exact for sources that have an on-off behavior. However, a lot of sources have a more complex behavior that can not be appropriately represented by these parameters. Therefore, a multistate traffic source model was suggested that specifies traffic sources statistically in terms of rates and associated probabilities to send at that rate. In addition, this traffic model does not make any assumption about the type of traffic or the arriving model of cells which gives a lot of flexibility because no traffic classes have to be updated or added if a new type of traffic should be defined.

It was shown how to use an incremental model to keep track of the aggregate bandwidth and how to update the values when new connections are added or existing connections are dropped. Using this model to represent the aggregate traffic, calculating the virtual cell loss probability takes constant time. The calculation time depends only on the number of bins with which the aggregate traffic is represented and on the number of states that are allowed to describe individual traffic sources. Since traffic sources can potentially have a large number of states an algorithm was presented that approximates them with a reduced state model that is guaranteed to be conservative while still being sufficiently accurate. The same method can be used to quantize the traffic descriptor to fit in the quantized rate used by the CAC algorithm to represent the aggregate traffic.

The incremental model used to represent the traffic descriptor is susceptible to calculation imprecisions. By using the cumulative probability upper and lower bounds can be calculated for the updated aggregate traffic descriptor which prevents the accumulation of errors. By further restricting certain parameters it was shown how to overcome the calculation problems.

Since traffic policing is an important issue especially in WANs a modified leaky bucket scheme was presented which can be used for the usage parameter control of multistate traffic sources.

The suggested CAC algorithm is tunable. Every network manager can determine the relation he wants between accuracy of the algorithm and processing power. This allows one to use a low cost processor for a small switch where the fraction of bandwidth lost is not so important or to be more accurate on a Gigabit switch where one percent of lost bandwidth is still several Megabits/s. Even with a CAC configuration that calculates very precisely the number of connections that can safely be multiplexed (by increasing the number of states with which connections are specified and the number of bins with which the aggregate traffic is represented) a calculation time below 1 millisecond is achieved. For a more typical configuration with 4 states and 100 bins the worst-case processing time is below 0.1 milliseconds.

The CAC algorithm was developed in C++ and integrated in the control software [18],[19] used for the Gigabit switch network of Washington University.

10 Acknowledgments

At this place I would like to thank everybody for their support of this work which stretched over a time period of almost a year.

I would like to thank my wife Yvonne, who gave birth to our wonderful daughter Tanja, and who volunteered to change diapers when this report was to be finished.

I also thank Dr. Turner, my advisor. Without his ideas and the fruitful discussions this work could not have been done.

Furthermore I would like to thank the ARL, especially John DeHart and Dakang Wu who explained to me the mysteries of CMAP and CMNP, and Jae Hwoon Lee who worked with me on this project at the beginning. Thanks also go to Ron Walkup who provided the JPEG encoded video data.

11 References

[1] Jonathan S. Turner, "Managing Bandwidth in ATM Networks with Bursty Traffic", IEEE Network, September 1992

[2] Hiroshi Saito, Kohei Shiomoto "Call Admission Control in an ATM Network Using Upper Bound of Cell Loss Probability", IEEE Transactions on Communications, Vol. 40, No. 9, September 1992.

[3] Totomu Murase et al., "A Call Admission Control Scheme for ATM Networks Using a Simple Quality Estimate", IEEE Journal on Selected Areas In Communications, Vol. 9 No 9, December 1991.

[4] Tsern-Huei Lee et alt., "Real Time Call Admission Control for ATM Networks with Heterogeneous Bursty Traffic", IEEE Supercomm/ICC Proceedings, 1994.

[5] Tao Yang, Danny H. K. Tsang, "A Novel Approach to EStimating the Cell Loss Probability in an ATM Multiplexer Loaded with Homogeneous On-Off Sources", IEEE Transactions on Communications, Vol. 43, No. 1, January 1995.

[6] Jimmy H. S. Chan, Danny H. K. Tsang, "Bandwidth Allocation of Multiple QOS Classes in ATM Environment", IEEE Infocom Proceedings, 1994.

[7] Zbigniew Dziong, Boris Shukhman, Lorne G. Mason, "Estimation of Aggregate Effective Bandwidth for Traffic Admission in ATM Networks", IEEE Infocom Proceedings, 1995.

[8] Roch Guerin, Hamid Ahmadi, Mahmoud Naghshineh, "Equivalent Capacity and Its Application to Bandwidth Allocation in High-speed Networks", IEEE Journal on Selected Areas in Communications, Vol. 9, No 7, September 1991.

[9] Sugih Jamin, Peter B. Danzig, Scott Shenker, Lixia Zhang, "A Measurement-based Admission Control Algorithm for Integrated Services Packet Networks", IEEE/ACM Transactions on Networking, 1996.

[10] Hiroshi Saito, Kohei Shiomoto, "Dynamic Call Admission Control in ATM Networks", IEEE Journal on Selected Areas in Communications, Vol9, No. 7, September 1991.

[11] Ray-Guang Cheng, Chung-Ju Chang, "A Neural-Net Based Fuzzy Admission Controller for an ATM Network", IEEE Infocom Proceedings, 1996.

[12] Richard G. Ogier, Nina T. Plotkin, Irfan Khan, "Neural Network Methods with Traffic Descriptor Compression for Call Admission Control", IEEE Infocom Proceedings, 1996.

[13] Nilesh R. Gohel, "Medical Video Transmission Via Bandwidth Constrained ATM Networks", Master's Thesis, Computer Science Department Washington University in St. Louis, 1994.

[14] Wassily Hoeffding, "Probability Inequalities for Sums of Bounded Random Variables", American Statistical Association Journal, March 1963.

[15] P. Tchebichef, "Sur les valeurs limites des integrales", Journal de Mathematiques Pures et Appliquees, Ser 2, 19 (1874).

[16] The ATM Forum, "Traffic Management Specification Version 4.0", af-95-0013R13, Letter Ballot, April 1996.

[17] International Telecommunication Union ITU-T, "Traffic Control and Congestion Control in B-ISDN", ITU_T Recommendation I.371, March 1993.

[18] Ken Cox, John DeHart, “CMAP”, WUCS-94-21, Department of Computer Science, Washington University in St. Louis, MO, July 1994.

[19] John DeHart and Dakang Wu, “Connection Management Network Protocol (CMNP)”, ARL-94-19, Department of Computer Science, Washington University in St. Louis, MO, September 1994.

12 Annex

The following sections show in a more detailed manner how the choice of different parameters influences the precision of the CAC algorithm.

For all tests the outputs look similar and describe both what input parameters were used and how this choice of parameters affected the precision of the calculation. Since the imprecision in calculation only gets noticeable after a large number of connections are added and dropped the imprecision was measured after adding and dropping different numbers of connections.

Two metrics are used to measure the imprecisions:

For an initial set of multiplexed connections the largest CBR connection (CBR connection with the maximum rate) is determined that can be multiplexed on the link. After adding and dropping connections, for the same number of initial connections the same test is done again. The difference between the now determined rate and the original rate is due to imprecisions in calculations. By dividing the difference by the link rate a relative error can be determined that can be compared among tests with different parameters.

The second metric is similar. Instead of determining the maximum CBR rate allowed on the link, the maximum number of additional connections of the same type are determined. Again, after adding and dropping connections, for the same number of initial connections the test is repeated and the number of free connections is given as output. To be able to make comparisons among different tests, a relative error is defined by dividing the difference through the total number of connections that could have been multiplexed.

The following list explains the abbreviations used in the test outputs:

- *linkRate*: transmission rate of the link.
- *Rbin*: number of bins between 0 and the link rate.
- *addBins*: number of bins above the link rate.
- *cellLossProb*: virtual cell loss probability.
- *overBookingRatio*: maximum allowed sum of all maximum rates of every descriptor divided by link rate.
- *packSize*: after adding at first the initial number of connections (*initConns*) connections are added and dropped in 'packs' of *packSize*.
- *printAfter*: defines when an output had to be printed.
- *maxRuns*: total number of times a 'pack' of connections was added and dropped.
- *initConns*: initial number of connections multiplexed on the link. For this number of connections the two metrics were determined.
- *TraffDesc*: this line shows the *connId* (not relevant), the *TrafficType* (0=CBR, 1=VBR) and the number of states of the traffic descriptor.
- *rates/probs*: For each rate of the traffic descriptor the associated probability is indicated below the rate.
- *initial free CBR bandwidth*: the maximum rate of a CBR descriptor that could be added at the beginning with the *initConns* connections multiplexed on the link.
- *initial free nb of conns*: the maximum number of additional connections that could be added at the beginning with the *initConns* connections multiplexed on the link.
- *free CBR*: The maximum CBR rate that can be accepted with the current calculation imprecision and with *initConns* connections already on the link.
- *lostCBR/linkRate*: the difference between (the original) *initial free CBR bandwidth* and (the current) *free CBR*, divided by the link rate.

- *free conns*: the maximum number of additional connections that can be accepted with the current calculation imprecision and with *initConns* connections already on the link.
- *lostConns/possibleConns*: the difference between (the original) *initial free nb of conns* and (the current) *free conns* divided by the total possible number of connections (at the beginning).

12.1 100 Bins For Aggregate Traffic

A first test measures the influence of the numbers of bins used to represent the aggregate traffic descriptor. While more bins allow a more precise description of the bandwidth, more bins also cause more calculation imprecisions.

This test also shows it is not necessarily the total number of connections dropped that determines the size of the calculation errors but more what the range of connections is where the dropping takes place.
If all connections were dropped every time then no calculation imprecisions would appear. This is due to the fact that all S[i] are equal to one when the last connection is dropped.
On the other hand, if almost the maximum of possible connection is multiplexed on the link and only 1 connection is dropped and added several thousand times, the calculation error is small, too.

The worst case happens if the number of connections on the link is always in the region of half the connections possible.

12.1.1 Four State Descriptors

```
linkRate: 100  Rbin: 100  addBins: 100  cellLossProb: 1e-05  overBookingRa-
tio: 1000
packSize: 60  printAfter:150  maxRuns:600  initConns:20
TraffDesc: connId=0  TrafficType= 1nb of states=4
rates:        0        1        2        3
probs:        0.6      0.2      0.1      0.1

initial free CBR bandwidth 69
initial free nb of Conns 71

'usable' after dropping 150 packs with 60 connections:
free CBR: 75          lostCBR/linkRate: -0.06
free conns: 64        lostConns/possibleConns: 0.0769231

'usable' after dropping 300 packs with 60 connections:
free CBR: 73          lostCBR/linkRate: -0.04
free conns: 64        lostConns/possibleConns: 0.0769231

'usable' after dropping 450 packs with 60 connections:
free CBR: 72          lostCBR/linkRate: -0.03
free conns: 64        lostConns/possibleConns: 0.0769231

'usable' after dropping 600 packs with 60 connections:
free CBR: 72          lostCBR/linkRate: -0.03
free conns: 63        lostConns/possibleConns: 0.0879121

==================================================================

linkRate: 100  Rbin: 100  addBins: 100  cellLossProb: 1e-05  overBookingRa-
tio: 1000
packSize: 40  printAfter:150  maxRuns:600  initConns:40
TraffDesc: connId=0  TrafficType= 1nb of states=4
rates:        0        1        2        3
probs:        0.6      0.2      0.1      0.1

initial free CBR bandwidth 48
initial free nb of Conns 51

'usable' after dropping 150 packs with 40 connections:
free CBR: 54          lostCBR/linkRate: -0.06
free conns: 45        lostConns/possibleConns: 0.0659341

'usable' after dropping 300 packs with 40 connections:
free CBR: 53          lostCBR/linkRate: -0.05
free conns: 45        lostConns/possibleConns: 0.0659341

'usable' after dropping 450 packs with 40 connections:
free CBR: 53          lostCBR/linkRate: -0.05
free conns: 45        lostConns/possibleConns: 0.0659341

'usable' after dropping 600 packs with 40 connections:
free CBR: 54          lostCBR/linkRate: -0.06
free conns: 46        lostConns/possibleConns: 0.0549451

==================================================================

linkRate: 100  Rbin: 100  addBins: 100  cellLossProb: 1e-05  overBookingRa-
tio: 1000
packSize: 20  printAfter:150  maxRuns:600  initConns:60
TraffDesc: connId=0  TrafficType= 1nb of states=4
rates:        0        1        2        3
probs:        0.6      0.2      0.1      0.1

initial free CBR bandwidth 29
initial free nb of Conns 31

'usable' after dropping 150 packs with 20 connections:
free CBR: 29          lostCBR/linkRate: 0
free conns: 31        lostConns/possibleConns: 0

'usable' after dropping 300 packs with 20 connections:
free CBR: 29          lostCBR/linkRate: 0
free conns: 31        lostConns/possibleConns: 0

'usable' after dropping 450 packs with 20 connections:
free CBR: 29          lostCBR/linkRate: 0
free conns: 31        lostConns/possibleConns: 0

'usable' after dropping 600 packs with 20 connections:
free CBR: 29          lostCBR/linkRate: 0
free conns: 31        lostConns/possibleConns: 0

==================================================================

linkRate: 100  Rbin: 100  addBins: 100  cellLossProb: 1e-05  overBookingRa-
tio: 1000
packSize: 50  printAfter:150  maxRuns:600  initConns:20
TraffDesc: connId=0  TrafficType= 1nb of states=4
rates:        0        1        2        3
probs:        0.6      0.14     0.13     0.13

initial free CBR bandwidth 66
initial free nb of Conns 60

'usable' after dropping 150 packs with 50 connections:
free CBR: 62          lostCBR/linkRate: 0.04
free conns: 53        lostConns/possibleConns: 0.0875

'usable' after dropping 300 packs with 50 connections:
free CBR: 67          lostCBR/linkRate: -0.01
free conns: 53        lostConns/possibleConns: 0.0875

'usable' after dropping 450 packs with 50 connections:
free CBR: 67          lostCBR/linkRate: -0.01
free conns: 53        lostConns/possibleConns: 0.0875

'usable' after dropping 600 packs with 50 connections:
free CBR: 67          lostCBR/linkRate: -0.01
free conns: 53        lostConns/possibleConns: 0.0875

==================================================================

linkRate: 100  Rbin: 100  addBins: 100  cellLossProb: 1e-05  overBookingRa-
tio: 1000
packSize: 30  printAfter:150  maxRuns:600  initConns:40
TraffDesc: connId=0  TrafficType= 1nb of states=4
rates:        0        1        2        3
probs:        0.6      0.14     0.13     0.13

initial free CBR bandwidth 43
initial free nb of Conns 40

'usable' after dropping 150 packs with 30 connections:
free CBR: 43          lostCBR/linkRate: 0
free conns: 40        lostConns/possibleConns: 0

'usable' after dropping 300 packs with 30 connections:
free CBR: 43          lostCBR/linkRate: 0
free conns: 40        lostConns/possibleConns: 0

'usable' after dropping 450 packs with 30 connections:
free CBR: 43          lostCBR/linkRate: 0
free conns: 40        lostConns/possibleConns: 0

'usable' after dropping 600 packs with 30 connections:
free CBR: 43          lostCBR/linkRate: 0
free conns: 40        lostConns/possibleConns: 0
```

```
==================================================

linkRate: 100 Rbin: 100 addBins: 100 cellLossProb: 1e-05 overBookingRa-
tio: 1000
packSize: 10 printAfter:150 maxRuns:600 initConns:60
TraffDesc: connId=0 TrafficType= 1nb of states=4
rates:      0       1       2       3
probs:      0.6     0.14    0.13    0.13

initial free CBR bandwidth 21
initial free nb of Conns 20

'usable' after dropping 150 packs with 10 connections:
free CBR: 21          lostCBR/linkRate: 0
free conns: 20        lostConns/possibleConns: 0

'usable' after dropping 300 packs with 10 connections:
free CBR: 21          lostCBR/linkRate: 0
free conns: 20        lostConns/possibleConns: 0

'usable' after dropping 450 packs with 10 connections:
free CBR: 21          lostCBR/linkRate: 0
free conns: 20        lostConns/possibleConns: 0

'usable' after dropping 600 packs with 10 connections:
free CBR: 21          lostCBR/linkRate: 0
free conns: 20        lostConns/possibleConns: 0

==================================================

linkRate: 100 Rbin: 100 addBins: 100 cellLossProb: 1e-05 overBookingRa-
tio: 1000
packSize: 70 printAfter:150 maxRuns:600 initConns:10
TraffDesc: connId=0 TrafficType= 1nb of states=4
rates:      0       1       3       4
probs:      0.6     0.38    0.01    0.01

initial free CBR bandwidth 86
initial free nb of Conns 143

'usable' after dropping 150 packs with 70 connections:
free CBR: 88          lostCBR/linkRate: -0.02
free conns: 130       lostConns/possibleConns: 0.0849673

'usable' after dropping 300 packs with 70 connections:
free CBR: 86          lostCBR/linkRate: 0
free conns: 130       lostConns/possibleConns: 0.0849673

'usable' after dropping 450 packs with 70 connections:
free CBR: 86          lostCBR/linkRate: 0
free conns: 130       lostConns/possibleConns: 0.0849673

'usable' after dropping 600 packs with 70 connections:
free CBR: 86          lostCBR/linkRate: 0
free conns: 130       lostConns/possibleConns: 0.0849673

==================================================

linkRate: 100 Rbin: 100 addBins: 100 cellLossProb: 1e-05 overBookingRa-
tio: 1000
packSize: 50 printAfter:150 maxRuns:600 initConns:30
TraffDesc: connId=0 TrafficType= 1nb of states=4
rates:      0       1       3       4
probs:      0.6     0.38    0.01    0.01

initial free CBR bandwidth 72
initial free nb of Conns 123

'usable' after dropping 150 packs with 50 connections:
free CBR: 83          lostCBR/linkRate: -0.11
free conns: 115       lostConns/possibleConns: 0.0522876

'usable' after dropping 300 packs with 50 connections:
free CBR: 82          lostCBR/linkRate: -0.1
free conns: 114       lostConns/possibleConns: 0.0588235

'usable' after dropping 450 packs with 50 connections:
free CBR: 80          lostCBR/linkRate: -0.08
free conns: 113       lostConns/possibleConns: 0.0653595

'usable' after dropping 600 packs with 50 connections:
free CBR: 80          lostCBR/linkRate: -0.08
free conns: 114       lostConns/possibleConns: 0.0588235

==================================================

linkRate: 100 Rbin: 100 addBins: 100 cellLossProb: 1e-05 overBookingRa-
tio: 1000
packSize: 30 printAfter:150 maxRuns:600 initConns:50
TraffDesc: connId=0 TrafficType= 1nb of states=4
rates:      0       1       3       4
probs:      0.6     0.38    0.01    0.01

initial free CBR bandwidth 59
initial free nb of Conns 103

'usable' after dropping 150 packs with 30 connections:
free CBR: 75          lostCBR/linkRate: -0.16
free conns: 101       lostConns/possibleConns: 0.0130719

'usable' after dropping 300 packs with 30 connections:
free CBR: 77          lostCBR/linkRate: -0.18
free conns: 102       lostConns/possibleConns: 0.00653595

'usable' after dropping 450 packs with 30 connections:
free CBR: 77          lostCBR/linkRate: -0.18
free conns: 102       lostConns/possibleConns: 0.00653595

'usable' after dropping 600 packs with 30 connections:
free CBR: 77          lostCBR/linkRate: -0.18
free conns: 102       lostConns/possibleConns: 0.00653595

==================================================

linkRate: 100 Rbin: 100 addBins: 100 cellLossProb: 1e-05 overBookingRa-
tio: 1000
packSize: 60 printAfter:150 maxRuns:600 initConns:10
TraffDesc: connId=0 TrafficType= 1nb of states=4
rates:      0       2       3       4
probs:      0.6     0.38    0.01    0.01

initial free CBR bandwidth 80
initial free nb of Conns 70

'usable' after dropping 150 packs with 60 connections:
free CBR: 80          lostCBR/linkRate: 0
free conns: 70        lostConns/possibleConns: 0

'usable' after dropping 300 packs with 60 connections:
free CBR: 80          lostCBR/linkRate: 0
free conns: 70        lostConns/possibleConns: 0

'usable' after dropping 450 packs with 60 connections:
free CBR: 76          lostCBR/linkRate: 0.04
free conns: 63        lostConns/possibleConns: 0.0875

'usable' after dropping 600 packs with 60 connections:
free CBR: 77          lostCBR/linkRate: 0.03
free conns: 63        lostConns/possibleConns: 0.0875

==================================================

linkRate: 100 Rbin: 100 addBins: 100 cellLossProb: 1e-05 overBookingRa-
tio: 1000
packSize: 40 printAfter:150 maxRuns:600 initConns:30
TraffDesc: connId=0 TrafficType= 1nb of states=4
rates:      0       2       3       4
probs:      0.6     0.38    0.01    0.01

initial free CBR bandwidth 55
initial free nb of Conns 50

'usable' after dropping 150 packs with 40 connections:
free CBR: 60          lostCBR/linkRate: -0.05
free conns: 44        lostConns/possibleConns: 0.075

'usable' after dropping 300 packs with 40 connections:
free CBR: 57          lostCBR/linkRate: -0.02
free conns: 44        lostConns/possibleConns: 0.075

'usable' after dropping 450 packs with 40 connections:
free CBR: 57          lostCBR/linkRate: -0.02
free conns: 44        lostConns/possibleConns: 0.075

'usable' after dropping 600 packs with 40 connections:
free CBR: 58          lostCBR/linkRate: -0.03
free conns: 44        lostConns/possibleConns: 0.075

==================================================

linkRate: 100 Rbin: 100 addBins: 100 cellLossProb: 1e-05 overBookingRa-
tio: 1000
packSize: 20 printAfter:150 maxRuns:600 initConns:50
TraffDesc: connId=0 TrafficType= 1nb of states=4
rates:      0       2       3       4
probs:      0.6     0.38    0.01    0.01

initial free CBR bandwidth 32
initial free nb of Conns 30

'usable' after dropping 150 packs with 20 connections:
free CBR: 33          lostCBR/linkRate: -0.01
free conns: 29        lostConns/possibleConns: 0.0125

'usable' after dropping 300 packs with 20 connections:
free CBR: 33          lostCBR/linkRate: -0.01
free conns: 29        lostConns/possibleConns: 0.0125
```

```
'usable' after dropping 450 packs with 20 connections:
free CBR: 33          lostCBR/linkRate: -0.01
free conns: 29        lostConns/possibleConns: 0.0125

'usable' after dropping 600 packs with 20 connections:
free CBR: 33          lostCBR/linkRate: -0.01
free conns: 29        lostConns/possibleConns: 0.0125

==============================================================
```

12.1.2 Three State Descriptors

```
linkRate: 100  Rbin: 100  addBins: 100  cellLossProb: 1e-05  overBookingRa-
tio: 1000
packSize: 40  printAfter:150  maxRuns:600  initConns:10
TraffDesc: connId=0  TrafficType= 1nb of states=3
rates:        0         1         3
probs:        0.6       0.1       0.3

initial free CBR bandwidth 75
initial free nb of Conns 51

'usable' after dropping 150 packs with 40 connections:
free CBR: 75          lostCBR/linkRate: 0
free conns: 51        lostConns/possibleConns: 0

'usable' after dropping 300 packs with 40 connections:
free CBR: 75          lostCBR/linkRate: 0
free conns: 51        lostConns/possibleConns: 0

'usable' after dropping 450 packs with 40 connections:
free CBR: 75          lostCBR/linkRate: 0
free conns: 51        lostConns/possibleConns: 0

'usable' after dropping 600 packs with 40 connections:
free CBR: 75          lostCBR/linkRate: 0
free conns: 51        lostConns/possibleConns: 0

==============================================================

linkRate: 100  Rbin: 100  addBins: 100  cellLossProb: 1e-05  overBookingRa-
tio: 1000
packSize: 20  printAfter:150  maxRuns:600  initConns:30
TraffDesc: connId=0  TrafficType= 1nb of states=3
rates:        0         1         3
probs:        0.6       0.1       0.3

initial free CBR bandwidth 44
initial free nb of Conns 31

'usable' after dropping 150 packs with 20 connections:
free CBR: 44          lostCBR/linkRate: 0
free conns: 30        lostConns/possibleConns: 0.0163934

'usable' after dropping 300 packs with 20 connections:
free CBR: 44          lostCBR/linkRate: 0
free conns: 31        lostConns/possibleConns: 0

'usable' after dropping 450 packs with 20 connections:
free CBR: 44          lostCBR/linkRate: 0
free conns: 31        lostConns/possibleConns: 0

'usable' after dropping 600 packs with 20 connections:
free CBR: 44          lostCBR/linkRate: 0
free conns: 30        lostConns/possibleConns: 0.0163934

==============================================================

linkRate: 100  Rbin: 100  addBins: 100  cellLossProb: 1e-05  overBookingRa-
tio: 1000
packSize: 10  printAfter:150  maxRuns:600  initConns:40
TraffDesc: connId=0  TrafficType= 1nb of states=3
rates:        0         1         3
probs:        0.6       0.1       0.3

initial free CBR bandwidth 29
initial free nb of Conns 21

'usable' after dropping 150 packs with 10 connections:
free CBR: 29          lostCBR/linkRate: 0
free conns: 21        lostConns/possibleConns: 0

'usable' after dropping 300 packs with 10 connections:
free CBR: 29          lostCBR/linkRate: 0
free conns: 21        lostConns/possibleConns: 0

'usable' after dropping 450 packs with 10 connections:
free CBR: 29          lostCBR/linkRate: 0
free conns: 21        lostConns/possibleConns: 0

'usable' after dropping 600 packs with 10 connections:
free CBR: 29          lostCBR/linkRate: 0
free conns: 21        lostConns/possibleConns: 0

==============================================================

linkRate: 100  Rbin: 100  addBins: 100  cellLossProb: 1e-05  overBookingRa-
tio: 1000
packSize: 45  printAfter:150  maxRuns:600  initConns:10
TraffDesc: connId=0  TrafficType= 1nb of states=3
rates:        0         2         3
probs:        0.6       0.2       0.2

initial free CBR bandwidth 77
initial free nb of Conns 53

'usable' after dropping 150 packs with 45 connections:
free CBR: 77          lostCBR/linkRate: 0
free conns: 53        lostConns/possibleConns: 0

'usable' after dropping 300 packs with 45 connections:
free CBR: 77          lostCBR/linkRate: 0
free conns: 53        lostConns/possibleConns: 0

'usable' after dropping 450 packs with 45 connections:
free CBR: 77          lostCBR/linkRate: 0
free conns: 53        lostConns/possibleConns: 0

'usable' after dropping 600 packs with 45 connections:
free CBR: 77          lostCBR/linkRate: 0
free conns: 53        lostConns/possibleConns: 0

==============================================================

linkRate: 100  Rbin: 100  addBins: 100  cellLossProb: 1e-05  overBookingRa-
tio: 1000
packSize: 25  printAfter:150  maxRuns:600  initConns:30
TraffDesc: connId=0  TrafficType= 1nb of states=3
rates:        0         2         3
probs:        0.6       0.2       0.2

initial free CBR bandwidth 46
initial free nb of Conns 33

'usable' after dropping 150 packs with 25 connections:
free CBR: 46          lostCBR/linkRate: 0
free conns: 33        lostConns/possibleConns: 0

'usable' after dropping 300 packs with 25 connections:
free CBR: 46          lostCBR/linkRate: 0
free conns: 33        lostConns/possibleConns: 0

'usable' after dropping 450 packs with 25 connections:
free CBR: 46          lostCBR/linkRate: 0
free conns: 33        lostConns/possibleConns: 0

'usable' after dropping 600 packs with 25 connections:
free CBR: 46          lostCBR/linkRate: 0
free conns: 33        lostConns/possibleConns: 0

==============================================================

linkRate: 100  Rbin: 100  addBins: 100  cellLossProb: 1e-05  overBookingRa-
tio: 1000
packSize: 5  printAfter:150  maxRuns:600  initConns:50
TraffDesc: connId=0  TrafficType= 1nb of states=3
rates:        0         2         3
probs:        0.6       0.2       0.2

initial free CBR bandwidth 18
initial free nb of Conns 13

'usable' after dropping 150 packs with 5 connections:
free CBR: 18          lostCBR/linkRate: 0
free conns: 13        lostConns/possibleConns: 0

'usable' after dropping 300 packs with 5 connections:
free CBR: 18          lostCBR/linkRate: 0
free conns: 13        lostConns/possibleConns: 0

'usable' after dropping 450 packs with 5 connections:
free CBR: 18          lostCBR/linkRate: 0
free conns: 13        lostConns/possibleConns: 0

'usable' after dropping 600 packs with 5 connections:
free CBR: 18          lostCBR/linkRate: 0
free conns: 13        lostConns/possibleConns: 0

==============================================================
```

12.1.3 Two State Descriptors

```
linkRate: 100  Rbin: 100  addBins: 100  cellLossProb: 1e-05  overBookingRa-
tio: 1000
packSize: 55  printAfter:150  maxRuns:600  initConns:10
TraffDesc: connId=0  TrafficType= 1nb of states=2
rates:        1        2
probs:        0.6      0.4

initial free CBR bandwidth 86
initial free nb of Conns 61

'usable' after dropping 150 packs with 55 connections:
free CBR: 86          lostCBR/linkRate: 0
free conns: 61        lostConns/possibleConns: 0

'usable' after dropping 300 packs with 55 connections:
free CBR: 86          lostCBR/linkRate: 0
free conns: 61        lostConns/possibleConns: 0

'usable' after dropping 450 packs with 55 connections:
free CBR: 86          lostCBR/linkRate: 0
free conns: 61        lostConns/possibleConns: 0

'usable' after dropping 600 packs with 55 connections:
free CBR: 86          lostCBR/linkRate: 0
free conns: 61        lostConns/possibleConns: 0

==================================================

linkRate: 100  Rbin: 100  addBins: 100  cellLossProb: 1e-05  overBookingRa-
tio: 1000
packSize: 35  printAfter:150  maxRuns:600  initConns:30
TraffDesc: connId=0  TrafficType= 1nb of states=2
rates:        1        2
probs:        0.6      0.4

initial free CBR bandwidth 58
initial free nb of Conns 41

'usable' after dropping 150 packs with 35 connections:
free CBR: 58          lostCBR/linkRate: 0
free conns: 41        lostConns/possibleConns: 0

'usable' after dropping 300 packs with 35 connections:
free CBR: 58          lostCBR/linkRate: 0
free conns: 41        lostConns/possibleConns: 0

'usable' after dropping 450 packs with 35 connections:
free CBR: 58          lostCBR/linkRate: 0
free conns: 41        lostConns/possibleConns: 0

'usable' after dropping 600 packs with 35 connections:
free CBR: 58          lostCBR/linkRate: 0
free conns: 41        lostConns/possibleConns: 0

==================================================

linkRate: 100  Rbin: 100  addBins: 100  cellLossProb: 1e-05  overBookingRa-
tio: 1000
packSize: 15  printAfter:150  maxRuns:600  initConns:50
TraffDesc: connId=0  TrafficType= 1nb of states=2
rates:        1        2
probs:        0.6      0.4

initial free CBR bandwidth 30
initial free nb of Conns 21

'usable' after dropping 150 packs with 15 connections:
free CBR: 30          lostCBR/linkRate: 0
free conns: 21        lostConns/possibleConns: 0

'usable' after dropping 300 packs with 15 connections:
free CBR: 30          lostCBR/linkRate: 0
free conns: 21        lostConns/possibleConns: 0

'usable' after dropping 450 packs with 15 connections:
free CBR: 30          lostCBR/linkRate: 0
free conns: 21        lostConns/possibleConns: 0

'usable' after dropping 600 packs with 15 connections:
free CBR: 30          lostCBR/linkRate: 0
free conns: 21        lostConns/possibleConns: 0

==================================================

linkRate: 100  Rbin: 100  addBins: 100  cellLossProb: 1e-05  overBookingRa-
tio: 1000
packSize: 25  printAfter:150  maxRuns:600  initConns:10
TraffDesc: connId=0  TrafficType= 1nb of states=2
rates:        0        3
probs:        0.6      0.4

initial free CBR bandwidth 73
initial free nb of Conns 41

'usable' after dropping 150 packs with 25 connections:
free CBR: 73          lostCBR/linkRate: 0
free conns: 41        lostConns/possibleConns: 0

'usable' after dropping 300 packs with 25 connections:
free CBR: 73          lostCBR/linkRate: 0
free conns: 41        lostConns/possibleConns: 0

'usable' after dropping 450 packs with 25 connections:
free CBR: 73          lostCBR/linkRate: 0
free conns: 41        lostConns/possibleConns: 0

'usable' after dropping 600 packs with 25 connections:
free CBR: 73          lostCBR/linkRate: 0
free conns: 41        lostConns/possibleConns: 0

==================================================

linkRate: 100  Rbin: 100  addBins: 100  cellLossProb: 1e-05  overBookingRa-
tio: 1000
packSize: 15  printAfter:150  maxRuns:600  initConns:20
TraffDesc: connId=0  TrafficType= 1nb of states=2
rates:        0        3
probs:        0.6      0.4

initial free CBR bandwidth 53
initial free nb of Conns 31

'usable' after dropping 150 packs with 15 connections:
free CBR: 53          lostCBR/linkRate: 0
free conns: 31        lostConns/possibleConns: 0

'usable' after dropping 300 packs with 15 connections:
free CBR: 53          lostCBR/linkRate: 0
free conns: 31        lostConns/possibleConns: 0

'usable' after dropping 450 packs with 15 connections:
free CBR: 53          lostCBR/linkRate: 0
free conns: 31        lostConns/possibleConns: 0

'usable' after dropping 600 packs with 15 connections:
free CBR: 53          lostCBR/linkRate: 0
free conns: 31        lostConns/possibleConns: 0

==================================================

linkRate: 100  Rbin: 100  addBins: 100  cellLossProb: 1e-05  overBookingRa-
tio: 1000
packSize: 5  printAfter:150  maxRuns:600  initConns:30
TraffDesc: connId=0  TrafficType= 1nb of states=2
rates:        0        3
probs:        0.6      0.4

initial free CBR bandwidth 36
initial free nb of Conns 21

'usable' after dropping 150 packs with 5 connections:
free CBR: 36          lostCBR/linkRate: 0
free conns: 21        lostConns/possibleConns: 0

'usable' after dropping 300 packs with 5 connections:
free CBR: 36          lostCBR/linkRate: 0
free conns: 21        lostConns/possibleConns: 0

'usable' after dropping 450 packs with 5 connections:
free CBR: 36          lostCBR/linkRate: 0
free conns: 21        lostConns/possibleConns: 0

'usable' after dropping 600 packs with 5 connections:
free CBR: 36          lostCBR/linkRate: 0
free conns: 21        lostConns/possibleConns: 0

==================================================
```

12.1.4 Reducing p_1/p_i

These tests show how the calculation error increases when this value is below 0.6.

```
linkRate: 100  Rbin: 100  addBins: 100  cellLossProb: 1e-05  overBookingRa-
tio: 1000
packSize: 41  printAfter:150  maxRuns:600  initConns:19
TraffDesc: connId=0  TrafficType= 1nb of states=4
rates:        0        1        2        3
```

```
probs:      0.5     0.2     0.15     0.15

initial free CBR bandwidth 65
initial free nb of Conns 50

'usable' after dropping 150 packs with 41 connections:
free CBR: 67          lostCBR/linkRate: -0.02
free conns: 45        lostConns/possibleConns: 0.0724638

'usable' after dropping 300 packs with 41 connections:
free CBR: 67          lostCBR/linkRate: -0.02
free conns: 45        lostConns/possibleConns: 0.0724638

'usable' after dropping 450 packs with 41 connections:
free CBR: 69          lostCBR/linkRate: -0.04
free conns: 45        lostConns/possibleConns: 0.0724638

'usable' after dropping 600 packs with 41 connections:
free CBR: 68          lostCBR/linkRate: -0.03
free conns: 45        lostConns/possibleConns: 0.0724638

=====================================================

linkRate: 100 Rbin: 100 addBins: 100 cellLossProb: 1e-05 overBookingRa-
tio: 1000
packSize: 41 printAfter:150 maxRuns:600 initConns:19
TraffDesc: connId=0 TrafficType= 1nb of states=4
rates:      0       1       2       3
probs:      0.5     0.17    0.17    0.16

initial free CBR bandwidth 64
initial free nb of Conns 48

'usable' after dropping 150 packs with 41 connections:
free CBR: 62          lostCBR/linkRate: 0.02
free conns: 42        lostConns/possibleConns: 0.0895522

'usable' after dropping 300 packs with 41 connections:
free CBR: 65          lostCBR/linkRate: -0.01
free conns: 44        lostConns/possibleConns: 0.0597015

'usable' after dropping 450 packs with 41 connections:
free CBR: 67          lostCBR/linkRate: -0.03
free conns: 42        lostConns/possibleConns: 0.0895522

'usable' after dropping 600 packs with 41 connections:
free CBR: 65          lostCBR/linkRate: -0.01
free conns: 42        lostConns/possibleConns: 0.0895522

=====================================================

linkRate: 100 Rbin: 100 addBins: 100 cellLossProb: 1e-05 overBookingRa-
tio: 1000
packSize: 35 printAfter:150 maxRuns:600 initConns:20
TraffDesc: connId=0 TrafficType= 1nb of states=4
rates:      0       1       3       4
probs:      0.5     0.3     0.1     0.1

initial free CBR bandwidth 57
initial free nb of Conns 40

'usable' after dropping 150 packs with 35 connections:
free CBR: 71          lostCBR/linkRate: -0.14
free conns: 37        lostConns/possibleConns: 0.05

'usable' after dropping 300 packs with 35 connections:
free CBR: 71          lostCBR/linkRate: -0.14
free conns: 37        lostConns/possibleConns: 0.05

'usable' after dropping 450 packs with 35 connections:
free CBR: 72          lostCBR/linkRate: -0.15
free conns: 37        lostConns/possibleConns: 0.05

'usable' after dropping 600 packs with 35 connections:
free CBR: 68          lostCBR/linkRate: -0.11
free conns: 37        lostConns/possibleConns: 0.05

=====================================================

linkRate: 100 Rbin: 100 addBins: 100 cellLossProb: 1e-05 overBookingRa-
tio: 1000
packSize: 41 printAfter:150 maxRuns:600 initConns:21
TraffDesc: connId=0 TrafficType= 1nb of states=4
rates:      0       1       2       3
probs:      0.5     0.1     0.3     0.1

initial free CBR bandwidth 62
initial free nb of Conns 47

'usable' after dropping 150 packs with 41 connections:
free CBR: 59          lostCBR/linkRate: 0.03
free conns: 41        lostConns/possibleConns: 0.0882353

'usable' after dropping 300 packs with 41 connections:
free CBR: 60          lostCBR/linkRate: 0.02
free conns: 42        lostConns/possibleConns: 0.0735294

'usable' after dropping 450 packs with 41 connections:
free CBR: 63          lostCBR/linkRate: -0.01
free conns: 42        lostConns/possibleConns: 0.0735294

'usable' after dropping 600 packs with 41 connections:
free CBR: 61          lostCBR/linkRate: 0.01
free conns: 41        lostConns/possibleConns: 0.0882353

=====================================================

linkRate: 100 Rbin: 100 addBins: 100 cellLossProb: 1e-05 overBookingRa-
tio: 1000
packSize: 35 printAfter:150 maxRuns:600 initConns:19
TraffDesc: connId=0 TrafficType= 1nb of states=4
rates:      0       1       2       3
probs:      0.45    0.19    0.19    0.17

initial free CBR bandwidth 62
initial free nb of Conns 43

'usable' after dropping 150 packs with 35 connections:
free CBR: 65          lostCBR/linkRate: -0.03
free conns: 39        lostConns/possibleConns: 0.0645161

'usable' after dropping 300 packs with 35 connections:
free CBR: 64          lostCBR/linkRate: -0.02
free conns: 39        lostConns/possibleConns: 0.0645161

'usable' after dropping 450 packs with 35 connections:
free CBR: 64          lostCBR/linkRate: -0.02
free conns: 39        lostConns/possibleConns: 0.0645161

'usable' after dropping 600 packs with 35 connections:
free CBR: 65          lostCBR/linkRate: -0.03
free conns: 38        lostConns/possibleConns: 0.0806452

=====================================================

linkRate: 100 Rbin: 100 addBins: 100 cellLossProb: 1e-05 overBookingRa-
tio: 1000
packSize: 20 printAfter:150 maxRuns:600 initConns:19
TraffDesc: connId=0 TrafficType= 1nb of states=4
rates:      0       1       2       3
probs:      0.4     0.2     0.2     0.2

initial free CBR bandwidth 60
initial free nb of Conns 38

'usable' after dropping 150 packs with 20 connections:
free CBR: 64          lostCBR/linkRate: -0.04
free conns: 34        lostConns/possibleConns: 0.0701754

'usable' after dropping 300 packs with 20 connections:
free CBR: 63          lostCBR/linkRate: -0.03
free conns: 34        lostConns/possibleConns: 0.0701754

'usable' after dropping 450 packs with 20 connections:
free CBR: 63          lostCBR/linkRate: -0.03
free conns: 34        lostConns/possibleConns: 0.0701754

'usable' after dropping 600 packs with 20 connections:
free CBR: 63          lostCBR/linkRate: -0.03
free conns: 34        lostConns/possibleConns: 0.0701754

=====================================================

linkRate: 100 Rbin: 100 addBins: 100 cellLossProb: 1e-05 overBookingRa-
tio: 1000
packSize: 20 printAfter:150 maxRuns:600 initConns:30
TraffDesc: connId=0 TrafficType= 1nb of states=3
rates:      0       1       3
probs:      0.6     0.1     0.3

initial free CBR bandwidth 44
initial free nb of Conns 31

'usable' after dropping 150 packs with 20 connections:
free CBR: 44          lostCBR/linkRate: 0
free conns: 30        lostConns/possibleConns: 0.0163934

'usable' after dropping 300 packs with 20 connections:
free CBR: 44          lostCBR/linkRate: 0
free conns: 31        lostConns/possibleConns: 0

'usable' after dropping 450 packs with 20 connections:
free CBR: 44          lostCBR/linkRate: 0
free conns: 31        lostConns/possibleConns: 0

'usable' after dropping 600 packs with 20 connections:
free CBR: 44          lostCBR/linkRate: 0
free conns: 30        lostConns/possibleConns: 0.0163934
```

```
=====================================================

linkRate: 100 Rbin: 100 addBins: 100 cellLossProb: 1e-05 overBookingRa-
tio: 1000
packSize: 20 printAfter:150 maxRuns:600 initConns:20
TraffDesc: connId=0 TrafficType= 1nb of states=3
rates:       0       1       3
probs:       0.5     0.1     0.4

initial free CBR bandwidth 53
initial free nb of Conns 29

'usable' after dropping 150 packs with 20 connections:
free CBR: 53            lostCBR/linkRate: 0
free conns: 29          lostConns/possibleConns: 0

'usable' after dropping 300 packs with 20 connections:
free CBR: 53            lostCBR/linkRate: 0
free conns: 29          lostConns/possibleConns: 0

'usable' after dropping 450 packs with 20 connections:
free CBR: 52            lostCBR/linkRate: 0.01
free conns: 28          lostConns/possibleConns: 0.0204082

'usable' after dropping 600 packs with 20 connections:
free CBR: 53            lostCBR/linkRate: 0
free conns: 29          lostConns/possibleConns: 0

=====================================================

linkRate: 100 Rbin: 100 addBins: 100 cellLossProb: 1e-05 overBookingRa-
tio: 1000
packSize: 25 printAfter:150 maxRuns:600 initConns:30
TraffDesc: connId=0 TrafficType= 1nb of states=3
rates:       0       1       3
probs:       0.5     0.25    0.25

initial free CBR bandwidth 46
initial free nb of Conns 34

'usable' after dropping 150 packs with 25 connections:
free CBR: 56            lostCBR/linkRate: -0.1
free conns: 31          lostConns/possibleConns: 0.046875

'usable' after dropping 300 packs with 25 connections:
free CBR: 53            lostCBR/linkRate: -0.07
free conns: 31          lostConns/possibleConns: 0.046875

'usable' after dropping 450 packs with 25 connections:
free CBR: 56            lostCBR/linkRate: -0.1
free conns: 31          lostConns/possibleConns: 0.046875

'usable' after dropping 600 packs with 25 connections:
free CBR: 55            lostCBR/linkRate: -0.09
free conns: 31          lostConns/possibleConns: 0.046875

=====================================================

linkRate: 100 Rbin: 100 addBins: 100 cellLossProb: 1e-05 overBookingRa-
tio: 1000
packSize: 30 printAfter:150 maxRuns:600 initConns:20
TraffDesc: connId=0 TrafficType= 1nb of states=3
rates:       0       1       3
probs:       0.4     0.3     0.3

initial free CBR bandwidth 57
initial free nb of Conns 35

'usable' after dropping 150 packs with 30 connections:
free CBR: 70            lostCBR/linkRate: -0.13
free conns: 34          lostConns/possibleConns: 0.0181818

'usable' after dropping 300 packs with 30 connections:
free CBR: 72            lostCBR/linkRate: -0.15
free conns: 34          lostConns/possibleConns: 0.0181818

'usable' after dropping 450 packs with 30 connections:
free CBR: 71            lostCBR/linkRate: -0.14
free conns: 34          lostConns/possibleConns: 0.0181818

'usable' after dropping 600 packs with 30 connections:
free CBR: 70            lostCBR/linkRate: -0.13
free conns: 34          lostConns/possibleConns: 0.0181818

=====================================================

linkRate: 100 Rbin: 100 addBins: 100 cellLossProb: 1e-05 overBookingRa-
tio: 1000
packSize: 25 printAfter:150 maxRuns:600 initConns:20
TraffDesc: connId=0 TrafficType= 1nb of states=2
rates:       0       3
probs:       0.6     0.4

initial free CBR bandwidth 53
initial free nb of Conns 31

'usable' after dropping 150 packs with 25 connections:
free CBR: 53            lostCBR/linkRate: 0
free conns: 31          lostConns/possibleConns: 0

'usable' after dropping 300 packs with 25 connections:
free CBR: 53            lostCBR/linkRate: 0
free conns: 31          lostConns/possibleConns: 0

'usable' after dropping 450 packs with 25 connections:
free CBR: 53            lostCBR/linkRate: 0
free conns: 31          lostConns/possibleConns: 0

'usable' after dropping 600 packs with 25 connections:
free CBR: 53            lostCBR/linkRate: 0
free conns: 31          lostConns/possibleConns: 0

=====================================================

linkRate: 100 Rbin: 100 addBins: 100 cellLossProb: 1e-05 overBookingRa-
tio: 1000
packSize: 20 printAfter:150 maxRuns:600 initConns:20
TraffDesc: connId=0 TrafficType= 1nb of states=2
rates:       0       3
probs:       0.55    0.45

initial free CBR bandwidth 51
initial free nb of Conns 27

'usable' after dropping 150 packs with 20 connections:
free CBR: 51            lostCBR/linkRate: 0
free conns: 27          lostConns/possibleConns: 0

'usable' after dropping 300 packs with 20 connections:
free CBR: 51            lostCBR/linkRate: 0
free conns: 27          lostConns/possibleConns: 0

'usable' after dropping 450 packs with 20 connections:
free CBR: 51            lostCBR/linkRate: 0
free conns: 27          lostConns/possibleConns: 0

'usable' after dropping 600 packs with 20 connections:
free CBR: 51            lostCBR/linkRate: 0
free conns: 27          lostConns/possibleConns: 0

=====================================================

linkRate: 100 Rbin: 100 addBins: 100 cellLossProb: 1e-05 overBookingRa-
tio: 1000
packSize: 30 printAfter:150 maxRuns:600 initConns:30
TraffDesc: connId=0 TrafficType= 1nb of states=2
rates:       1       2
probs:       0.55    0.45

initial free CBR bandwidth 56
initial free nb of Conns 38

'usable' after dropping 150 packs with 30 connections:
free CBR: 56            lostCBR/linkRate: 0
free conns: 38          lostConns/possibleConns: 0

'usable' after dropping 300 packs with 30 connections:
free CBR: 56            lostCBR/linkRate: 0
free conns: 38          lostConns/possibleConns: 0

'usable' after dropping 450 packs with 30 connections:
free CBR: 56            lostCBR/linkRate: 0
free conns: 38          lostConns/possibleConns: 0

'usable' after dropping 600 packs with 30 connections:
free CBR: 56            lostCBR/linkRate: 0
free conns: 38          lostConns/possibleConns: 0

=====================================================

linkRate: 100 Rbin: 100 addBins: 100 cellLossProb: 1e-05 overBookingRa-
tio: 1000
packSize: 25 printAfter:150 maxRuns:600 initConns:20
TraffDesc: connId=0 TrafficType= 1nb of states=2
rates:       1       3
probs:       0.55    0.45

initial free CBR bandwidth 62
initial free nb of Conns 32

'usable' after dropping 150 packs with 25 connections:
free CBR: 62            lostCBR/linkRate: 0
free conns: 32          lostConns/possibleConns: 0

'usable' after dropping 300 packs with 25 connections:
free CBR: 62            lostCBR/linkRate: 0
free conns: 32          lostConns/possibleConns: 0
```

```
'usable' after dropping 450 packs with 25 connections:
free CBR: 62          lostCBR/linkRate: 0
free conns: 32        lostConns/possibleConns: 0

'usable' after dropping 600 packs with 25 connections:
free CBR: 62          lostCBR/linkRate: 0
free conns: 32        lostConns/possibleConns: 0

--------------------------------------------------------

linkRate: 100  Rbin: 100  addBins: 100  cellLossProb: 1e-05  overBookingRa-
tio: 1000
packSize: 10  printAfter:150  maxRuns:600  initConns:50
TraffDesc: connId=0  TrafficType= 1nb of states=2
rates:        0        2
probs:        0.5      0.5

initial free CBR bandwidth 26
initial free nb of Conns 21

'usable' after dropping 150 packs with 10 connections:
free CBR: 26          lostCBR/linkRate: 0
free conns: 20        lostConns/possibleConns: 0.0140845

'usable' after dropping 300 packs with 10 connections:
free CBR: 26          lostCBR/linkRate: 0
free conns: 21        lostConns/possibleConns: 0

'usable' after dropping 450 packs with 10 connections:
free CBR: 26          lostCBR/linkRate: 0
free conns: 21        lostConns/possibleConns: 0

'usable' after dropping 600 packs with 10 connections:
free CBR: 26          lostCBR/linkRate: 0
free conns: 21        lostConns/possibleConns: 0

--------------------------------------------------------
```

12.2 500 Bins for Aggregate Traffic

If the granularity is increased and especially small rate traffic descriptors are treated as normal connections (rather than treat them as CBR traffic) the possible number of VBR connections in the aggregate traffic increases and the calculation error explodes. The following examples could never be used in the real world since the calculation error is just too big. As mentioned, the calculation error is reduced to an acceptable level while still guaranteeing a fine level of granularity, if small rate traffic descriptors are treated as CBR traffic so that they cannot increase the calculation error.

12.2.1 Four State Descriptors

```
linkRate: 500  Rbin: 500  addBins: 100  cellLossProb: 1e-05  overBookingRa-
tio: 1000
packSize: 150  printAfter:150  maxRuns:600  initConns:50
TraffDesc: connId=0  TrafficType= 1nb of states=4
rates:        0        1        3        5
probs:        0.6      0.2      0.1      0.1

initial free CBR bandwidth 412
initial free nb of Conns 339

'usable' after dropping 150 packs with 150 connections:
free CBR: 479         lostCBR/linkRate: -0.134
free conns: 346       lostConns/possibleConns: -0.0179949

'usable' after dropping 300 packs with 150 connections:
free CBR: 480         lostCBR/linkRate: -0.136
free conns: 346       lostConns/possibleConns: -0.0179949

'usable' after dropping 450 packs with 150 connections:
free CBR: 481         lostCBR/linkRate: -0.138
free conns: 346       lostConns/possibleConns: -0.0179949

'usable' after dropping 600 packs with 150 connections:
free CBR: 481         lostCBR/linkRate: -0.138
free conns: 347       lostConns/possibleConns: -0.0205656

--------------------------------------------------------

linkRate: 500  Rbin: 500  addBins: 100  cellLossProb: 1e-05  overBookingRa-
tio: 1000
packSize: 100  printAfter:150  maxRuns:600  initConns:100
TraffDesc: connId=0  TrafficType= 1nb of states=4
rates:        0        1        3        5
probs:        0.6      0.2      0.1      0.1

initial free CBR bandwidth 346
initial free nb of Conns 289

'usable' after dropping 150 packs with 100 connections:
free CBR: 465         lostCBR/linkRate: -0.238
free conns: 334       lostConns/possibleConns: -0.115681

'usable' after dropping 300 packs with 100 connections:
free CBR: 469         lostCBR/linkRate: -0.246
free conns: 336       lostConns/possibleConns: -0.120823

'usable' after dropping 450 packs with 100 connections:
free CBR: 468         lostCBR/linkRate: -0.244
free conns: 336       lostConns/possibleConns: -0.120823

'usable' after dropping 600 packs with 100 connections:
free CBR: 471         lostCBR/linkRate: -0.25
free conns: 338       lostConns/possibleConns: -0.125964

--------------------------------------------------------

linkRate: 500  Rbin: 500  addBins: 100  cellLossProb: 1e-05  overBookingRa-
tio: 1000
packSize: 50  printAfter:150  maxRuns:600  initConns:150
TraffDesc: connId=0  TrafficType= 1nb of states=4
rates:        0        1        3        5
probs:        0.6      0.2      0.1      0.1

initial free CBR bandwidth 283
initial free nb of Conns 239

'usable' after dropping 150 packs with 50 connections:
free CBR: 441         lostCBR/linkRate: -0.316
free conns: 313       lostConns/possibleConns: -0.190231

'usable' after dropping 300 packs with 50 connections:
free CBR: 451         lostCBR/linkRate: -0.336
free conns: 323       lostConns/possibleConns: -0.215938

'usable' after dropping 450 packs with 50 connections:
free CBR: 456         lostCBR/linkRate: -0.346
free conns: 326       lostConns/possibleConns: -0.22365

'usable' after dropping 600 packs with 50 connections:
free CBR: 458         lostCBR/linkRate: -0.35
free conns: 327       lostConns/possibleConns: -0.226221

--------------------------------------------------------

linkRate: 500  Rbin: 500  addBins: 100  cellLossProb: 1e-05  overBookingRa-
tio: 1000
packSize: 150  printAfter:150  maxRuns:600  initConns:50
TraffDesc: connId=0  TrafficType= 1nb of states=4
rates:        0        1        3        5
probs:        0.6      0.14     0.13     0.13

initial free CBR bandwidth 399
initial free nb of Conns 279

'usable' after dropping 150 packs with 150 connections:
free CBR: 473         lostCBR/linkRate: -0.148
free conns: 289       lostConns/possibleConns: -0.0303951

'usable' after dropping 300 packs with 150 connections:
free CBR: 476         lostCBR/linkRate: -0.154
free conns: 290       lostConns/possibleConns: -0.0334347

'usable' after dropping 450 packs with 150 connections:
free CBR: 475         lostCBR/linkRate: -0.152
free conns: 290       lostConns/possibleConns: -0.0334347

'usable' after dropping 600 packs with 150 connections:
free CBR: 475         lostCBR/linkRate: -0.152
free conns: 290       lostConns/possibleConns: -0.0334347

--------------------------------------------------------

linkRate: 500  Rbin: 500  addBins: 100  cellLossProb: 1e-05  overBookingRa-
tio: 1000
```

```
packSize: 100 printAfter:150 maxRuns:600 initConns:100
TraffDesc: connId=0 TrafficType= 1nb of states=4
rates:      0       1       3       5
probs:      0.6     0.14    0.13    0.13

initial free CBR bandwidth 322
initial free nb of Conns 229

'usable' after dropping 150 packs with 100 connections:
free CBR: 436        lostCBR/linkRate: -0.228
free conns: 263      lostConns/possibleConns: -0.103343

'usable' after dropping 300 packs with 100 connections:
free CBR: 458        lostCBR/linkRate: -0.272
free conns: 276      lostConns/possibleConns: -0.142857

'usable' after dropping 450 packs with 100 connections:
free CBR: 459        lostCBR/linkRate: -0.274
free conns: 278      lostConns/possibleConns: -0.148936

'usable' after dropping 600 packs with 100 connections:
free CBR: 461        lostCBR/linkRate: -0.278
free conns: 279      lostConns/possibleConns: -0.151976

----------------------------------------------------------------

linkRate: 500 Rbin: 500 addBins: 100 cellLossProb: 1e-05 overBookingRa-
tio: 1000
packSize: 50 printAfter:150 maxRuns:600 initConns:150
TraffDesc: connId=0 TrafficType= 1nb of states=4
rates:      0       1       3       5
probs:      0.6     0.14    0.13    0.13

initial free CBR bandwidth 249
initial free nb of Conns 179

'usable' after dropping 150 packs with 50 connections:
free CBR: 413        lostCBR/linkRate: -0.328
free conns: 244      lostConns/possibleConns: -0.197568

'usable' after dropping 300 packs with 50 connections:
free CBR: 422        lostCBR/linkRate: -0.346
free conns: 251      lostConns/possibleConns: -0.218845

'usable' after dropping 450 packs with 50 connections:
free CBR: 437        lostCBR/linkRate: -0.376
free conns: 262      lostConns/possibleConns: -0.25228

'usable' after dropping 600 packs with 50 connections:
free CBR: 440        lostCBR/linkRate: -0.382
free conns: 263      lostConns/possibleConns: -0.255319

----------------------------------------------------------------

linkRate: 500 Rbin: 500 addBins: 100 cellLossProb: 1e-05 overBookingRa-
tio: 1000
packSize: 150 printAfter:150 maxRuns:600 initConns:50
TraffDesc: connId=0 TrafficType= 1nb of states=4
rates:      0       3       4       5
probs:      0.6     0.38    0.01    0.01

initial free CBR bandwidth 405
initial free nb of Conns 281

'usable' after dropping 150 packs with 150 connections:
free CBR: 451        lostCBR/linkRate: -0.092
free conns: 278      lostConns/possibleConns: 0.00906344

'usable' after dropping 300 packs with 150 connections:
free CBR: 451        lostCBR/linkRate: -0.092
free conns: 278      lostConns/possibleConns: 0.00906344

'usable' after dropping 450 packs with 150 connections:
free CBR: 450        lostCBR/linkRate: -0.09
free conns: 278      lostConns/possibleConns: 0.00906344

'usable' after dropping 600 packs with 150 connections:
free CBR: 448        lostCBR/linkRate: -0.086
free conns: 277      lostConns/possibleConns: 0.0120846

----------------------------------------------------------------

linkRate: 500 Rbin: 500 addBins: 100 cellLossProb: 1e-05 overBookingRa-
tio: 1000
packSize: 100 printAfter:150 maxRuns:600 initConns:100
TraffDesc: connId=0 TrafficType= 1nb of states=4
rates:      0       3       4       5
probs:      0.6     0.38    0.01    0.01

initial free CBR bandwidth 329
initial free nb of Conns 231

'usable' after dropping 150 packs with 100 connections:
free CBR: 421        lostCBR/linkRate: -0.184
free conns: 257      lostConns/possibleConns: -0.0785498

'usable' after dropping 300 packs with 100 connections:
free CBR: 421        lostCBR/linkRate: -0.184
free conns: 258      lostConns/possibleConns: -0.081571

'usable' after dropping 450 packs with 100 connections:
free CBR: 425        lostCBR/linkRate: -0.192
free conns: 260      lostConns/possibleConns: -0.0876133

'usable' after dropping 600 packs with 100 connections:
free CBR: 424        lostCBR/linkRate: -0.19
free conns: 261      lostConns/possibleConns: -0.0906344

----------------------------------------------------------------

linkRate: 500 Rbin: 500 addBins: 100 cellLossProb: 1e-05 overBookingRa-
tio: 1000
packSize: 50 printAfter:150 maxRuns:600 initConns:150
TraffDesc: connId=0 TrafficType= 1nb of states=4
rates:      0       3       4       5
probs:      0.6     0.38    0.01    0.01

initial free CBR bandwidth 256
initial free nb of Conns 181

'usable' after dropping 150 packs with 50 connections:
free CBR: 383        lostCBR/linkRate: -0.254
free conns: 230      lostConns/possibleConns: -0.148036

'usable' after dropping 300 packs with 50 connections:
free CBR: 391        lostCBR/linkRate: -0.27
free conns: 236      lostConns/possibleConns: -0.166163

'usable' after dropping 450 packs with 50 connections:
free CBR: 394        lostCBR/linkRate: -0.276
free conns: 238      lostConns/possibleConns: -0.172205

'usable' after dropping 600 packs with 50 connections:
free CBR: 395        lostCBR/linkRate: -0.278
free conns: 239      lostConns/possibleConns: -0.175227

----------------------------------------------------------------

linkRate: 500 Rbin: 500 addBins: 100 cellLossProb: 1e-05 overBookingRa-
tio: 1000
packSize: 150 printAfter:150 maxRuns:600 initConns:50
TraffDesc: connId=0 TrafficType= 1nb of states=4
rates:      0       3       5       7
probs:      0.6     0.38    0.01    0.01

initial free CBR bandwidth 401
initial free nb of Conns 270

'usable' after dropping 150 packs with 150 connections:
free CBR: 449        lostCBR/linkRate: -0.096
free conns: 267      lostConns/possibleConns: 0.009375

'usable' after dropping 300 packs with 150 connections:
free CBR: 450        lostCBR/linkRate: -0.098
free conns: 267      lostConns/possibleConns: 0.009375

'usable' after dropping 450 packs with 150 connections:
free CBR: 451        lostCBR/linkRate: -0.1
free conns: 268      lostConns/possibleConns: 0.00625

'usable' after dropping 600 packs with 150 connections:
free CBR: 451        lostCBR/linkRate: -0.1
free conns: 268      lostConns/possibleConns: 0.00625

----------------------------------------------------------------

linkRate: 500 Rbin: 500 addBins: 100 cellLossProb: 1e-05 overBookingRa-
tio: 1000
packSize: 100 printAfter:150 maxRuns:600 initConns:100
TraffDesc: connId=0 TrafficType= 1nb of states=4
rates:      0       3       5       7
probs:      0.6     0.38    0.01    0.01

initial free CBR bandwidth 322
initial free nb of Conns 220

'usable' after dropping 150 packs with 100 connections:
free CBR: 414        lostCBR/linkRate: -0.184
free conns: 245      lostConns/possibleConns: -0.078125

'usable' after dropping 300 packs with 100 connections:
free CBR: 425        lostCBR/linkRate: -0.206
free conns: 249      lostConns/possibleConns: -0.090625

'usable' after dropping 450 packs with 100 connections:
free CBR: 420        lostCBR/linkRate: -0.196
free conns: 249      lostConns/possibleConns: -0.090625
```

```
'usable' after dropping 600 packs with 100 connections:
free CBR: 421          lostCBR/linkRate: -0.198
free conns: 250        lostConns/possibleConns: -0.09375
```

```
---------------------------------------------------------------

linkRate: 500  Rbin: 500  addBins: 100  cellLossProb: 1e-05  overBookingRa-
tio: 1000
packSize: 50  printAfter:150  maxRuns:600  initConns:150
TraffDesc: connId=0  TrafficType= 1nb of states=4
rates:        0       3       5       7
probs:        0.6     0.38    0.01    0.01

initial free CBR bandwidth 247
initial free nb of Conns 170

'usable' after dropping 150 packs with 50 connections:
free CBR: 379          lostCBR/linkRate: -0.264
free conns: 218        lostConns/possibleConns: -0.15

'usable' after dropping 300 packs with 50 connections:
free CBR: 392          lostCBR/linkRate: -0.29
free conns: 227        lostConns/possibleConns: -0.178125

'usable' after dropping 450 packs with 50 connections:
free CBR: 392          lostCBR/linkRate: -0.29
free conns: 227        lostConns/possibleConns: -0.178125

'usable' after dropping 600 packs with 50 connections:
free CBR: 392          lostCBR/linkRate: -0.29
free conns: 227        lostConns/possibleConns: -0.178125
```

12.2.2 Three State Descriptors

```
---------------------------------------------------------------

linkRate: 500  Rbin: 500  addBins: 100  cellLossProb: 1e-05  overBookingRa-
tio: 1000
packSize: 150  printAfter:150  maxRuns:600  initConns:50
TraffDesc: connId=0  TrafficType= 1nb of states=3
rates:        0       1       3
probs:        0.6     0.1     0.3

initial free CBR bandwidth 421
initial free nb of Conns 359

'usable' after dropping 150 packs with 150 connections:
free CBR: 476          lostCBR/linkRate: -0.11
free conns: 364        lostConns/possibleConns: -0.0122249

'usable' after dropping 300 packs with 150 connections:
free CBR: 472          lostCBR/linkRate: -0.102
free conns: 364        lostConns/possibleConns: -0.0122249

'usable' after dropping 450 packs with 150 connections:
free CBR: 476          lostCBR/linkRate: -0.11
free conns: 364        lostConns/possibleConns: -0.0122249

'usable' after dropping 600 packs with 150 connections:
free CBR: 476          lostCBR/linkRate: -0.11
free conns: 365        lostConns/possibleConns: -0.0146699

---------------------------------------------------------------

linkRate: 500  Rbin: 500  addBins: 100  cellLossProb: 1e-05  overBookingRa-
tio: 1000
packSize: 100  printAfter:150  maxRuns:600  initConns:100
TraffDesc: connId=0  TrafficType= 1nb of states=3
rates:        0       1       3
probs:        0.6     0.1     0.3

initial free CBR bandwidth 357
initial free nb of Conns 309

'usable' after dropping 150 packs with 100 connections:
free CBR: 455          lostCBR/linkRate: -0.196
free conns: 346        lostConns/possibleConns: -0.0904645

'usable' after dropping 300 packs with 100 connections:
free CBR: 456          lostCBR/linkRate: -0.198
free conns: 347        lostConns/possibleConns: -0.0929095

'usable' after dropping 450 packs with 100 connections:
free CBR: 460          lostCBR/linkRate: -0.206
free conns: 351        lostConns/possibleConns: -0.102689

'usable' after dropping 600 packs with 100 connections:
free CBR: 461          lostCBR/linkRate: -0.208
free conns: 351        lostConns/possibleConns: -0.102689

---------------------------------------------------------------

linkRate: 500  Rbin: 500  addBins: 100  cellLossProb: 1e-05  overBookingRa-
tio: 1000
packSize: 50  printAfter:150  maxRuns:600  initConns:150
TraffDesc: connId=0  TrafficType= 1nb of states=3
rates:        0       1       3
probs:        0.6     0.1     0.3

initial free CBR bandwidth 297
initial free nb of Conns 259

'usable' after dropping 150 packs with 50 connections:
free CBR: 426          lostCBR/linkRate: -0.258
free conns: 321        lostConns/possibleConns: -0.151589

'usable' after dropping 300 packs with 50 connections:
free CBR: 433          lostCBR/linkRate: -0.272
free conns: 327        lostConns/possibleConns: -0.166259

'usable' after dropping 450 packs with 50 connections:
free CBR: 439          lostCBR/linkRate: -0.284
free conns: 333        lostConns/possibleConns: -0.180929

'usable' after dropping 600 packs with 50 connections:
free CBR: 442          lostCBR/linkRate: -0.29
free conns: 335        lostConns/possibleConns: -0.185819

---------------------------------------------------------------

linkRate: 500  Rbin: 500  addBins: 100  cellLossProb: 1e-05  overBookingRa-
tio: 1000
packSize: 150  printAfter:150  maxRuns:600  initConns:50
TraffDesc: connId=0  TrafficType= 1nb of states=3
rates:        0       2       3
probs:        0.6     0.2     0.2

initial free CBR bandwidth 423
initial free nb of Conns 365

'usable' after dropping 150 packs with 150 connections:
free CBR: 467          lostCBR/linkRate: -0.088
free conns: 363        lostConns/possibleConns: 0.00481928

'usable' after dropping 300 packs with 150 connections:
free CBR: 465          lostCBR/linkRate: -0.084
free conns: 363        lostConns/possibleConns: 0.00481928

'usable' after dropping 450 packs with 150 connections:
free CBR: 466          lostCBR/linkRate: -0.086
free conns: 363        lostConns/possibleConns: 0.00481928

'usable' after dropping 600 packs with 150 connections:
free CBR: 466          lostCBR/linkRate: -0.086
free conns: 363        lostConns/possibleConns: 0.00481928

---------------------------------------------------------------

linkRate: 500  Rbin: 500  addBins: 100  cellLossProb: 1e-05  overBookingRa-
tio: 1000
packSize: 100  printAfter:150  maxRuns:600  initConns:100
TraffDesc: connId=0  TrafficType= 1nb of states=3
rates:        0       2       3
probs:        0.6     0.2     0.2

initial free CBR bandwidth 360
initial free nb of Conns 315

'usable' after dropping 150 packs with 100 connections:
free CBR: 442          lostCBR/linkRate: -0.164
free conns: 341        lostConns/possibleConns: -0.0626506

'usable' after dropping 300 packs with 100 connections:
free CBR: 446          lostCBR/linkRate: -0.172
free conns: 344        lostConns/possibleConns: -0.0698795

'usable' after dropping 450 packs with 100 connections:
free CBR: 446          lostCBR/linkRate: -0.172
free conns: 344        lostConns/possibleConns: -0.0698795

'usable' after dropping 600 packs with 100 connections:
free CBR: 444          lostCBR/linkRate: -0.168
free conns: 343        lostConns/possibleConns: -0.0674699

---------------------------------------------------------------

linkRate: 500  Rbin: 500  addBins: 100  cellLossProb: 1e-05  overBookingRa-
tio: 1000
packSize: 50  printAfter:150  maxRuns:600  initConns:150
TraffDesc: connId=0  TrafficType= 1nb of states=3
rates:        0       2       3
probs:        0.6     0.2     0.2

initial free CBR bandwidth 301
initial free nb of Conns 265
```

```
'usable' after dropping 150 packs with 50 connections:
free CBR: 414          lostCBR/linkRate: -0.226
free conns: 316        lostConns/possibleConns: -0.122892

'usable' after dropping 300 packs with 50 connections:
free CBR: 419          lostCBR/linkRate: -0.236
free conns: 323        lostConns/possibleConns: -0.139759

'usable' after dropping 450 packs with 50 connections:
free CBR: 423          lostCBR/linkRate: -0.244
free conns: 324        lostConns/possibleConns: -0.142169

'usable' after dropping 600 packs with 50 connections:
free CBR: 426          lostCBR/linkRate: -0.25
free conns: 327        lostConns/possibleConns: -0.149398

==================================================
```

12.2.3 Two State Descriptors

```
linkRate: 500  Rbin: 500  addBins: 100  cellLossProb: 1e-05  overBookingRa-
tio: 1000
packSize: 150  printAfter:150  maxRuns:600  initConns:13
TraffDesc: connId=0  TrafficType= 1nb of states=2
rates:        1       2
probs:        0.6     0.4

initial free CBR bandwidth 481
initial free nb of Conns 344

'usable' after dropping 150 packs with 150 connections:
free CBR: 499          lostCBR/linkRate: -0.036
free conns: 344        lostConns/possibleConns: 0

'usable' after dropping 300 packs with 150 connections:
free CBR: 499          lostCBR/linkRate: -0.036
free conns: 344        lostConns/possibleConns: 0

'usable' after dropping 450 packs with 150 connections:
free CBR: 499          lostCBR/linkRate: -0.036
free conns: 344        lostConns/possibleConns: 0

'usable' after dropping 600 packs with 150 connections:
free CBR: 499          lostCBR/linkRate: -0.036
free conns: 344        lostConns/possibleConns: 0

==================================================

linkRate: 500  Rbin: 500  addBins: 100  cellLossProb: 1e-05  overBookingRa-
tio: 1000
packSize: 100  printAfter:150  maxRuns:600  initConns:18
TraffDesc: connId=0  TrafficType= 1nb of states=2
rates:        1       2
probs:        0.6     0.4

initial free CBR bandwidth 474
initial free nb of Conns 339

'usable' after dropping 150 packs with 100 connections:
free CBR: 499          lostCBR/linkRate: -0.05
free conns: 339        lostConns/possibleConns: 0

'usable' after dropping 300 packs with 100 connections:
free CBR: 499          lostCBR/linkRate: -0.05
free conns: 339        lostConns/possibleConns: 0

'usable' after dropping 450 packs with 100 connections:
free CBR: 499          lostCBR/linkRate: -0.05
free conns: 339        lostConns/possibleConns: 0

'usable' after dropping 600 packs with 100 connections:
free CBR: 499          lostCBR/linkRate: -0.05
free conns: 339        lostConns/possibleConns: 0

==================================================

linkRate: 500  Rbin: 500  addBins: 100  cellLossProb: 1e-05  overBookingRa-
tio: 1000
packSize: 50  printAfter:150  maxRuns:600  initConns:8
TraffDesc: connId=0  TrafficType= 1nb of states=2
rates:        1       2
probs:        0.6     0.4

initial free CBR bandwidth 488
initial free nb of Conns 349

'usable' after dropping 150 packs with 50 connections:
free CBR: 499          lostCBR/linkRate: -0.022
free conns: 349        lostConns/possibleConns: 0

'usable' after dropping 300 packs with 50 connections:
free CBR: 499          lostCBR/linkRate: -0.022
free conns: 349        lostConns/possibleConns: 0

'usable' after dropping 450 packs with 50 connections:
free CBR: 499          lostCBR/linkRate: -0.022
free conns: 349        lostConns/possibleConns: 0

'usable' after dropping 600 packs with 50 connections:
free CBR: 499          lostCBR/linkRate: -0.022
free conns: 349        lostConns/possibleConns: 0

==================================================

linkRate: 500  Rbin: 500  addBins: 100  cellLossProb: 1e-05  overBookingRa-
tio: 1000
packSize: 150  printAfter:150  maxRuns:600  initConns:50
TraffDesc: connId=0  TrafficType= 1nb of states=2
rates:        0       3
probs:        0.6     0.4

initial free CBR bandwidth 408
initial free nb of Conns 292

'usable' after dropping 150 packs with 150 connections:
free CBR: 452          lostCBR/linkRate: -0.088
free conns: 287        lostConns/possibleConns: 0.0146199

'usable' after dropping 300 packs with 150 connections:
free CBR: 452          lostCBR/linkRate: -0.088
free conns: 287        lostConns/possibleConns: 0.0146199

'usable' after dropping 450 packs with 150 connections:
free CBR: 443          lostCBR/linkRate: -0.07
free conns: 285        lostConns/possibleConns: 0.0204678

'usable' after dropping 600 packs with 150 connections:
free CBR: 455          lostCBR/linkRate: -0.094
free conns: 289        lostConns/possibleConns: 0.00877193

==================================================

linkRate: 500  Rbin: 500  addBins: 100  cellLossProb: 1e-05  overBookingRa-
tio: 1000
packSize: 100  printAfter:150  maxRuns:600  initConns:100
TraffDesc: connId=0  TrafficType= 1nb of states=2
rates:        0       3
probs:        0.6     0.4

initial free CBR bandwidth 334
initial free nb of Conns 242

'usable' after dropping 150 packs with 100 connections:
free CBR: 413          lostCBR/linkRate: -0.158
free conns: 264        lostConns/possibleConns: -0.0643275

'usable' after dropping 300 packs with 100 connections:
free CBR: 422          lostCBR/linkRate: -0.176
free conns: 267        lostConns/possibleConns: -0.0730994

'usable' after dropping 450 packs with 100 connections:
free CBR: 425          lostCBR/linkRate: -0.182
free conns: 268        lostConns/possibleConns: -0.0760234

'usable' after dropping 600 packs with 100 connections:
free CBR: 428          lostCBR/linkRate: -0.188
free conns: 269        lostConns/possibleConns: -0.0789474

==================================================

linkRate: 500  Rbin: 500  addBins: 100  cellLossProb: 1e-05  overBookingRa-
tio: 1000
packSize: 50  printAfter:150  maxRuns:600  initConns:150
TraffDesc: connId=0  TrafficType= 1nb of states=2
rates:        0       3
probs:        0.6     0.4

initial free CBR bandwidth 263
initial free nb of Conns 192

'usable' after dropping 150 packs with 50 connections:
free CBR: 389          lostCBR/linkRate: -0.252
free conns: 241        lostConns/possibleConns: -0.143275

'usable' after dropping 300 packs with 50 connections:
free CBR: 398          lostCBR/linkRate: -0.27
free conns: 248        lostConns/possibleConns: -0.163743

'usable' after dropping 450 packs with 50 connections:
free CBR: 398          lostCBR/linkRate: -0.27
free conns: 248        lostConns/possibleConns: -0.163743

'usable' after dropping 600 packs with 50 connections:
free CBR: 398          lostCBR/linkRate: -0.27
free conns: 248        lostConns/possibleConns: -0.163743
```

12.3 Additional Bins

This section shows how the number of additional bins (above the link rate) with which the aggregate traffic descriptor is represented influences the precision of the calculations.

12.3.1 Four State Descriptors

```
linkRate: 500 Rbin: 500 addBins: 0 cellLossProb: 1e-05 overBookingRatio:
1000
packSize: 20 printAfter:200 maxRuns:200 initConns:20
TraffDesc: connId=0 TrafficType= 1nb of states=4
rates:      1       2       3       4
probs:      0.6     0.2     0.1     0.1

initial free CBR bandwidth 452
initial free nb of Conns 248

'usable' after dropping 200 packs with 20 connections:
free CBR: 466        lostCBR/linkRate: -0.028
free conns: 234      lostConns/possibleConns: 0.0522388

==================================================

linkRate: 500 Rbin: 500 addBins: 50 cellLossProb: 1e-05 overBookingRatio:
1000
packSize: 20 printAfter:200 maxRuns:200 initConns:20
TraffDesc: connId=0 TrafficType= 1nb of states=4
rates:      1       2       3       4
probs:      0.6     0.2     0.1     0.1

initial free CBR bandwidth 452
initial free nb of Conns 243

'usable' after dropping 200 packs with 20 connections:
free CBR: 466        lostCBR/linkRate: -0.028
free conns: 234      lostConns/possibleConns: 0.0342205

==================================================

linkRate: 500 Rbin: 500 addBins: 100 cellLossProb: 1e-05 overBookingRa-
tio: 1000
packSize: 20 printAfter:200 maxRuns:200 initConns:20
TraffDesc: connId=0 TrafficType= 1nb of states=4
rates:      1       2       3       4
probs:      0.6     0.2     0.1     0.1

initial free CBR bandwidth 452
initial free nb of Conns 243

'usable' after dropping 200 packs with 20 connections:
free CBR: 466        lostCBR/linkRate: -0.028
free conns: 234      lostConns/possibleConns: 0.0342205

==================================================

linkRate: 500 Rbin: 500 addBins: 500 cellLossProb: 1e-05 overBookingRa-
tio: 1000
packSize: 20 printAfter:200 maxRuns:200 initConns:20
TraffDesc: connId=0 TrafficType= 1nb of states=4
rates:      1       2       3       4
probs:      0.6     0.2     0.1     0.1

initial free CBR bandwidth 452
initial free nb of Conns 243

'usable' after dropping 200 packs with 20 connections:
free CBR: 466        lostCBR/linkRate: -0.028
free conns: 234      lostConns/possibleConns: 0.0342205

==================================================
```

12.3.2 Three State Descriptors

```
linkRate: 500 Rbin: 500 addBins: 0 cellLossProb: 1e-05 overBookingRatio:
1000
packSize: 100 printAfter:200 maxRuns:200 initConns:130
TraffDesc: connId=0 TrafficType= 1nb of states=3
rates:      1       2       3
probs:      0.6     0.1     0.3

initial free CBR bandwidth 279
initial free nb of Conns 164

'usable' after dropping 200 packs with 100 connections:
free CBR: 327        lostCBR/linkRate: -0.096
free conns: 164      lostConns/possibleConns: 0

==================================================

linkRate: 500 Rbin: 500 addBins: 50 cellLossProb: 1e-05 overBookingRatio:
1000
packSize: 100 printAfter:200 maxRuns:200 initConns:130
TraffDesc: connId=0 TrafficType= 1nb of states=3
rates:      1       2       3
probs:      0.6     0.1     0.3

initial free CBR bandwidth 279
initial free nb of Conns 164

'usable' after dropping 200 packs with 100 connections:
free CBR: 279        lostCBR/linkRate: 0
free conns: 164      lostConns/possibleConns: 0

==================================================

linkRate: 500 Rbin: 500 addBins: 100 cellLossProb: 1e-05 overBookingRa-
tio: 1000
packSize: 100 printAfter:200 maxRuns:200 initConns:130
TraffDesc: connId=0 TrafficType= 1nb of states=3
rates:      1       2       3
probs:      0.6     0.1     0.3

initial free CBR bandwidth 279
initial free nb of Conns 164

'usable' after dropping 200 packs with 100 connections:
free CBR: 279        lostCBR/linkRate: 0
free conns: 164      lostConns/possibleConns: 0

==================================================

linkRate: 500 Rbin: 500 addBins: 500 cellLossProb: 1e-05 overBookingRa-
tio: 1000
packSize: 100 printAfter:200 maxRuns:200 initConns:130
TraffDesc: connId=0 TrafficType= 1nb of states=3
rates:      1       2       3
probs:      0.6     0.1     0.3

initial free CBR bandwidth 279
initial free nb of Conns 164

'usable' after dropping 200 packs with 100 connections:
free CBR: 279        lostCBR/linkRate: 0
free conns: 164      lostConns/possibleConns: 0

==================================================

linkRate: 500 Rbin: 500 addBins: 0 cellLossProb: 1e-05 overBookingRatio:
1000
packSize: 100 printAfter:200 maxRuns:200 initConns:130
TraffDesc: connId=0 TrafficType= 1nb of states=3
rates:      0       1       2
probs:      0.6     0.1     0.3

initial free CBR bandwidth 378
initial free nb of Conns 499

'usable' after dropping 200 packs with 100 connections:
free CBR: 456        lostCBR/linkRate: -0.156
free conns: 525      lostConns/possibleConns: -0.0413355

==================================================

linkRate: 500 Rbin: 500 addBins: 50 cellLossProb: 1e-05 overBookingRatio:
1000
packSize: 100 printAfter:200 maxRuns:200 initConns:130
TraffDesc: connId=0 TrafficType= 1nb of states=3
rates:      0       1       2
probs:      0.6     0.1     0.3

initial free CBR bandwidth 378
initial free nb of Conns 481

'usable' after dropping 200 packs with 100 connections:
free CBR: 458        lostCBR/linkRate: -0.16
free conns: 527      lostConns/possibleConns: -0.0752864

==================================================

linkRate: 500 Rbin: 500 addBins: 100 cellLossProb: 1e-05 overBookingRa-
tio: 1000
packSize: 100 printAfter:200 maxRuns:200 initConns:130
TraffDesc: connId=0 TrafficType= 1nb of states=3
rates:      0       1       2
probs:      0.6     0.1     0.3
```

```
initial free CBR bandwidth 378
initial free nb of Conns 481

'usable' after dropping 200 packs with 100 connections:
free CBR: 458          lostCBR/linkRate: -0.16
free conns: 527        lostConns/possibleConns: -0.0752864

=====================================================

linkRate: 500  Rbin: 500  addBins: 500  cellLossProb: 1e-05  overBookingRa-
tio: 1000
packSize: 100  printAfter:200  maxRuns:200  initConns:130
TraffDesc: connId=0  TrafficType= 1nb of states=3
rates:       0      1      2
probs:       0.6    0.1    0.3

initial free CBR bandwidth 378
initial free nb of Conns 481

'usable' after dropping 200 packs with 100 connections:
free CBR: 458          lostCBR/linkRate: -0.16
free conns: 527        lostConns/possibleConns: -0.0752864

=====================================================
```

12.3.3 Two State Descriptors

```
linkRate: 500  Rbin: 500  addBins: 0  cellLossProb: 1e-05  overBookingRatio:
1000
packSize: 100  printAfter:200  maxRuns:200  initConns:150
TraffDesc: connId=0  TrafficType= 1nb of states=2
rates:       1      2
probs:       0.6    0.4

initial free CBR bandwidth 290
initial free nb of Conns 207

'usable' after dropping 200 packs with 100 connections:
free CBR: 413          lostCBR/linkRate: -0.246
free conns: 207        lostConns/possibleConns: 0

=====================================================

linkRate: 500  Rbin: 500  addBins: 50  cellLossProb: 1e-05  overBookingRatio:
1000
packSize: 100  printAfter:200  maxRuns:200  initConns:150
TraffDesc: connId=0  TrafficType= 1nb of states=2
rates:       1      2
probs:       0.6    0.4

initial free CBR bandwidth 290
initial free nb of Conns 207

'usable' after dropping 200 packs with 100 connections:
free CBR: 363          lostCBR/linkRate: -0.146
free conns: 207        lostConns/possibleConns: 0

=====================================================

linkRate: 500  Rbin: 500  addBins: 100  cellLossProb: 1e-05  overBookingRa-
tio: 1000
packSize: 100  printAfter:200  maxRuns:200  initConns:150
TraffDesc: connId=0  TrafficType= 1nb of states=2
rates:       1      2
probs:       0.6    0.4

initial free CBR bandwidth 290
initial free nb of Conns 207

'usable' after dropping 200 packs with 100 connections:
free CBR: 313          lostCBR/linkRate: -0.046
free conns: 207        lostConns/possibleConns: 0

=====================================================

linkRate: 500  Rbin: 500  addBins: 500  cellLossProb: 1e-05  overBookingRa-
tio: 1000
packSize: 100  printAfter:200  maxRuns:200  initConns:150
TraffDesc: connId=0  TrafficType= 1nb of states=2
rates:       1      2
probs:       0.6    0.4

initial free CBR bandwidth 290
initial free nb of Conns 207

'usable' after dropping 200 packs with 100 connections:
free CBR: 290          lostCBR/linkRate: 0
free conns: 207        lostConns/possibleConns: 0

=====================================================
```

```
linkRate: 500  Rbin: 500  addBins: 0  cellLossProb: 1e-05  overBookingRatio:
1000
packSize: 100  printAfter:50  maxRuns:50  initConns:80
TraffDesc: connId=0  TrafficType= 1nb of states=2
rates:       0       100
probs:       0.999   0.001

initial free CBR bandwidth 228
initial free nb of Conns 637


ASSERTION  (cacVal==TRUE)  FAILED!  FILE:  LostBandwidthErrorInter-
preter.C; LINE: 158
linkRate: 500  Rbin: 500  addBins: 50  cellLossProb: 1e-05  overBookingRatio:
1000
packSize: 100  printAfter:50  maxRuns:50  initConns:80
TraffDesc: connId=0  TrafficType= 1nb of states=2
rates:       0       100
probs:       0.999   0.001

initial free CBR bandwidth 228
initial free nb of Conns 379

'usable' after dropping 50 packs with 100 connections:
free CBR: 200          lostCBR/linkRate: 0.056
free conns: 379        lostConns/possibleConns: 0

=====================================================

linkRate: 500  Rbin: 500  addBins: 100  cellLossProb: 1e-05  overBookingRa-
tio: 1000
packSize: 100  printAfter:50  maxRuns:50  initConns:80
TraffDesc: connId=0  TrafficType= 1nb of states=2
rates:       0       100
probs:       0.999   0.001

initial free CBR bandwidth 228
initial free nb of Conns 379

'usable' after dropping 50 packs with 100 connections:
free CBR: 200          lostCBR/linkRate: 0.056
free conns: 379        lostConns/possibleConns: 0

=====================================================

linkRate: 500  Rbin: 500  addBins: 500  cellLossProb: 1e-05  overBookingRa-
tio: 1000
packSize: 100  printAfter:50  maxRuns:50  initConns:80
TraffDesc: connId=0  TrafficType= 1nb of states=2
rates:       0       100
probs:       0.999   0.001

initial free CBR bandwidth 228
initial free nb of Conns 392

'usable' after dropping 50 packs with 100 connections:
free CBR: 200          lostCBR/linkRate: 0.056
free conns: 392        lostConns/possibleConns: 0

=====================================================
```