A Stochastic Analysis of Network Traffic Based on Histogram Workload Modelling

Enrique Hernández-Orallo *, Joan Vila-Carbó

Departamento de Informática de Sistemas y Computadores. Universidad Politécnica de Valencia. Camino de Vera, S/N. Valencia, Spain

Abstract

Finding a simple network traffic model described by a reasonable number of parameters that enables powerful analysis methods and provides accurate results has been a challenging problem during the last decades. This paper proposes a discrete statistical description of network traffic, known as histograms, and a stochastic analysis method based on the concept of statistical convolution. Unlike other histogram based methods, it does not require approximating traffic to a Poisson distribution (or any other distribution) nor solving queueing models. The method shows that the buffer length problem is a stochastic process that converges to a steady state probability when the average traffic arrival rate is less or equal than the output service rate, although this arrival rate can be higher during transient overloads. The paper shows that *workload isolation* is a desirable property of traffic scheduling algorithms that highly eases the analysis and makes it algorithm independent. A more accurate method based on calculating interferences among workloads is sketched for the case of the GPS algorithm. The proposed method has been extensively evaluated using synthetic and real traffic traces. These evaluations analyse the influence of some factors on accuracy and show that the method improves the results of queueing models. Applications of this model are very wide: analysis and prediction of QoS parameters, network dimensioning and provisioning, traffic admission control, etc.

Key words: Traffic Modelling, Networks QoS, Discrete Statistical Traffic Modelling, Stochastic Analysis.

* Corresponding author.
Email addresses: ehernandez@disca.upv.es (Enrique Hernández-Orallo), jvila@disca.upv.es (Joan Vila-Carbó).

[1] This work was developed under grant of the Spanish Government CICYT

1 Introduction

Providing Quality-of-Service (QoS) requirements is a key issue in real-time networking applications, such as video and audio conferencing, video on demand, etc. since they pose stringent requirements on delivering delays and packet-loss ratios. Guaranteeing performance on this kind of applications usually requires some network resource allocation, like bandwidth and buffers. Accurately evaluating these resources is one of the main challenges. This problem has been analysed in the literature using two main approaches [1]: *deterministic* and *statistical* techniques.

Deterministic approaches are based on simplistic workload characterisations and worst-case analyses. QoS requirements are specified in absolute terms and performance guarantees are mostly based on resource reservation [2]. A well known example of deterministic bounds are the delay and buffer limits of Parekh and Gallager [3, 4] developed under the Generalized Processor Sharing (GPS) theory and a leaky-bucket traffic characterisation [5, 6]. Much work has been done since then using the deterministic model [7–9]. The problem is that it leads to resource overallocation and network infrautilisation, especially with variable and bursty traffic.

Some approaches for improving resource utilisation of deterministic models are based on reducing traffic variability. This includes considering traffic aggregates rather than individual flows [10, 11], or traffic smoothing [12, 13]. However, keeping a high network utilisation mainly implies replacing absolute limits by probabilistic ones. In other words, allowing some small percentage of deadline violations and packet loss ratios. Packet losses are due either to buffer overflow or to explicitly packet *draining* [14]. This is done for the purpose of policing [15] or to avoid further congesting the network with excessively delayed packets.

Statistical approaches are related to QoS guarantees. They have been also analyzed in the literature, although to a lesser extent than the deterministic model [16]. This approach was first introduced by Ferrari and Verma [17], who proposed a channel-establishment condition for statistical performance guarantees using the Earliest-Due-Date (EDD) packet scheduling policy. Zhang and Knightly [18] provided connection statistical performance guarantees under Rate-Controlled Static-Priority (RCSP) scheduling using a traffic model with a bounded number of packets in several time intervals and traffic shaping in intermediate nodes to reduce traffic variability. Zhang et al. [19] also derived a statistical bound on the end-to-end delay using GPS scheduling and the Exponentially-Bounded Burstiness (EBB) model.

Closely related to the analysis methods of previous approaches is *traffic mod-*

elling. As more accurate is traffic description, better results can be obtained, but usually the analysis method becomes more complex. There has been a considerable amount of work on traffic characterisation in the literature [20]:

In the *deterministic approach*, traffic is described by providing just a few parameters, as the TSpec of RSVP [21], based on the token-bucket parameters. This description is very coarse and does not capture the notion of traffic variability. Some improvements to this model are the ATM traffic description, that provides a bound on the traffic bursts, or the concept of *empirical envelope* [22] which bounds the accumulated traffic in a given interval and its approximation by a discrete number of points called *envelope points* [23]. However, all these descriptions are still pessimistic and usually provide a bound on a (highly improbable) worst-case delay.

Some works based on the *statistical* model concentrate on matching statistical parameters of models of video sources which describe traffic real traffic fairly well [24–26]. However, there is no consensus on these models [20] that usually degenerate to a readily analysable Gaussian model on very large networks.

An interesting approximation for traffic characterisation are *histogram* based models which describe traffic as a discrete statistical distribution. They extend *deterministic* models, that usually describe traffic with one or two classes (average and peak rate), with a discrete number of classes, quantifying their probability. The model, known as the *Histogram Model* [27] or the *Generalized Histogram Model* [28], was introduced by Skelly et al. [27] to predict buffer occupancy, distributions and loss rate for multiplexed streams. A modified version was also used by Kweon and Shin [29] to propose an implementation of statistical real-time channels in ATM using the Traffic Controlled Rate-Monotonic Priority Scheduling (TCRM), where they introduce a method for determining the transmission capacity of multiplexed channels to statistically guarantee a cell loss-ratio. These works use an analysis method based on a M/D/1/N queueing system. The number of ATM cells generated during a frame period is approximated to a Poisson distribution with a given rate λ. For a given video sequence, λ is modelled as a histogram. The buffer occupancy is calculated by solving the M/D/1/N system as a function of λ and then weighting the solutions according to the histogram probabilities. This methods yields good results with a reduced number of cells in the buffer, but the inaccuracy increases with the number of cells.

This paper shows that the buffer length problem is a stochastic process that converges to a steady state solution under certain conditions. The proposed method does not require approximating traffic to a Poisson distribution nor solving queueing models. The solution is easily obtained assuming a constant bandwidth for each traffic stream (*workload isolation*). This is also implicitly assumed in all previous works. But the paper sketches how the method can

be extended for considering workload interferences under a particular packet scheduler. This is interesting for a real-time analysis of the system.

The proposed method has been extensively evaluated using synthetic and real traffic traces. These evaluations analyse the influence of the number of histogram classes on precision, showing that 10 classes is enough to obtain good results, although it is necessary to use a technique called *overclassing* for obtaining accurate results. The evaluations also show that the method improves the results of queueing models with a simpler calculation method.

2 Histogram workload characterisation

The basic idea behind this work is improving the results of deterministic analyses of network calculus and real-time systems by using a probabilistic workload description that describes more accurately the workload variability than deterministic or simplistic descriptions and provides a powerful analysis method.

Network workloads will be characterised by the number of transmission units produced by a traffic source during a pre-established time period called the *sampling period*[2]. The proposed model characterises variable workloads not as a function of time, but as a discrete statistical distribution (see Fig. 1b). Choosing the sampling period is an important issue. In the case of a periodic workload, the sampling period usually matches the "natural" period. For example, in video transmission it is usually the video frame period. The bit rate during a sampling period is, in general, variable. The proposed method assumes that traffic arrives at uniform rate. In other words, if we have N bits in a sampling period, the inter-arrival distribution is deterministic with value $1/N$. Some authors, like [27] or [29], consider that the arrival process during a sampling period is approximately Poisson with rate λ. This allows to analyse the system by solving a $M/D/1/N$ queueing system as a function of λ. The method proposed in this paper is not based on the solution to the Poisson process but on defining histogram operators and iterative algorithms to solve the steady state of a stochastic process.

2.1 Histogram basics

Statistical variables are, in general, defined over a continuous range of values. A *grouped probability distribution* (gpd) is the probability function of a random

[2] For convenience we use the bit as the base unit. There is no variation in the precision of the results using another unit (bytes, packets)

variable defined over intervals: the probabilities of values in an interval are grouped together. Intervals will be also referred as *classes*. All intervals are the same width and class 0 should always represent the probability of workload zero or close to zero. Table 1a shows an example gpd.

[Fig. 1 about here.]

Classes in a gpd will be characterised by the following attributes: class number i, interval lower limit x_i^-, interval upper limit x_i^+, interval midpoint x_i, interval probability $p_X(i)$ and cumulative probability $p_X^+(i) = \sum_0^i p_X(i)$.

The example of table 1a corresponds to a workload that has been analysed using a sampling period of $T_X = 0.1\ s$. The range of the transmission units measured during this sampling period is in $[0, 120[\ kb$. This range is divided into $n = 6$ intervals or classes so the *interval length* is $l_X = 20\ kb$. Class 0 corresponds to interval $[0, 20[\ kb$ whose midpoint is $x_0 = 10\ kb$. The probability that the traffic source produces a number of transmission units in this interval is $p_X(0) = 0$. Similarly, class 1 corresponds to interval $[20, 40[\ kb$ with probability $p_X(1) = 0.1$, and so on.

A *histogram* is a form of a bar graph representation of a gpd. Fig. 1b shows the histogram of table 1a. The x-axis of this graph will be either the class number i or its midpoint x_i.

A gpd X is usually managed through two arrays of values: the *array of interval midpoints*, denoted as $\dot{X}$, and the *array of interval probabilities*, also referred as *probability mass function* (pmf), denoted as $\mathcal{X}$:

$$X \begin{cases} \dot{X} = [x_i : i = 0 \dots n-1] \\ \mathcal{X} = [p_X(i) : i = 0 \dots n-1] \end{cases} \tag{1}$$

Formally: $p_X(i) \equiv P(x_i^- \leq X < x_i^+)$.

In the example of Fig. 1a:

$$X \begin{cases} \dot{X} = [10, 30, 50, 70, 90, 110] \\ \mathcal{X} = [0, 0.1, 0.4, 0.2, 0.15, 0.15] \end{cases}$$

Its important to note that a gpd X is usually defined over a domain of real numbers $\dot{X}$ while its corresponding pmf $\mathcal{X}$ is usually defined over a domain of integers $i = 0 \dots n-1$ representing the class numbers [3]. Most of the calculus

[3] Although traffic workload is in the natural domain, we generalize the gpd definition using a real domain

and algorithms performed on a gpd X only involve its pmf $\mathcal{X}$. This means that they depend on the class number rather than $\dot{X}$. However, when several gpd's are involved in a calculation (for example in a convolution), the correctness of the results requires to use the same sampling period (T_X) and the same *interval length* (l_X).

The correspondence between some value x in the domain of X and its class number $\hat{x}$ is given by the following equation:

$$\hat{x} = class_X(x) = \left\lfloor \frac{x}{l_X} \right\rfloor \tag{2}$$

For example, given $x = 55$, its corresponding class can be obtained as: $\hat{x} = class_X(55) = \left\lfloor \frac{55}{20} \right\rfloor = 2$.

Some important operators on random variables that will be used throughout the paper are introduced next.

- The *mean value* (or expectation) of gpd X is defined as: $E[X] = \sum_0^{n-1} p_X(i) \cdot x_i$. Analogously, the mean value of pmf $\mathcal{X}$ is defined as $E[\mathcal{X}] = \sum_0^{n-1} p_X(i) \cdot i$. In the previous example, $E[X] = 67\ kb$ and $E[\mathcal{X}] = 2.85$. Recall that this value is referred to the sampling period of $T_X = 0.1\ s$.
- The *maximum midpoint* of X is defined as $M[X] = \max(x_i : p_X(i) > 0)$ and $M[\mathcal{X}] = n - 1$), In the previous example, $M[X] = 110\ kb$ and $M[\mathcal{X}] = 5$.
- The *scalar multiplication* of X by a constant c is a new random variable $Y = c \cdot X$ where $y_i = c \cdot x_i$ and $p_Y(i) = p_X(i)$ for $i = 0 \ldots n$. Note that variable Y has the same pmf than X, that is, $\mathcal{X} = \mathcal{Y}$. Multiplying by a scalar only affects the interval length: $l_Y = c \cdot l_X$.
- The *convolution* of two random variables X and Y, denoted as $X \otimes Y$, is the statistical equivalent to the notion of summing two deterministic variables. The results is a new variable Z whose domain is in the range [0, M[X] +M[Y]] and with an associated pmf $\mathcal{Z} = [p_Z(i) : i = 0 \ldots n + m - 2]$ where n and m are the number of intervals of X and Y respectively, and $p_Z(i) = \sum_{k=0}^{i} p_X(i-k) \cdot p_Y(k)$. The convolution is only defined for variables with the same sampling period and the same interval length. Convolutioning variables with different interval lengths require previously adjusting them to the same length using the transformations defined next. The convolution has the following interesting property: $E[\mathcal{X} \otimes \mathcal{Y}] = E[\mathcal{X}] + E[\mathcal{Y}]$.
- *Increasing the zoom factor* of variable X by k means applying a transformation $\Delta_k^+ : X \longrightarrow Y$ to increase the number of classes of X in a uniform way by uniformly splitting each interval of X into k intervals of Y. The interval length of Y is $l_Y = \frac{l_X}{k}$ and its pmf $\mathcal{Y} = [p_Y(j), i = 0 \ldots m-1]$ has a larger number of classes $m = k \times n$. Probabilities $p_Y(j)$ are obtained by linear interpolation of values $p_X(i)$ in the range $i : 0 \ldots n-1$ to the

range $j : 0 \dots m-1$.

- Analogously, *decreasing the zoom factor* of X by k means applying a transformation $\Delta_k^- : X \longrightarrow Y$ to reduce the number of classes of X by merging k consecutive intervals of X (starting at interval 0) into one interval of Y. The interval length of Y is $l_Y = l_X \times k$ and its pmf $\mathcal{Y} = [p_Y(j), i = 0 \dots m-1]$ has a shorter number of classes $m = \lfloor \frac{n}{k} \rfloor$. Probabilities $p_Y(j)$ are obtained by summing the probabilities of the k intervals of X that correspond to interval j of Y.

2.2 Histogram number of classes and accuracy

One of the most critical problems of the histogram method is accuracy. The key issue is to determine the number of classes of a histogram. This is in general a trade off between representation economy and precision: with too many intervals, the representation will be cumbersome and histogram processing expensive, since the complexity of algorithms mostly depends on the number of classes but, on the other hand, too few intervals may cause loosing information about the distribution and masking trends in data.

Another important problem is that histogram processing with a low number of classes results in important precision errors. It is paradoxical that these errors occur even if that low number of classes is enough to properly describe a given workload without loosing much information. The reason for those inaccuracies seems to be the effect of the low number of classes when using iterative algorithms. The solution proposed in this paper consists in *overclassing* the histogram which is a transformation for "artificially" increasing the number of classes by splitting each interval into m intervals with the same probabilities. That implies increasing the *zoom factor of the distribution* by m.

3 The histogram method

This section introduces the problem formulation and its resolution using histograms. This resolution includes the mathematical foundation and the practical algorithms.

3.1 Problem statement

The problem addressed in this paper is an end-to-end transmission system. The analysis starts by considering a single node of this system. Such a node is a

router processor that multiplexes a set of traffic streams (see Fig.2 (a)). Input traffic is supplied through buffers of finite or infinite capacity. These buffers accumulate pending traffic that cannot be transmitted over a sampling period. The system will be said to be *stable* if the pending traffic converges to a finite value. The goals of the analysis are studying system stability, transmission delay and some other QoS parameters as data loss rates.

Traffic generated at a data source is, in general, time dependent. According to this, the traffic arrival rates on a node will be also a set of time dependent functions, denoted as: $\{A_k(t), k : 0 \dots m\}$, where $m+1$ is the number of traffic sources. We will start our discussion by expressing $A_k(t)$ as time dependent functions and they will be later transformed into discrete statistical variables.

The service time for a particular traffic source, denoted as $S_k(t)$ will be defined as the bandwidth allocated to stream i at time t. In the general case (Fig. 2 (a)), the service time for source k is a function of the set of all traffic sources and the packet scheduling algorithm F. This can be expressed as: $S_k(t) = F(\{A_k(t), k : 0 \dots m-1\})$.

Expressing the service time as a function of all traffic sources leads to a difficult problem. However, the problem can be decomposed into m independent problems, as illustrated in Fig.2 (b), by requiring the hypothesis of *strong workload isolation*: every data source is assigned a constant bandwidth that is time and workload independent: $S_i(t) = R_i, \forall t$.

This is an ideal assumption because it only occurs in the fluid model of GPS [3]. Some other algorithms exhibit a weaker form of this property that will be referred simply as *workload isolation*: the average service time is constant. This is true for GPS based algorithms with full bandwidth utilisation. The general case where the service time is a complex function of all traffic sources will be analysed using the, so called, *interference method*. This kind of analysis is usual in real-time systems and has been used in [30] for an statistical study of the RM (Rate Monotonic) algorithm and in [31] for the case of the GPS (General Processor Sharing) algorithm. This paper concentrates on the *strong workload isolation* hypothesis. The interference method will be shortly introduced in section 6, but it is not fully developed in this paper.

It is worth to say that in current networks *strong workload isolation* is usually achieved in a simple way: all traffic sources with the same output link are aggregated into one aggregated traffic. This aggregated traffic uses all the bandwidth R of the output link and has its own buffer queue Q. Therefore, all the work presented here can be applied to the aggregate traffic flows.

[Fig. 2 about here.]

3.2 Method foundation

This section develops a method for computing the buffer length and system stability with the hypothesis of *strong workload isolation*, that is $S(t) = R$ [4]. This scenario assumes a single traffic source served at a constant rate. The traffic arrival rate $A(t)$, will be expressed, in principle, as a time dependent function. Assuming a buffer of finite capacity l, the queue or buffer length $Q(t)$ at a given time t can be expressed as:

$$Q(t) = \int_0^t \phi_0^l(A(t) - S(t))\ dt \tag{3}$$

where, $S(t)$ is the node service rate and operator ϕ limits buffer lengths so they cannot be negative and cannot overflow value l either. This operator is defined as follows:

$$\phi_a^b(x) = \begin{cases} 0, \text{ for } x < a \\ x, \text{ for } a \leq x < b + a \\ b, \text{ for } x \geq b + a \end{cases} \tag{4}$$

This expression can be rewritten as a recurrence equation (known as Lindley's equation [32]) assuming a discrete time space $\tau = t_0, t_1, t_2, \ldots$ where $t_k = k \times T$ is a multiple of the sampling period T_X. This way, functions $A(t), S(t)$ and $Q(t)$ can be replaced by discrete time functions:

$$Q(k) = \phi_0^l(Q(k-1) + A(k) - S(k))$$

In this expression $Q(k)$ is the cumulative number of bits that the traffic source puts into the buffer during the k-th sampling period. $A(k)$ is the number of bits that arrives at period k. Analogously the service rate $S(k)$ is the number of cumulative bits that the router removes from the buffer during the same sampling period. The service rate can be expressed as a constant r, that is the output rate R multiplied by the period T_X ($r = R \times T_X$):

$$Q(k) = \phi_0^l(Q(k-1) + A(k) - r) = \phi_r^l(Q(k-1) + A(k))$$

The foundation of the histogram method basically consists of suppressing the time dependence of $A(k)$ in the previous expression and replacing it by a discrete random variable with pmf $\mathcal{A} = [p_A(k), k = 0 \ldots n]$. This way, previous equation can be expressed transformed into a statistical equation:

$$\mathcal{Q}(k) = \Phi_{\hat{r}}^{\hat{l}}(\mathcal{Q}(k-1) \otimes \mathcal{A}) \tag{5}$$

[4] From now on, for notational convenience, we drop the subscript i

where $\mathcal{Q}(k)$ is now a stochastic process, $\hat{r} = class_X(r)$ and $\hat{l} = class_X(l)$, operator $\otimes$ stands for the standard statistical convolution and the *bound operator* $\Phi_a^b()$ is defined as the statistical generalisation of the previously defined $\phi_a^b()$ operator:

$$\Phi_a^b(\mathcal{X}) = \Phi_a^b([p_X(0), p_X(1) \dots p_X(n)]) =$$
$$= \Big[\sum_{i=0}^{a} p_X(i), p_X(a+1), p_X(a+2), \dots, p_X(b+a-1), \sum_{i=b+a}^{n-1} p_X(i)\Big] \qquad (6)$$

Notation $\Phi_a(\mathcal{X})$ will be used as an equivalent for $\Phi_a^\infty(\mathcal{X})$.

As an example of how this operator performs, given $\mathcal{X} = [0, 0.1, 0.4, 0.2, 0.15, 0.15]$, then $\Phi_2(\mathcal{X}) = [0+0.1+0.4, 0.2, 0.15, 0.15] = [0.5, 0.2, 0.15, 0.15]$ and $\Phi_2^2(\mathcal{X}) = [0 + 0.1 + 0.4, 0.2, 0.15 + 0.15] = [0.5, 0.2, 0.3]$.

Note that the transformation to the histogram class domain produces discretization errors. The effect of this transformation will be studied in detail in the evaluation experiments.

As previously said, $\mathcal{Q}(k)$ has become a discrete time stochastic process. Moreover, this stochastic process is shown to be Markovian in appendix A.1. The evolution in time of this stochastic process can be analysed in terms of the mean value of $\mathcal{A}$.

When $M[\mathcal{A}] \leq \hat{r}$ ($r = R \times T_X$), then the buffer length is zero because it is easy to prove, from its definition, that $\Phi_{\hat{r}}(\mathcal{A})$ is zero in this case.

The case when $M[\mathcal{A}] > \hat{r}$ is the most interesting one because, unlike worst-case kind of analyses, statistical analyses allow arrival rates to exceed occasionally processor capacity during transitory overloads and still have a stable system depending on $E[\mathcal{A}]$. Two subcases must be considered: infinite and finite buffer.

In the infinite buffer case, the system converges to a steady-state pmf iff $E[\mathcal{A}] \leq \hat{r}$. With a finite buffer, the process always converges because it is always bounded by operator $\Phi_a^b()$. Although intuitive, this conditions are formally proven in appendixes A.2 and A.3 respectively.

System evolution for the above considered cases is shown in Figs. 3, 4 and 5. The example workload $\mathcal{A} = [0, 0.1, 0.4, 0.2, 0.15, 0.15]$ of Fig. 1b has $E[\mathcal{A}] = 2.85$ and $M[\mathcal{A}] = 5$. Figure 3 shows that the stochastic process converges to a steady state solution for an infinite buffer and a constant service rate $\hat{r} = 3$, since $E[\mathcal{A}] \leq \hat{r}$. Figure 4 shows the evolution for the case of an infinite buffer and $\hat{r} = 2$. With $E[\mathcal{A}] > \hat{r}$ the system is unstable and the pmf of the buffer length is shifted to the right in each iteration. Figure 5 shows the situation

with $\hat{r} = 2$ and a finite buffer $\hat{b} = 30$. It can be seen that the probability of buffer full tends to 1.

[Fig. 3 about here.]

3.3 Algorithm for buffer length calculation

This subsection describes in detail the algorithm to calculate the buffer length probability mass function and illustrates it with examples.

According to previous subsection the buffer length problem is, in general, a stochastic process whose steady state solution can be calculated through an iterative process. The algorithm for calculating the steady state probability mass function of the buffer length is shown in Fig.6 and will be referenced as the HBSP (Histogram Based Stochastic Process) algorithm.

[Fig. 4 about here.]

The explanation to this algorithm is provided through the example of Fig. 1a. Recall that this workload has been obtained with a sampling period of $T_X = 0.1\ s$ and it has 6 intervals of length $l_A = 20kb$ whose midpoints are $\dot{A} = [10, 30, 50, 70, 90, 110]$ expressed in kb. The mean and maximum values of A are $E[A] = 57\ kb$, $M[A] = 120\ kb$. The corresponding histogram is shown in Fig. 7a and the pmf is $\mathcal{A} = [0, 0.1, 0.4, 0.2, 0.15, 0.15]$.

The algorithm is described using an output rate $R = 600kb/s$ (that is, a service rate of $r = 60\ kb$ per sampling period) and a bounded buffer length of 100 kb. In terms of the pmf, those values correspond to $\hat{r} = class_X(r) = 3$ and $\hat{b} = class_X(b) = 5$.

[Fig. 5 about here.]

In the first iteration of the HBSP algorithm, the buffer histogram $\mathcal{Q}$ is obtained by summing classes 0..3 of $\mathcal{A}$ (this workload is transmitted without queueing) and shifting it to the left (Fig. 7d):

$$\mathcal{Q}(1) = \Phi_3(\mathcal{A}) = [0.7, 0.15, 0.15]$$

The probability that the buffer is 1 and 2 is 0.15 in each case. Since the buffer length is $\hat{b} = 5$ there is no probability of exceeding buffer capacity after the first iteration but, in general, the bound operator establishes an upper limit on the queued workload due to finite buffer length:

$$\mathcal{Q}(1) = \Phi_3^5(\mathcal{A}) = [0.7, 0.15, 0.15]$$

In the second iteration (next sampling period), the buffer already stores a pending workload of $\mathcal{Q}(1)$ and, in addition, a new workload $\mathcal{A}$ arrives. The *cumulative workload* histogram in the buffer after this iteration, is the convolution of the previous histograms (Fig. 7b):

$$\mathcal{I}(2) = \mathcal{Q}(1) \otimes \mathcal{A} =$$
$$= [0, 0.0700, 0.2950, 0.2150, 0.1950, 0.1575, 0.0450, 0.0225]$$

Now the effect of the finite buffer (5 classes) will produce a loss in the cases where there is a probability that the buffer length is greater than 5. For example, for a 6 units length, 5 units are stored into the buffer and the other one is discarded, so this probability has to be added to the probability class 5. Analogously, in the case of 7 units, 2 are accumulated and 5 are discarded. According to this the result of the second iteration is (Fig. 7e):

$$\mathcal{Q}(2) = \Phi_3^5(\mathcal{I}(2)) = [0.5800, 0.1950, 0.2250]$$

For a precision of $\varepsilon = 1 \times 10^{-4}$ the algorithm converges after 41 iterations to the following solution (see Figs. 7c and 7f):

$$\begin{aligned}\mathcal{Q} &= [\,0.3275, 0.1625, 0.1699, 0.1291, 0.1077, 0.1033] \\ \mathcal{I} &= [\,0, 0.0327, 0.1472, 0.1475, 0.1625, 0.1699, 0.1291, 0.1077, 0.0562, \\ &\quad\; 0.0317, 0.0155]\end{aligned}$$

The values of $\dot{Q}$ and $\dot{I}$ can be calculated taking into account that the interval length is 20 kb:

$$\begin{aligned}\dot{Q} &= [10, 30, 50, 70, 90, 110]\ kb \\ \dot{I} &= [10, 30, 50, 70, 90, 110, 130, 150, 170, 190, 210]\ kb\end{aligned}$$

3.4 QoS parameters

Some of the most important performance parameters of a router are delay and loss ratio. This section shows how to obtain these parameters using the histogram method.

The *router delay* D is the time between message arrival at that station and message departure from the station. It is the sum of the *queuing delay* U and

the *transmission delay* T. This can be expressed in statistical terms as:

$$\mathcal{D} = \mathcal{U} \otimes \mathcal{T} \tag{7}$$

The *queueing delay* is the time spent by the message waiting for previous buffered messages to be transmitted. In the case of a router with a output rate of R, and a buffer length characterised by a gpd Q the queueing delay is proportional to Q, so it has the same pmf.

$$U = \frac{1}{R} \cdot Q \tag{8}$$

In statistical terms, multiplying Q by a scalar $\frac{1}{R}$ (*scalar multiplication*) only affects its interval length (and thus $\dot{U}$). Then the interval length of U is $l_U = l_Q/R$, expressed in seconds.

The *transmission delay* is the time spent by the network interface in processing the message and it is closely related to the transmission speed. Assuming d is the delay for any transmission unit of size lesser than the MTU (Maximum Transmission Unit) and using the same interval length of U we obtain the class interval as $\hat{d} = class_T(d)$. In statistical terms, T is a deterministic distribution of the form $\mathcal{T} = [t_0, \ldots, t_{\hat{d}}]$ with $t_i = 0$ for $i \leq \hat{d}$ and $t_i = 1$ for $i = \hat{d}$ and $\dot{T} = [l_U, 2 \cdot l_U, \ldots \hat{d} \cdot l_U]$. Then, $\mathcal{D} = \mathcal{U} \otimes \mathcal{T}$ can be calculated convolutioning $\mathcal{Q}$ and $\mathcal{T}$:

$$\mathcal{D} = \mathcal{Q} \otimes [0, \ldots, 0, 1_k] \tag{9}$$

As an example, consider the stationary buffer length of the example of previous section, obtained with a service rate $R = 600\ kb/s$ and assume that the transmission delay is $d = 0.1\ s$. First, we obtain the interval length of U as $l_U = 20/600 = 0.0333s$. The class of d is $\hat{d} = class_U(0.1) = 3$. The router delay is calculated as:

$$\begin{aligned}\mathcal{D} &= \mathcal{Q} \otimes \mathcal{T} = \\ &= [0.3275, 0.1625, 0.1699, 0.1291, 0.1077, 0.1033] \otimes [0, 0, 0, 1] = \\ &= [0, 0, 0, 0.3275, 0.1625, 0.1699, 0.1291, 0.1077, 0.1033]\end{aligned}$$

and the array of midpoints is:

$$\dot{D} = [\,0.033, 0.066, 0.1, 0.133, 0.166, 0.2, 0.233, 0.266, 0.3]$$

Another QoS parameter is the *overflow probability*, that is usually used as an approximation of the loss ratio, due that is most amenable to mathematical analysis [20]. The overflow probability is defined as the probability that the

number of workload units in the buffer exceeds a given limit. This probability can be easily obtained using the buffer histogram with infinite buffer.

$$P(Q \geq t) = \sum_{i=t}^{n-1} x_i \tag{10}$$

The *loss ratio* is defined as the ratio between the number of lost transmission bits and the number of arriving transmission bits:

$$P_{loss} = \frac{E[\mathcal{C}]}{E[\mathcal{A}]} \tag{11}$$

where the lost transmission bits can be easily calculated using the bound operator:

$$\mathcal{C} = \Phi_{\hat{r}+\hat{b}}(\mathcal{I}) \tag{12}$$

with $\mathcal{I}$ being the cumulative workload pmf, $\hat{r}$ the service rate class and $\hat{b}$ the buffer length class.

The calculus of the loss ratio can be clearly understood using the example of previous section. Consider the stationary cumulative workload pmf ($\mathcal{I}$) (see Fig.7c). With a service rate class $\hat{r} = 3$ and a buffer length class $\hat{b} = 5$, from a workload of 10 units, 3 units are sent, 2 units are stored in the buffer and 2 units are lost. Therefore, the *loss pmf* ($\mathcal{C}$) can be obtained by shifting (with accumulation) $\hat{r} + \hat{b} = 3 + 5 = 8$ positions to the right. Using the bound operator:

$$\mathcal{C} = \Phi_8(\mathcal{I}) = [0.9528, 0.0317, 0.0155]$$

Histogram $\mathcal{C}$ reflects that 0.9528 is the probability of no loss, 0.0317 is the probability of 1 unit loss and 0.0155 is the probability of 2 units loss. According to this, the probability of at least 1 unit loss is the following weighted sum: $E[\mathcal{C}] = 0.0317 * 1 + 0.0155 * 2 = 0.0627$. Then, the loss ratio is the proportion between the lost units and the arrival workload:

$$P_{loss} = \frac{E[\mathcal{C}]}{E[\mathcal{A}]} = \frac{0.0627}{2.85} = 2.2\% \tag{13}$$

4 Evaluation

This section presents an evaluation aimed to evaluate and to validate the HBSP method. The evaluation comprises experiments with synthetic traffic, experiments with real traffic traces and comparisons with other methods. One of the key aspects of the experiments is to evaluate the accuracy of the HBSP

method. This is achieved by comparing analytical results of the HBSP with results obtained through simulation. All the experiments of the HBSP algorithm use an accuracy factor of $\varepsilon = 1 \times 10^{-6}$.

4.1 Synthetic Workload Experiments

The goal of these experiments is to evaluate the buffer length and loss ratio using the HBSP algorithm and comparing results with the ones obtained through simulation. Two synthetic arrival workloads are used in this subsection: the first one uses no grouping of classes, so no discretization errors can occur. The second one corresponds to a grouped probability distribution and it is aimed to evaluate discretizations errors.

Simulations were performed using a synthetic traffic trace generated from a given arrival workload pmf $\mathcal{A}$. This means that the synthetic trace was generated to have a pmf of $\mathcal{A}$. The simulation is performed assuming that generated traffic arrives at uniform rate. If a trace consists of n samples, the simulation obtains the buffer length Q_i and loss units for each sample. The final result is a vector of buffer lengths $Q = \{Q_i; \quad i = 0..n\}$ whose pmf ($\mathcal{Q}_S$) can be easily obtained, and the final loss ratio P_{loss}.

The first experiment was done with an arrival workload $\mathcal{A}$ =[0.0, 0.3, 0.1, 0.3, 0.1, 0.1, 0.05, 0.05] (see Fig.8a). The interval length was assumed to be 1 unit. This means that the traffic range of $\mathcal{A}$ is from 0 to 7 units with a mean value $E[\mathcal{A}] = 2.95$. A synthetic traffic with 20,000 samples was generated using $\mathcal{A}$ (see Fig.8b). The output rate of the node was set to $R = 4$ units/sec ($\hat{r} = r = 4$) and an infinite buffer was assumed. Fig.8c shows the histogram obtained using the algorithm and by simulation. The experiment was repeated 100 times with different traffics in order to obtain a mean difference between both histograms. The average normalized difference between the histogram generated buffer and the simulated histogram buffer ($|\mathcal{Q} - \mathcal{Q}_S|$) was 0.003843$\pm$ 0.000241 (that is, with a 95% confidence interval of [0.003601,0.004084]).

[Fig. 6 about here.]

The second experiment used the same synthetic traffic of previous experiment. The output rate was also set to $R = 4$ units/sec ($\hat{r} = r = 4$), but the buffer length was set to $b = 6$ units ($\hat{b} = 6$). Fig.8d shows the histograms obtained using the algorithm and by simulation. As in the previous experiment, the differences are negligible. The obtained average difference for 100 experiments was 0.003350 ± 0.000206. The estimated loss ratio was P_{loss}= 0.002725 and the simulated one was 0.002739 ± 0.000074 (very close to the estimated loss ratio).

The goal of the third and fourth experiments is to evaluate the effect of discretization, that is, reducing the number of classes by grouping them by a given factor. The sample period for these experiments was set to $T = 0.1$ sec. The workload was grouped in intervals of length 5 units to define the workload histogram $\mathcal{A} = [0.1, 0.3, 0.2, 0.1, 0.1, 0.1, 0.05, 0.02, 0.02, 0.01]$. This corresponds to a grouped distribution whose midpoints are $\dot{A} = [2.5, 7.5, 12.5, 17.5, 22.5, 27.5, 32.5, 37.5, 42.5, 47.5]$ (see Fig.9a) with $E[\mathcal{A}] = 2.59$ and $E[A] = 15.45$. A synthetic traffic of 20,000 samples was generated from the grouped distribution, so it does not model accurately the original traffic. This traffic ranges from 0 to 49 units and the first 500 samples are shown in Fig.9b. As stated before, the goal of these experiments is to evaluate the discretization errors.

[Fig. 7 about here.]

For the third experiment the output rate was set to $R = 300$ units/sec ($r = R \times T = 30$ units and $\hat{r} = 6$) and an infinite buffer was considered. Fig.9c shows the histogram obtained using the HBSP algorithm ($\mathcal{Q}$) and simulating the buffer load ($\mathcal{Q}_S$) (lines 'Simulation' and 'Histogram') [5]. Graphically, the difference between the histograms is low, but non negligible. After repeating 100 times the experiments the average difference between the HBSP and simulation is 0.030041 ± 0.000187.

In the fourth experiment the output rate was set to $R = 200$ units/sec ($r = R \times T = 20$ units and $\hat{r} = 4$) and the buffer length was set to $b = 30$ units ($\hat{b} = 6$). Fig.9d shows the histograms for the HBSP method and simulations. The difference is now more significant for some classes. The estimated loss ratio was P_{loss}= 0.000234. The experiments were repeated 100 times in order to obtain the average difference for the histogram and loss ratio. The obtained average difference for the histograms was 0.214726 ± 0.000513 and for the loss ratio 0.013460 ± 0.000149. It can be said that there is a loss of accuracy due to reducing the number of classes of the workload definition: for example, in the second experiment the resulting pmf $\mathcal{Q}$ has only 5 classes.

The experiments were repeated *overclassing* the original workload in order to see if the differences were due to the information loss that grouping conveys or simply to the effect of a reduced number of classes in the algorithm. This way, A was overclassed by a factor of 5. The new histogram has 50 classes and an interval length of 1 unit. The results are shown in the Fig.9c and Fig.9d (line 'Histogram x 5'). Now, the resulting pmf $\mathcal{Q}$ has more clases (about 25) and the results are more accurate. In the case of infinite buffer, the difference was reduced to 0.003552 ± 0.000211. With finite buffer, the difference was reduced to 0.023018 ± 0.000442 and the loss ratio was 0.014651. This results are more

[5] The histogram is now represented using a line chart with logarithmic Y scale to see the differences in detail. Classes are marked with points.

accurate than using the histogram without overclassing. The influence of the number of classes and overclassing will be studied in more detail in a further subsection.

The experiments were repeated with different histograms, buffer and rate values. The results were very similar as the previous ones: the difference between the results obtained using the HBSP algorithm and the simulations were very low.

4.2 Real Traffic Experiments

There are several real traffic traces available such as the Qbone traces, NLANR from the University of Auckland, MAWI from the WIDE backbone, etc. Real traffic experiments are based on the MAWI traffic traces [33] due to their high resolution. This is a trace representing an IP traffic from a network link and its is suitable to test the histogram model. Specifically, we took a 1-hour trace from May 14, 1999 11:00 to 12:00 of a US-Japan link. This traffic trace has 6,016,846 packets with a total size of 3.6 *Gbytes* and an average rate of 8.43 Mb/s. These MAWI traces are in tcpdump raw format (near 9 Gbytes of traces), so we distilled them to obtain a simple file that contains the arrival time (in microseconds) and size (in bytes) of all the packets transmitted during this hour. Using a sampling period of $T = 40\ ms$ (25 samples per second), the resulting traffic trace has 90,000 frames (see Fig.10a). This is a bursty traffic with a burst ratio of 3.12 (the burst ratio is defined as the peak rate divided by the mean rate). The arrival load histogram of this traffic using 10 classes is shown in Fig.10b and has $E[A] = 0.253\ Mb$.

Simulations of a network node using this traffic trace were performed, although these simulations differ from synthetic workloads in that packet arrivals are given by the traffic trace, so it is not necessary to assume a uniform packet arrival rate. That leads to an event driven simulation. In each period, the simulation calculates the buffer length and lost packets. The simulation histogram and packet loss can be easily derived from these data. As the MAWI trace has a great resolution, this simulation is very realistic.

[Fig. 8 about here.]

In the first experiment, an infinite buffer was considered and the output rate was set to $R = 9\ Mb/s$ ($r = 0.36Mb$). The HBSP algorithm was executed considering the following arrival workloads: a) a pmf of 10 classes (as shown in Fig.10b) , b) a pmf of 100 classes and c) 10 classes with an overclassing factor of 10. For each simulation a histogram ($\mathcal{Q}_S$) was obtained. The results are in Fig.10c and show that there is an accuracy loss for buffer lengths greater than 50Kb. This is mainly due mainly to accuracy in performing convolutions:

values more to the right are the result of accumulating a lot of products. Nevertheless, the difference between the simulated histograms and the histogram obtained applying the algorithm are about 0.072, so they are really close.

In the second experiment the output rate was set to $R = 7\ Mb/s$ ($r = 0.27\ Mb$) and the buffer length was set to $b = 0.2\ Mb$. The same workloads than in previous experiments were considered. Fig.10d shows the results. Histograms with 100 classes and 10 classes with overclassing exhibit very accurate results. Regarding the loss ratio, the simulation provided a value of 0.138464 while the HBSP algorithm estimated a value of 0.215495 with 10 classes, 0.136039 with 100 classes and 0.136196 with 10 classes and overclassing, that are very close to the simulated one.

Previous experiments were also repeated with different traffic traces (using MAWI traces from another day and hour and even a trace from the NLANR repository), output rates and buffer lengths. Results were very similar to the ones presented here.

4.3 Accuracy

This subsection is devoted to identify and evaluate factors that may affect results accuracy.

The first experiment analyses the relation between buffer length and loss ratio. Loss ratios are calculated for different output rates varying the buffer length between 10 kb and 1 Mb (this corresponds to a maximal queue delay of nearly 0.1 s, that is nearly to the ITU G.114 delay recommendations). Results are presented in the form of a loss ratio curve of Fig.11a. Histogram prediction of loss rate is very accurate, since it is very close to simulations. Best results are obtained when the loss ratio is high. There is a loss of accuracy when the loss ratio is very low. This is mainly due to the convolution operator.

[Fig. 9 about here.]

The second experiment analyses the accuracy when we use long-term traces instead of short-term traces. Previous experiments use a 1-hour trace for obtaining the histogram, producing very good results. This experiment uses a 24-hour trace from May 14, 1999 (the same day of the previous trace), a rate $R = 8\ Mb/s$ and a buffer length $b = 0.2\ Mb$. This means using long-term traces instead of short-term traces. Results are still very accurate, as shown in Fig.11b. Regarding the loss ratio, the HBSP method predicted a value of 0.0064, while the simulation yielded 0.016. In summary there is a little loss of accuracy when using long-term traces, as it could be expected, due to information loss in the histogram definition.

The third experiment analyses the relation between the sampling period and accuracy. This experiment shows the differences between histograms obtained using the HBSP algorithm and simulations varying the sample rate from 0.01 s to 20 s. Results are shown in Fig.12a and Fig.12b. There is a little precision loss using higher sampling periods.

[Fig. 10 about here.]

The last experiment evaluates the relation between the number of classes of a histogram and accuracy. Previous experiments already showed that accuracy depends on the number of classes and the overclassing factor. Histograms can be a powerful and compact description of the traffic as long as they allow to obtain good accuracy a low number of classes. The key questions are: how many classes are necessary to get a good accuracy? and, when is necessary to use overclassing in order to obtain good results?.

This experiments uses the same scenarios than previous subsection (the 1-hour MAWI traffic). The output rate was set initially to $R = 9\ Mb/s$ and an infinite buffer was considered. The number of classes was varied from 6 to 100 and 4 histograms were calculated: the first one using the original histogram with no overclassing and the other 3 using overclassing factors of 5, 10 and 20. The normalized difference between these histograms and the one obtained through simulation is shown in Fig.13a. More experiments were done with a finite buffer length of $b = 0.2\ Mb$ and an output rate of $R = 7\ Mb/s$ and $b = 0.2\ Mb$. Results about the buffer length probability are in Fig.13b. The loss ratio is compared with the loss ratio of simulations in Fig.13c.

[Fig. 11 about here.]

Results show that the main effect of overclassing is to smooth the results reducing the original peeks. It can be also seen that there is no significant variation using a overclassing factor greater than 10. Regarding on the number of original classes, it can be seen that accuracy is not greatly improved using more than 15 of 20 classes. For the original histogram, accuracy is better in some cases using more than 60 classes (see Fig.13a) but in some other cases it is worst. Therefore, in the average case, it is better to use overclassing. The final conclusion is that the best results are obtained using 10 to 20 classes with an overclassing factor of 10.

4.4 Comparison with other methods

This section compares the proposed HBSP method with previous published methods for analysing buffer length and loss ratio. Regarding the calculation of the buffer length, the best known approach is the method introduced

by Skelly and Shroff (known as the *Histogram Model* [27] or the *Generalized Histogram Model* [28] and used with few modifications in [34] and [29]). This approximation is based on resolving an M/D/1/N queue for each arrival rate of the histogram. We implemented this method and compared the obtained buffer length histogram with the one obtained using the HBSP algorithm. Fig.14a shows that the differences between the HBSP method and the M/D/1/K method are really high. The results using the M/D/1/N are very bad. The problems with the M/D/1/K is that the buffer curve collapses when the buffer size if high. The results presented in [27] used very low buffer lengths (about 50 cells) so the results were more accurate. Nevertheless when larger buffer (about 500) the buffer curve begins to collapse.

[Fig. 12 about here.]

Regarding the loss ratio, there are a lot of methods for obtaining the packet or cell loss ratio. The most simple methods use only the peak or mean bandwidth of the traffic. Other methods obtain the loss ratio of a multiplexed (or aggregated) traffic in a network node based on certain known characteristics of the individual traffics (see [16,34,35] and references inside). These methods can not be compared with the one presented here because we obtain directly the loss ratio from the aggregated traffic. Other methods uses a fractal (or self-similar) model of the traffic (see [20, 25, 26] and references inside). The challenge is very interesting: try to model a traffic with 3 or 4 statistical parameters and feed these parameters through a process for obtaining performance results similar to the ones using original traffic. Until now, the results presented are inaccurate and depends mainly in the traffic statistical characteristics (it must be Gaussian). A most similar approach to the one presented here is partially described in [28]. In this case the traffic is temporally divided into parts, and in each part we obtain the loss ratio. Is easy to compare this approach with our HBSP method. In Fig.14b we can see that the results of using 'temporal division' (the X samples curves) versus using 'histogram division'. The results show that only with 10000 samples the 'temporal division' approaches to HBSP method.

5 Applications of the HBSP algorithm

There is a wide spectrum of applications of the HBSP method. We can obtain the traffic QoS parameters, as loss ratio or node delay using the HBSP method. Using the router delay of the nodes we can obtain the network delay pmf $\mathcal{D}_N$. This pmf is obtained as the sum (convolution) of the node pmfs that traverses a packet:

$$\mathcal{D}_N = \bigotimes_{i \in path} \mathcal{D}_i \tag{14}$$

This pmf is very useful because we can obtain the mean delay, or for example, the probability that a packet is delayed more than a certain value. For example, if we transmit video or audio, the delay histogram can be useful in the end nodes to adapt their transmissions rates or to configure the buffer in the reception nodes. This information can be used for *admission control* as well.

Another important application is for *traffic provisioning and network configuration*. Optimal provisioning of network resources is crucial for reducing the service cost of network transmission. This is the goal of *Traffic Engineering*: the design, provisioning, performance evaluation and tuning of operational networks. The fundamental problem with provisioning is to have methods and tools to decide the network resource reservation for a given Quality of Service requirements [36]. Therefore, the HBSP method can be very useful for Traffic Engineering.

The HBSP method allows to obtain the load histogram of the nodes of a network. These histograms can be used to configure the network. It also allows to evaluate parameters like the loss ratio (for a given buffer and output ratio), the node delay, the buffer/output ratio needed for a required loss, etc. One important decision that must be taken is the time-scale of the provisioning. The measured traffic can be a long-term trace (daily or weekly traces) or a short-term trace (hourly traces). This depends on the network capability to support dynamical variation in the reservation of the channel resources (for example, an hour) (see [37]).

A great advantage of the HBSP method is the easy implementation of the histograms. Is very easy to capture and store a load histogram with few classes (about 10) in a network node.

6 Extending the method

The proposed method assumes the hypothesis of *strong workload isolation* as a way to make several traffic flows in a router independent and, thus, easing system analysis. This hypothesis allows to consider that every traffic source has a constant service rate and there are no interferences or variations in the service rate due to other traffic sources which are multiplexed by the router using a packet scheduling algorithm. This section briefly outlines the implications of relaxing the hypothesis of workload isolation. This leads to a much more complex analysis, but it also shows the power of the method for analysing packet scheduling algorithms. It will be shown that, in this case, the probability distribution of the buffer length is also a stochastic process. The solution to this problem is a method that will be referred as the *interferences method*.

The service rate for each workload depends, in general, of the multiplexer bandwidth R, the set of workloads $\overline{\mathcal{A}} = \{\mathcal{A}_k, k : 0 \dots m\}$, and the packet scheduling algorithm, referred as $\digamma$.

The *interference histogram* of workload set $\overline{\mathcal{A}}$ on workload $\mathcal{A}_j$ in a multiplexer with scheduling algorithm $\digamma$, denoted as $\digamma_{\overline{\mathcal{A}}}(\mathcal{A}_j)$, is defined as the histogram of workload $\mathcal{A}_j$ after being processed by a router (at the output of a router), that is, multiplexing a workload set $\overline{\mathcal{A}}$ using the scheduling algorithm $\digamma$.

As an example of the interferences method, consider the workload $\mathcal{A}_0$ and the workload set $\overline{\mathcal{A}} = \{\mathcal{A}_0, \mathcal{A}_1\}$ whose execution time histograms are shown in figures 15a and 15b respectively. $\mathcal{A}_0$ has a uniform execution time in the interval $[1, 5]$ and $\mathcal{A}_1$ is a deterministic workload with execution time 3. Assuming a GPS algorithm and assigning the same weight to both workloads, the interference of $\mathcal{A}_1$ on $\mathcal{A}_0$, i.e. $\digamma_{\overline{\mathcal{A}}}(\mathcal{A}_0)$ is shown in Fig.15c. In the interval $[1, 3]$ both workloads share the processor with equal weight, so the time to execute $\mathcal{A}_0$ in this interval is double than when $\mathcal{A}_0$ is executing with fully processor utilization. At time 3 workload $\mathcal{A}_1$ finishes its execution and $\mathcal{A}_0$ gets the whole processor bandwidth. So when the execution time of $\mathcal{A}_0$ is in the interval $]3, 5]$ $\digamma_{\overline{\mathcal{A}}}(\mathcal{A}_0)$ is what $\mathcal{A}_1$ takes to complete plus the remaining execution time of $\mathcal{A}_0$ executing with 100% of the processor bandwidth. This can be calculated by making zero the elements of $\mathcal{A}_0$ in the interval $[0, 3]$ and convolutioning it with $\mathcal{A}_1$: $[0, 0, 0, 0, 0.2, 0.2] \otimes [0, 0, 0, 1] = [0, 0, 0, 0, 0, 0, 0, 0.2, 0.2]$.

[Fig. 13 about here.]

The calculus of the buffer length of $\mathcal{A}_j$ when workload isolation does not hold can be expressed as the following stochastic process:

$$\mathcal{Q}_j(K) = \Phi_r^l(\digamma_{\overline{\mathcal{A}(k-1)}}(\mathcal{Q}_j(k-1) \otimes \mathcal{A}_j)) \tag{15}$$

where $\overline{\mathcal{A}(k-1)}$ is the interfering task set in sampling period k. After the first iteration, the task set that interferes $\mathcal{A}_j$ may have changed. This is because workloads in the new sampling period have to be convolutioned with the corresponding pending workload. In other words, in each iteration, k the interfering workload has to be recalculated as:

$$\overline{\mathcal{A}(0)} = \overline{\mathcal{A}} \tag{16}$$

$$\overline{\mathcal{A}(k)} = \{\mathcal{Q}_j(k-1) \otimes \mathcal{A}_j, i = 1 \dots m\} \tag{17}$$

The calculus of $\digamma_{\overline{\mathcal{A}}}(\mathcal{A}_j)$ is $\digamma$-dependent and may have a high computational cost for some scheduling algorithms, but it allows to accurately find the solution to the buffer length problem in the general case. A method for computing

$F_{\overline{\mathcal{A}}}(\mathcal{A}_j)$ in the case of the GPS algorithm is presented in [31]. In [38] a similar method is used for the case of the RM (Rate Monotonic) algorithm in the context of real-time systems.

Summarizing, it could be stated that the property of workload isolation makes the system much easier to understand and to analyze, so it would be interesting to compare algorithms that provide this property against other algorithms (Rate Monotonic, etc.) not only in terms of performance but also of easy analyze ability.

7 Conclusions

This paper deals with a well known problem that can be resumed using the following question by Addie et al. [20]: *'Is there an accurate and useful traffic model in the form of a simple stochastic process which can be described by a small number of parameter and, when fed into a single server queue gives the same performance as a real traffic stream?'*.

This paper presents a new model to answer the question. The model is based in a stochastic process working with histograms (the HBSP model). There is no need for approximating the traffic source to a Poisson distribution (or any other distribution) nor solving queueing models. The HBSP model defines an iterative stochastic process using simple operators working with histograms (pmf).

This model is shown to be very accurate. Experiments were performed using synthetic and real-traffic traces. The results show that using a histogram of about 10 classes is enough to obtain good results, so the HBSP model is very compact.

Finally. we can affirm that the HBSP model answers in great manner the question stated by Addie: (a) it is compact: about 10 classes are needed to obtain accurate results (b) it is easy to implement: is simply to sample and store the traffic load of network routers in classes (c) it is accurate: the results obtained are very accurate (d) it is practical: from the buffer load histogram we can obtain another useful QoS parameters as loss ratio and delay .

A Appendix

This appendix contains a brief description of the buffer analysis as a Discrete Time Markov Chain and the proofs of the convergence of the HBSP algorithm.

First, we proof the convergence using infinite buffer and then, we proof the convergence with finite buffer.

A.1 Buffer Analysis as a Discrete Time Markov Chain

In this appendix we show that the $\{\mathcal{Q}(n)\}$ stochastic process is a Discrete-Time Markov Chain (DTMC). Additionally, we can easily obtain the transition probability matrix P. Using this probability matrix we can obtain the values for $\mathcal{Q}(n)$. The problem of using DTMF it that is not easy to obtain an analytical solution for the steady state (that is, when $n \to \infty$). Therefore, we used the iterative method described in this paper to obtain the buffer.

A Discrete-Time Markov Chain is a stochastic process whose probabilities distributions in state j only depends on the previous state i, and not on how the process arrived to state i. It is easy to proof that $\{\mathcal{Q}(n)\}$ is a DTMC. The probability that the buffer in period k takes the value j can be expressed using the buffer probabilities of period $k-1$ as follows:

$$P[\mathcal{Q}(k) = j] = \sum_i P[\mathcal{Q}(k) = i] \cdot P[\mathcal{Q}(k) = j | \mathcal{Q}(k-1) = i] \tag{A.1}$$

The term $p_{ij}(k-1,k) = P[\mathcal{Q}(k) = j | \mathcal{Q}(k-1) = i]$ denotes the probability that the process makes a transition from state i at period $k-1$ to state j at period k. This probability is obtained from the arrival load $\mathcal{A}$ and given that $\mathcal{A}$ is the same in all the periods, then the $p_{ij}(k-1,k)$ does not depend on the period k. Therefore, we can represent $p_{ij}(k-1,k)$ as p_{ij} and Eq.A.2 is reduced to:

$$P[\mathcal{Q}(k) = j] = \sum_i P[\mathcal{Q}(k) = i] \cdot p_{ij} \tag{A.2}$$

p_{ij} is known as the one-step transition probability. From this we can obtain the transition probability matrix:

$$P = [p_{ij}] = \begin{bmatrix} p_{00} & p_{01} & p_{02} & \cdots \\ p_{10} & p_{11} & p_{12} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{A.3}$$

The components of this matrix are easy to obtain using the definition of the stochastic process $\{\mathcal{Q}(n)\}$. That is, for obtaining the i-row of P we apply one iteration of the stochastic process using an initial load of one unit in j. For example, the first row is obtained as $\Phi_{\hat{r}}^{\hat{b}}([1,0,0,0,\ldots] \otimes \mathcal{A})$.

Using the matrix P we can obtain the pmf of $\mathcal{Q}(k)$ as:

$$\mathcal{Q}(k) = \mathcal{Q}(1)P^k \tag{A.4}$$

Nevertheless, determining the asymptotic behavior (that is, the *steady state*) poses problems. This implies obtaining the steady-state probability vector $\mathbf{v}$ as:

$$\mathbf{v} = \mathbf{v}P \qquad v_j \geq 0, \quad \sum_j v_j = 1 \tag{A.5}$$

As the matrix dimensions depends on the $\hat{r}$ and $\hat{b}$ values two cases are studied. When b is infinite (the no-buffer case), this matrix is infinite. This matrix is well studied in [30]. It is shown that the matrix presents a certain regularity in its rows and when the utilization is less than 1 it converges (it is a positive recurrent chain). In the other hand, when $\hat{b}$ is finite the matrix has a finite size of $\hat{b}+1 \times \hat{b}+1$ and it more amenable to work with it. Nevertheless, numerically resolving Eq.A.5 is not easy even for a little matrix. Therefore, we must use iterative methods as the *power method* or something similar.

Using the example of subsection refsubsec:alg, $\mathcal{A}$ =[0, 0.1, 0.4, 0.2, 0.15, 0.15] with $\hat{r}$=3 and $\hat{b}$=5 we obtain the following matrix:

$$P = \begin{bmatrix} 0.70 & 0.15 & 0.15 & 0.00 & 0.00 & 0.00 \\ 0.50 & 0.20 & 0.15 & 0.15 & 0.00 & 0.00 \\ 0.10 & 0.40 & 0.20 & 0.15 & 0.15 & 0.00 \\ 0.00 & 0.10 & 0.40 & 0.20 & 0.15 & 0.15 \\ 0.00 & 0.00 & 0.10 & 0.40 & 0.20 & 0.30 \\ 0.00 & 0.00 & 0.00 & 0.10 & 0.40 & 0.50 \end{bmatrix} \tag{A.6}$$

We can obtain the second iteration state as $\mathcal{Q}(2) = \mathcal{Q}(1)P$ = [0.580, 0.195, 0.1575, 0.045, 0.0225]. The steady state probability vector is $\mathbf{v}$ = [0.3275, 0.1625, 0.1699, 0.1291, 0.1077, 0.1033] that is the same result as using the HBSP algorithm.

A.2 Buffer Load Sequence with infinite buffer

The sequence $\{\mathcal{Q}(n)\}$ is defined as:

$$\begin{aligned} \mathcal{Q}(0) &= [1] \\ \mathcal{Q}(n) &= \Phi_{\hat{r}}(\mathcal{Q}(n-1) \otimes \mathcal{A}) \end{aligned}$$

Proving the convergence directly with histograms can be a very difficult task. Additionally, we need a clear convergence criteria in order to implement a direct algorithm. Thus, we convert the previous sequence $\{\mathcal{Q}(n)\}$ to a sequence of means $\{E[\mathcal{Q}(n)]\}$. Then, our goal is prove the convergence of this sequence. We can easily see that if $\{E[\mathcal{Q}(n)]\}$ converges then $\{\mathcal{Q}(n)\}$ converges.

Theorem 1 *If* $\{E[\mathcal{Q}(n)]\}$ *converges then* $\{\mathcal{Q}(n)\}$ *converges.*

PROOF. $\{\mathcal{Q}(n)\}$ is a sequence of histograms $\{[q_0, q_1 \cdots q_m]\}$ $q_i \geq 0$. If $\{\mathcal{Q}(n)\}$ converges then $\lim_{n\to\infty} q_i = c_i$ $0 \leq i \leq m$. Then, if $\{E[\mathcal{Q}(n)]\}$ converges we have:

$$\lim_{n\to\infty} E[\mathcal{Q}(n)] = \lim_{n\to\infty} \sum_{i=0}^{m} q_{i_n} \cdot i = \sum_{i=0}^{m} \lim_{n\to\infty} q_{i_n} \cdot i = C \tag{A.7}$$

By definition $q_i \geq 0$, so if some element q_i does not converge then $\lim_{n\to\infty} q_{i_n} = \infty$. That is a contradiction with Eq.A.7. Therefore, we have an histogram limit $[c_0, c_1 \cdots c_m]$:

$$\lim_{n\to\infty} q_{i_n} = c_i \quad \sum_{i=0}^{m} c_i \cdot i = C$$

□

We calculated the first 30 elements of the sequence using the load of section 3.2. This arrival load has a mean of 2.85. Additionally, we calculated $E[\mathcal{I}(k)]$ and the difference $E[\mathcal{I}(k)] - E[\mathcal{Q}(k)]$. The results are in table A.1. The table shows that the sequence converges. Furthermore, the sequence $E[\mathcal{I}(k)] - E[\mathcal{Q}(k)]$ converges to $E[\mathcal{A}]$. That is:

$$\lim_{n\to\infty} (E[\mathcal{I}(n)] - E[\mathcal{Q}(n)]) = E[\mathcal{A}] < \hat{r} \tag{A.8}$$

This means that the traffic that is sent to network in each iteration is $E[\mathcal{A}]$. This is logical, because if we send less traffic than $E[\mathcal{A}]$ this traffic is accumulated in $E[\mathcal{I}]$ and the algorithm does not converge.

[Table 1 about here.]

The main difficult of proving the convergence of the sequence is the *bound operator* $\Phi_a(\mathcal{P})$. We can obtain an expression for $\Phi_a()$

Theorem 2 $E[\Phi_a(\mathcal{P})] = E[\mathcal{P}] - a + \sum_{i=0}^{a-1} p_i \cdot (a - i)$.

PROOF. By definition, we have that $E[\mathcal{P}] = \sum_{i=0}^{n} p_i \cdot i$. If we apply the $\Phi_a()$ operator to $\mathcal{P}$ we have:

$$E[\Phi_a(\mathcal{P})] = \sum_{i=0}^{n-a} p_{i+a} \cdot i = \sum_{i=a}^{n} p_i \cdot (i - a) = \sum_{i=a}^{n} p_i \cdot i - \sum_{i=a}^{n} p_i \cdot a$$

If we add and subtract the following term $\sum_{i=0}^{a-1} p_i \cdot i$ and regroup terms, we have:

$$= \underbrace{\sum_{i=a}^{n} p_i \cdot i + \sum_{i=0}^{a-1} p_i \cdot i}_{E[\mathcal{P}]} - \sum_{i=0}^{a-1} p_i \cdot i - \sum_{i=a}^{n} p_i \cdot a = E[\mathcal{P}] - \sum_{i=0}^{a-1} p_i \cdot i - \sum_{i=a}^{n} p_i \cdot a$$

Now if we add and subtract the following term $\sum_{i=0}^{a-1} p_i \cdot a$ to the final term we have:

$$E[\mathcal{P}] - \sum_{i=0}^{a-1} p_i \cdot i - (\underbrace{\sum_{i=a}^{n} p_i \cdot a + \sum_{i=0}^{a-1} p_i \cdot a}_{\sum_{i=0}^{n} p_i = 1} - \sum_{i=0}^{a-1} p_i \cdot a) =$$

$$E[\mathcal{P}] - \sum_{i=0}^{a-1} p_i \cdot i - a + \sum_{i=0}^{a-1} p_i \cdot a = E[\mathcal{P}] - a + \sum_{i=0}^{a-1} p_i \cdot (a - i)$$

□

Corolian 1 *$E[\Phi_a(\mathcal{P})] = E[\mathcal{P}] - a$ if $\sum_{i=0}^{a-1} p_i = 0$.*

PROOF. Applying the previous theorem we can see that the term $\sum_{i=0}^{a-1} p_i \cdot (a - i)$ is zero. Therefore $E[\Phi_a(\mathcal{P})] = E[\mathcal{P}] - a$. □

Now, we proof the convergence of $E[Q]$.

Theorem 3 *The sequence $E[\mathcal{Q}(n)]$ converges if $E[\mathcal{A}] \leq \hat{r}$ and $M[\mathcal{A}] > \hat{r}$*

PROOF. Proving directly that $E[\mathcal{Q}(n)]$ is increasing has the problem that the $\Phi_a()$ operator is not linear. Therefore, instead of proving the convergence of $E[\mathcal{Q}(n)]$ we will prove that $E[\mathcal{Q}(n) \otimes \mathcal{D}]$ converges. The histogram $\mathcal{D}$ is selected as $p_i = 0 \quad \forall i < \hat{r}$ and $p_{\hat{r}} = 1$. The convolution of $\mathcal{P}$ with $\mathcal{D}$ has the effect of shifting $\hat{r}$ positions to the right the histogram $\mathcal{P}$.

It is easy to see that if we prove that $E[\mathcal{Q}(n) \otimes D]$ converges, then $E[\mathcal{Q}(n)]$ converges:

$$E[\mathcal{Q}(n) \otimes \mathcal{D}] = E[\mathcal{Q}(n)] + E[\mathcal{D}] = E[\mathcal{Q}(n)] + \hat{r}$$
$$E[\mathcal{Q}(n) \otimes \mathcal{D}] - \hat{r} = E[\mathcal{Q}(n)]$$

Therefore, as $E[\mathcal{Q}(n)] = E[\mathcal{Q}(n) \otimes \mathcal{D}] - \hat{r}$ and $\hat{r}$ is constant, if $E[\mathcal{Q}(n) \otimes \mathcal{D}]$ converges then $E[\mathcal{Q}(n)]$ converges.

To prove the convergence of a sequence we need to prove that is increasing and bounded.

FIRST: $E[\mathcal{Q}(n) \otimes \mathcal{D}]$ is increasing when $M[\mathcal{A}] > \hat{r}$. We proof by induction.

For $n = 1$, we have to prove that $E[\mathcal{Q}(0) \otimes \mathcal{D}] < E[\mathcal{Q}(1) \otimes \mathcal{D}]$. Then we have:

$$\begin{gathered} E[\mathcal{Q}(0) \otimes \mathcal{D}] < E[\mathcal{Q}(1) \otimes \mathcal{D}] \\ E[\mathcal{Q}(0)] + E[\mathcal{D}] < E[\mathcal{Q}(1)] + E[D] \\ E[\mathcal{Q}(0)] + \hat{r} < E[\mathcal{Q}(1)] + \hat{r} \\ E[\mathcal{Q}(0)] < E[\mathcal{Q}(1)] \end{gathered}$$

So, we have to prove that $E[\mathcal{Q}(1)] > E[\mathcal{Q}(0)]$. As $E[\mathcal{Q}(0)] = 0$, then $E[\mathcal{Q}(1)]$ must be greater than 0. Therefore:

$$E[\mathcal{Q}(1)] = E[\Phi_{\hat{r}}(\mathcal{Q}(0) + \mathcal{A})] = E[\Phi_{\hat{r}}(0 + \mathcal{A})]$$

It is easy to see that the last term is greater than zero only if $M[\mathcal{A}] > \hat{r}$

For $n = k$, given that $E[\mathcal{Q}(k) \otimes \mathcal{D}] < E[\mathcal{Q}(k+1) \otimes \mathcal{D}]$ we have to prove that $E[\mathcal{Q}(k+1) \otimes \mathcal{D}] < E[\mathcal{Q}(k+2) \otimes \mathcal{D}]$. Then, if we add $E[\mathcal{A}] - \hat{r}$ in both sides we have:

$$\begin{gathered} E[\mathcal{Q}(k) \otimes \mathcal{D}] + E[\mathcal{A}] - \hat{r} < E[\mathcal{Q}(k+1) \otimes \mathcal{D}] + E[\mathcal{A}] - \hat{r} \\ E[\mathcal{Q}(k) \otimes \mathcal{D} \otimes \mathcal{A}] - \hat{r} < E[\mathcal{Q}(k+1) \otimes \mathcal{D} \otimes \mathcal{A}] - \hat{r} \\ E[(\mathcal{Q}(k) \otimes \mathcal{A}) \otimes \mathcal{D}] - \hat{r} < E[(\mathcal{Q}(k+1) \otimes \mathcal{A}) \otimes \mathcal{D}] - \hat{r} \end{gathered}$$

The effect of the convolution of $\mathcal{D}$ with the left term is shift to the right $\hat{r}$ positions. Therefore, we have $E[\Phi_{\hat{r}}(\mathcal{P} \otimes \mathcal{D})]$ is $E[\mathcal{P} \otimes \mathcal{D}] - \hat{r}$.

$$E[\Phi_{\hat{r}}(\mathcal{Q}(k) \otimes \mathcal{A} \otimes \mathcal{D})] < E[\Phi_{\hat{r}}(\mathcal{Q}(k+1) \otimes \mathcal{A} \otimes \mathcal{D})]$$

We can see that $E[\Phi_{\hat{r}}(\mathcal{Q}(k) \otimes \mathcal{A} \otimes \mathcal{D})]$ is $E[\mathcal{Q}(k+1) \otimes \mathcal{D}]$ and $E[\Phi_{\hat{r}}(\mathcal{Q}(k+1) \otimes \mathcal{A} \otimes \mathcal{D})]$ is $E[\mathcal{Q}(k+2) \otimes \mathcal{D}]$. So $E[\mathcal{Q}(k+1) \otimes \mathcal{D}] < E[\mathcal{Q}(k+2) \otimes \mathcal{D}]$.

SECOND: $E[\mathcal{Q}(n) \otimes \mathcal{D}]$ is bounded when $E[\mathcal{A}] \leq \hat{r}$.

First, we choose $\mathcal{C}$ as $[0, \cdots, p_{\hat{r}}, \cdots p_n]$ so $E[\mathcal{C}] \geq \hat{r}$. Applying the previous corolian we can see that $\Phi_{\hat{r}}(\mathcal{C}) = E[\mathcal{C}] - \hat{r}$.

Now we proof that $E[\mathcal{Q}(n) \otimes \mathcal{D}]$ is bounded by $E[\mathcal{C}]$. We proof by induction.

For $n = 1$, we have:

$$E[\mathcal{Q}(1) \otimes \mathcal{D}] = E[\Phi_{\hat{r}}(\mathcal{A} \otimes \mathcal{D})] = E[\mathcal{A} \otimes \mathcal{D}] - \hat{r} = E[\mathcal{A}] + E[\mathcal{D}] - \hat{r} = E[\mathcal{A}]$$

So, if $E[\mathcal{A}] \leq \hat{r}$ then $E[\mathcal{A}] \leq E[\mathcal{C}]$.

For $n = k$ we have that $E[\mathcal{Q}(k) \otimes \mathcal{D}] < E[\mathcal{C}]$. If we add in both sides $E[\mathcal{A}] - \hat{r}$ we have:

$$E[\mathcal{Q}(k) \otimes \mathcal{D}] + E[\mathcal{A}] - \hat{r} < E[\mathcal{C}] + E[\mathcal{A}] - \hat{r}$$
$$E[\mathcal{Q}(k) \otimes \mathcal{D} \otimes \mathcal{A}] - \hat{r} < E[\mathcal{C} \otimes \mathcal{A}] - \hat{r}$$

As $\mathcal{P} = \mathcal{Q}(k) \otimes \mathcal{D} \otimes \mathcal{A}$ has $\sum_{i=0}^{\hat{r}-1} p_i = 0$ then we can replace $E[\mathcal{Q}(k) \otimes \mathcal{D} \otimes \mathcal{A}] - \hat{r}$ by $E[\Phi_{\hat{r}}(\mathcal{Q}(k) \otimes \mathcal{D} \otimes \mathcal{A})]$.

$$E[\Phi_{\hat{r}}(\mathcal{Q}(k) \otimes \mathcal{D} \otimes \mathcal{A})] < E[\mathcal{C} \otimes \mathcal{A}] - \hat{r}$$

As $E[\Phi_{\hat{r}}(\mathcal{Q}(k) \otimes \mathcal{D} \otimes \mathcal{A})]$ is $E[\mathcal{Q}(k+1) \otimes \mathcal{D}]$ and $E[\mathcal{A}] \leq \hat{r}$ then:

$$E[\mathcal{Q}(k+1) \otimes \mathcal{D}] < E[\mathcal{C}] + E[\mathcal{A}] - \hat{r} < E[\mathcal{C}] + \hat{r} - \hat{r} = E[\mathcal{C}]$$

□

A.3 Buffer Load Sequence with finite buffer

In this appendix we proof the convergence of the following sequence.

$$\mathcal{Q}(0) = [1]$$
$$\mathcal{Q}(n) = \Phi_{\hat{r}}^{\hat{b}}(\mathcal{Q}(n-1) \otimes \mathcal{A})$$

As in the previous demonstration we will proof the convergence of $\{E[\mathcal{Q}(n)]\}$. We proof that is increasing and bounded.

Theorem 4 *The sequence $E[\mathcal{Q}(n)]$ converges if $M[\mathcal{A}] > \hat{r}$*

PROOF.

FIRST: $E[\mathcal{Q}(n)]$ is increasing when $M[\mathcal{A}] > \hat{r}$.

We want to proof that in each iteration $E[\mathcal{Q}(i)]$ is increasing. We distinguish two cases:

(1) While $M[\mathcal{Q}(i)] < \hat{b}$. This is the same as $\hat{b} = \infty$ so it is increasing.
(2) If $M[\mathcal{Q}(i)] \geq \hat{b}$. In this case all the probabilities that are greater than $\hat{b}$ are acummulated in $p_{\hat{b}}$, so it is easy to see that is increasing.

SECOND: $\{E[\mathcal{Q}(n)]\}$ is bounded by $\hat{b}$.

This is easy to proof because the operator $\Phi_{\hat{r}}^{\hat{b}}()$ always cut probabilities upper than $\hat{b}$ so $\{E[\mathcal{Q}(n)]\}$ is always bounded by $\hat{b}$.

□

References

[1] C. M. Aras, J. F. Kurose, D. S. Reeves, H. Schulzrinne, Real-time communication in packet-switched networks, Proceedings of the IEEE 82 (41) (1994) 122–139.

[2] R. Braden, L. Zhang, S. Berson, S.Herzog, S. Jamin, Resource reservation protocol (rsvp). versin 1 functional specification, RFC 2205.

[3] A.K.Parekh, R.G.Gallager, A generalized processor sharing approach to flow control in integrated services networks: The single node case, IEEE/ACM Transactions on Networking 1 (3) (1993) 344–357.

[4] A.K.Parekh, R.G.Gallager, A generalized processor sharing approach to flow control in integrated services networks: multiple node case, IEEE/ACM Transactions on Networking 2 (2) (1994) 137–150.

[5] R. L. Cruz, A calculus for network delay, part i : Network elements in isolation, IEEE/ACM Transactions on Information Theory 37 (1) (1991) 114–131.

[6] R. L. Cruz, A calculus for network delay, part ii : Network analysis, IEEE/ACM Transactions on Information Theory 37 (1) (1991) 132–141.

[7] L. Georgiadis, R. Guerin, V. Peris, K. Sivarajan, Efficient network qos provisioning based on per node traffic shaping, IEEE/ACM Transactions on Networking 4 (4) (1996) 482–501.

[8] J.-Y. L. Boudec, P. Thiran, Network Calculus, Vol. 2050 of Lecture Notes in Computer Science, Springer Verlag, 2002.

[9] E. Wrege, E. W. Knightly, H. Zhang, J. Liebeherr, Deterministic delay bounds for vbr video in packet-switching networks : Fundamental limits and practical tradeoffs, IEEE/ACM Transactions on Information Theory 4 (3) (1996) 352–362.

[10] J. Schmitt, M. Karsten, L. Wolf, R. Steinmetz., Aggregation of guaranteed service flows., in: IWQoS: International Workshop on Quality of Service, 1999.

[11] H. Fu, E. W. Knightly, Aggregation and scalable qos: A performance study, in: IWQoS: International Workshop on Quality of Service, 2001.

[12] E. Knightly, P. Rossaro, On the effects of smoothing for deterministic qos, Distributed Systems Engineering Journal 4 (1) (1997) 3–15.

[13] J. Salehi, Z. Zhang, J. Kurose, D. Towsley, Supporting stored video : Reducing rate variability and end-to-end resource requirements through optimal smoothing, IEEE/ACM Transactions on Networking 6 (4) (1998) 397–410.

[14] S. Floyd, V. Jacobson, Random early detection gateways for congestion avoidance, IEEE/ACM Transactions on Networking 1 (4) (1993) 397–412.

[15] D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, An architecture for differentiated services, RFC 2475.

[16] E. W. Knightly, N. Shroff, Real-time transport of mpeg video with a statistically guaranteed loss ratio in atm networks, IEEE Network (1999) 20–29.

[17] D. Ferrari, D. Verma, A scheme for real-time channel establishment in wide-area networks, IEEE Journal of Selected Areas Communication 8 (2) (1990) 368–379.

[18] H. Zhang, E. W. Knightly, Rcsp and stop-and-go: A comparison of two non-work-conserving disciplines for supporting multimedia communication, ACM/Springer-Verlag Multimedia Systems Journal 4 (6).

[19] Z. L. Zhang, D. Towsley, J. Kurose, Statistical analysis of the generalized processor sharing scheduling discipline, IEEE Journal of Selected Areas Communication 14 (6) (1995) 1071–1080.

[20] R. G. Addie, M. Zukerman, T. D. Neame, Broadband traffic modeling: Simple solutions to hard problems, IEEE Communications Magazine (1998) 88–95.

[21] S. Schenker, C.Partridge, R.Guerinm, Specification of guaranteed quality of service, RFC 2212.

[22] E.W.Knightly, H. Zhang, Traffic characterization and switch utilization using a deterministic bounding interval dependent traffic model, in: IEEE Infocom, 1995.

[23] E. Hernández-Orallo, J. Vila-Carbó, A new approach to optimise bandwidth reservation for real-time video transmission with deterministic guarantees, Real-Time Imaging 9 (1) (2003) 11–26.

[24] W.E.Leland, M.S.Taqqu, W.Willinger, D.V.Wilson, On the self-similar nature of ethernet traffic (extended version), IEEE/ACM Transactions on Networking 2 (1) (1994) 1–15.

[25] B. Tsybakov, N. Georganas, On self-similar traffic in atm queues: definitions, overflow probability bound, and cell delay distribution, IEEE/ACM Transactions on Networking 5 (3) (1997) 397–409.

[26] M. Zukerman, T. D. Neame, R. G. Addie, Internet traffic modeling and future technology implications, in: IEEE Infocom, 2003.

[27] P. Skelly, M. Schwartz, S. Dixit, A histogram-based model for video traffic behavior in an atm multiplexer, IEEE/ACM Transactions on Networking 1 (4) (1993) 446–459.

[28] N. B. Shroff, M. Schwartz, Video modeling withing networks using deterministic smoothing at the source, in: IEEE Infocom, 1994, pp. 342–349.

[29] S.-K. Kweon, K. G. Shin, Real-time transport of mpeg video with a statistically guaranteed loss ratio in atm networks, IEEE Transactions In Parallel and Distributed Computing 12 (4) (2001) 387–403.

[30] J. Dìaz, Tecnicas estocasticas para el calculo del tiempo de respuesta en sistemas de tiempo real, Phd thesis, Universidad de Oviedo, Spain (2003).

[31] J. Vila, E. Hernandez, Histogram based analysis of real-time systems and networks, Tech. Rep. UPV-DISCA-06-01, Dept. Informatica de Sistemas y Computadores. Universidad Politecnica de Valencia, Valencia, Spain (January 2006).

[32] L. Kleinrock, Queueing Systems. Volume 2: Computer Applications, Wiley-Interscience, New York, 1976.

[33] K. Cho, et al, Traffic data repository at the wide project, in: USENIX 2000 FREENIX Track, 2000.

[34] N. B. Shroff, M. Schwartz, Improved loss calculations at an atm multiplexer, IEEE/ACM Transactions on Networking 6 (4) (1998) 411–21.

[35] M. Krunz, R. Sass, H. Huhhes, Statistical characteristics and multiplexing of mpeg streams, in: IEEE Infocom, 1995, pp. 452–462.

[36] D. Awduche, et al, Overview and principles of internet traffic engineering, RFC 3272.

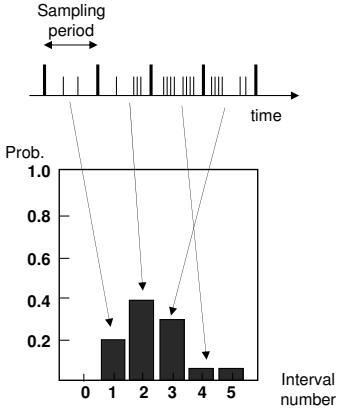
[37] E. Hernández-Orallo, J. Vila-Carbó, S. Saez-Barona, S. Terrasa-Barrena, Provisioning expedited forwarding diffserv channels using multimedia aggregates, in: Euromicro 2004, 2004.

[38] J. Diaz, D. Garcia, K. Kim, C. Lee, L. L. Bello, J. López, S. L.Min, O.Mirabella, Stochastic analysis of periodic real-time systems, in: Proc. of the 23rd IEEE Real-Time Systems Symposium,, 2002, pp. 289–300.

List of Figures

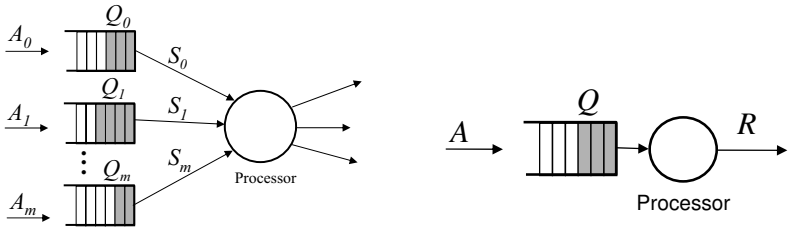
Class number	Interval	Midpoint $\dot{X}$	Probability $\mathcal{X}$	Cumulative probability
i	$[x_i^-, x_i^+[$	x_i	$p_X(i)$	$p_X^+(i)$
0	$[0, 20[$	10	0	0
1	$[20, 40[$	30	0.1	0.1
2	$[40, 60[$	50	0.4	0.5
3	$[60, 80[$	70	0.2	0.7
4	$[90, 100[$	90	0.15	0.85
5	$[100, 120[$	110	0.15	1.0

(a) Grouped probability distribution



(b) Histogram

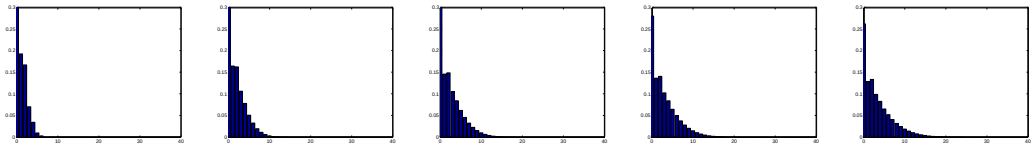
Fig. A.1. This figure shows the sampling of a traffic workload. The traffic load range is $[0, 120[$ kb.



(a) General scenario

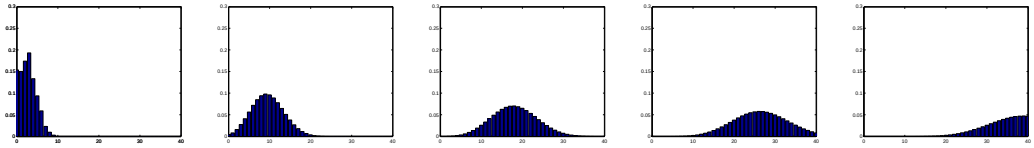
(b) Scenario with strong workload isolation

Fig. A.2. Scenarios of the buffer length problem



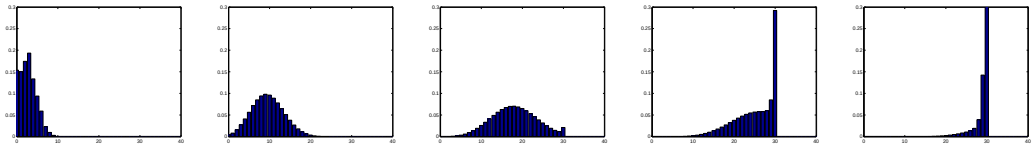
(a) Iteration 2 (b) Iteration 10 (c) Iteration 20 (d) Iteration 30 (e) Iteration 45

Fig. A.3. Stochastic process with $E[\mathcal{A}] = 2.85$, $\hat{r} = 3$ and infinite buffer



(a) Iteration 2 (b) Iteration 10 (c) Iteration 20 (d) Iteration 30 (e) Iteration 45

Fig. A.4. Stochastic process with $E[\mathcal{A}] = 2.85$, $\hat{r} = 2$ and infinite buffer

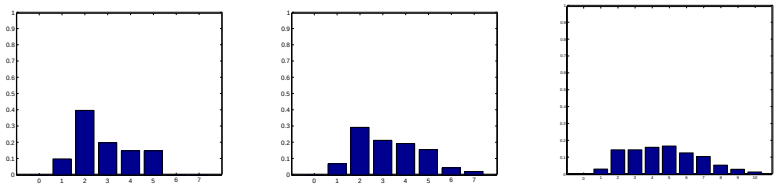


(a) Iteration 2 (b) Iteration 10 (c) Iteration 20 (d) Iteration 30 (e) Iteration 45

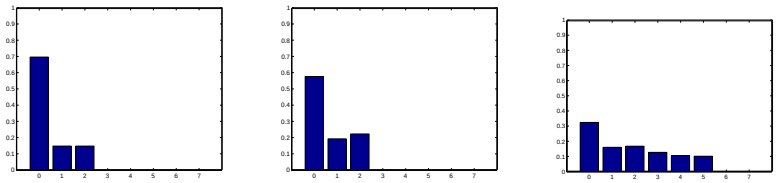
Fig. A.5. Stochastic process with $E[\mathcal{A}] = 2.85$, $\hat{r} = 2$ and a finite buffer of 30 bits

	Algorithm HBSP($\mathcal{A}$,$\hat{r}$,$\hat{b}$)
	$\mathcal{A}$: arrival rate
	$\hat{r}$: service rate class
	$\hat{b}$: buffer length class
1	$\mathcal{Q}(0) = [1]$
2	$k = 0$
3	**do**
4	$k = k + 1$
5	$\mathcal{I}(k) = \mathcal{Q}(k-1) \otimes \mathcal{A}$
6	$\mathcal{Q}(k) = \Phi_{\hat{r}}^{\hat{b}}(\mathcal{I}(k))$
7	**while** $E[\mathcal{Q}(k)] - E[\mathcal{Q}(k-1)] > \varepsilon$
8	**return** $\mathcal{Q}(k)$

Fig. A.6. HBSP algorithm

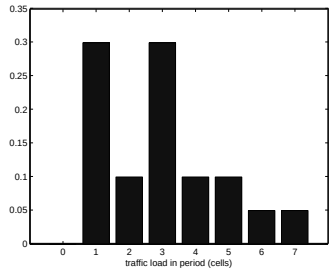


(a) $\mathcal{A}$ (b) $\mathcal{I}(2)$ (c) Stationary $\mathcal{I}$

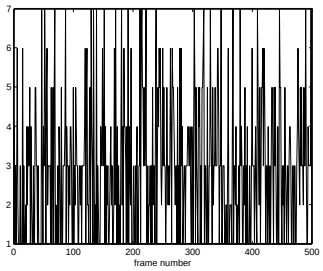


(d) $\mathcal{Q}(1)$ (e) $\mathcal{Q}(2)$ (f) Stationary $\mathcal{Q}$

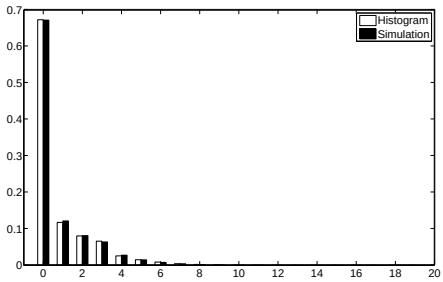
Fig. A.7. Evolution of the buffer histogram with the HBSP algorithm.



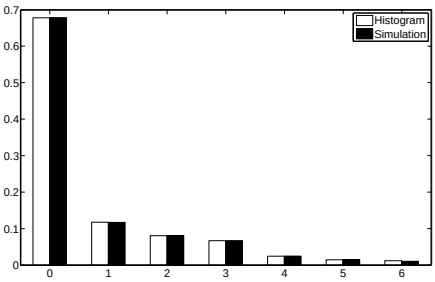
(a) Load histogram ($\mathcal{A}$)



(b) Synthetic traffic (first 500 samples)

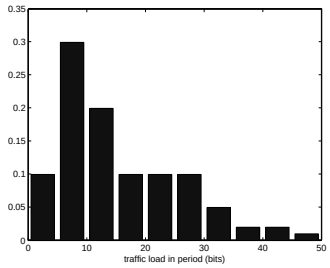


(c) Infinite buffer experiment

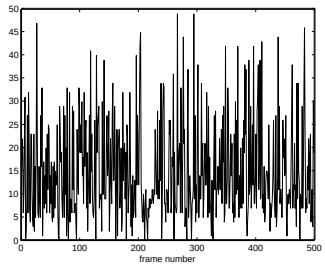


(d) Finite buffer experiment

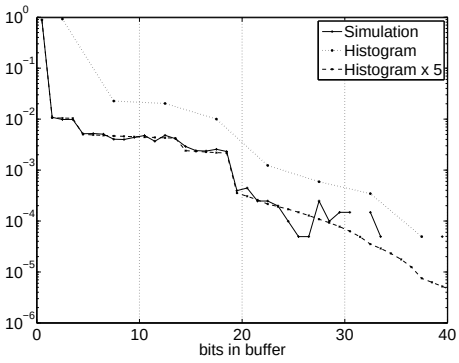
Fig. A.8. Histogram Experiments with an interval size of 1 unit



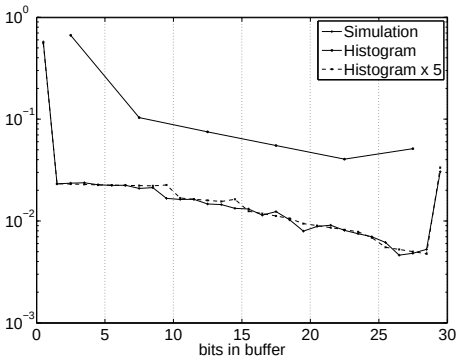
(a) Load histogram ($\mathcal{A}$)



(b) Synthetic traffic (first 500 samples)

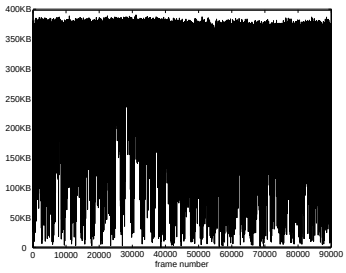


(c) Infinite buffer experiment results

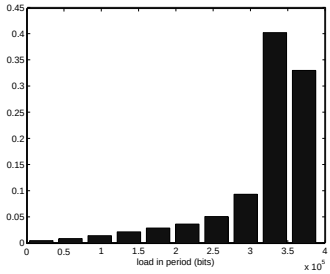


(d) Finite buffer experiment results

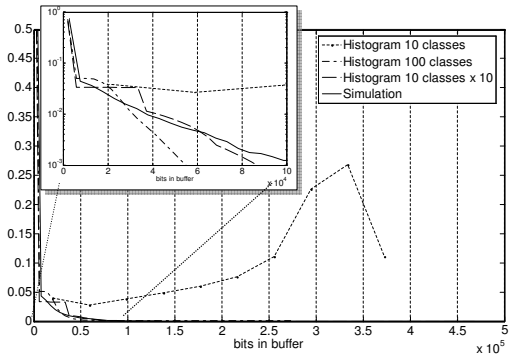
Fig. A.9. Histogram Experiments with an interval size of 5



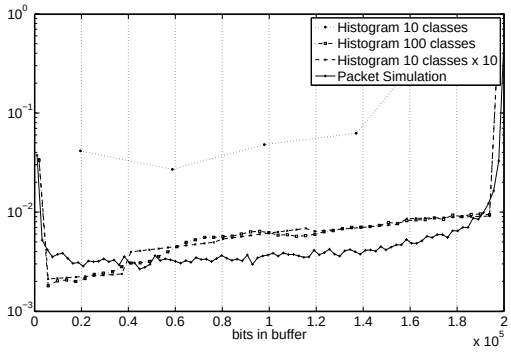
(a) MAWI traffic trace



(b) MAWI arrival load histogram

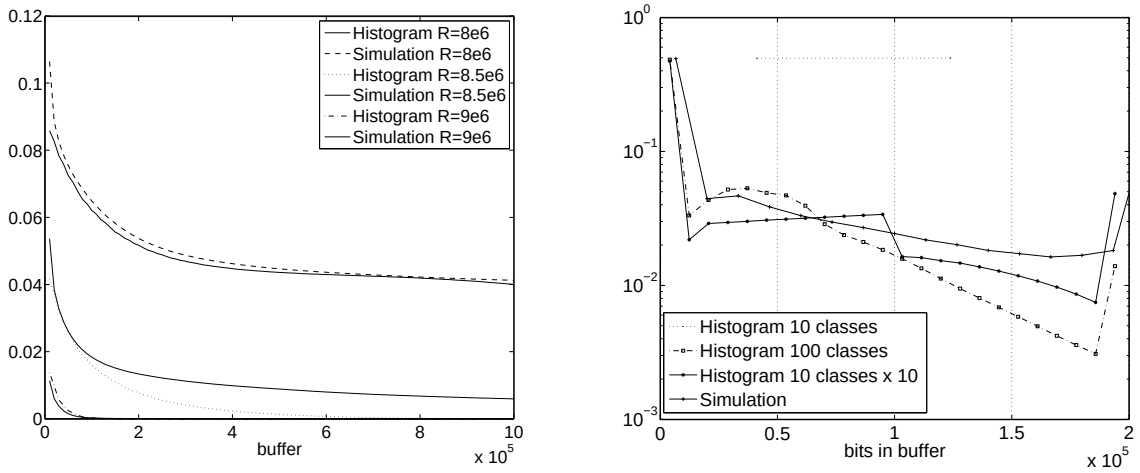


(c) Infinite buffer experiment



(d) Finite buffer experiment

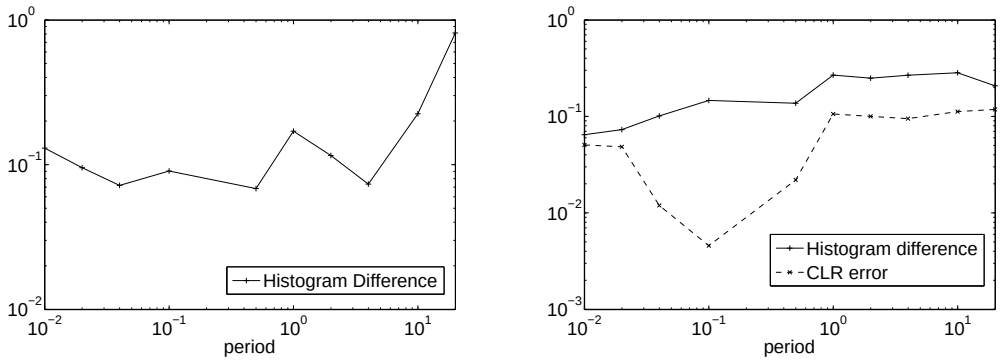
Fig. A.10. Experiment results using MAWI Traffic traces.



(a) Loss Ratio Curve

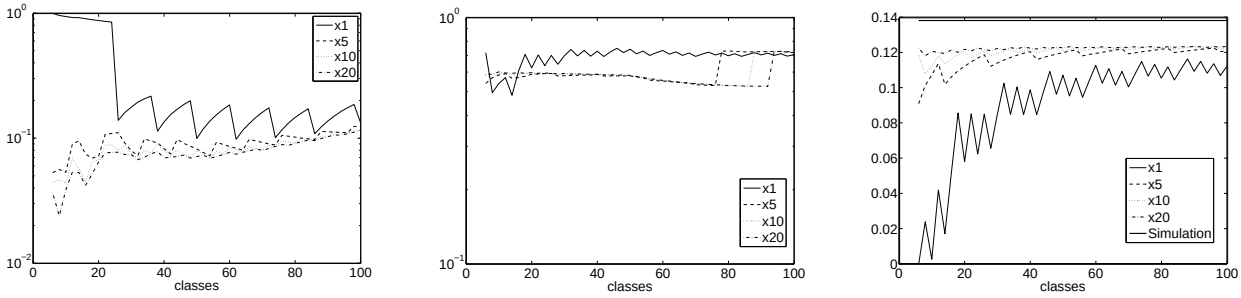
(b) 24-hour finite buffer experiment

Fig. A.11. Loss Curve Precision and Long-term buffer experiment.



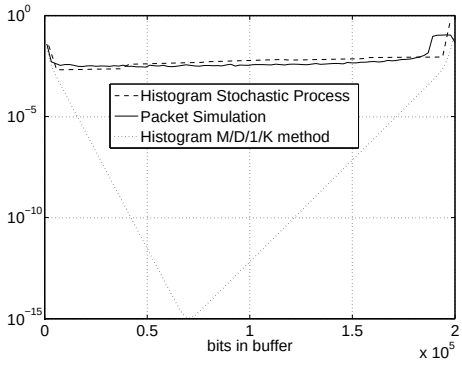
(a) R = 9Mb/s Infinite buffer (b) R = 7Mb/s b = 0.2Mb

Fig. A.12. Sample Period and precision.

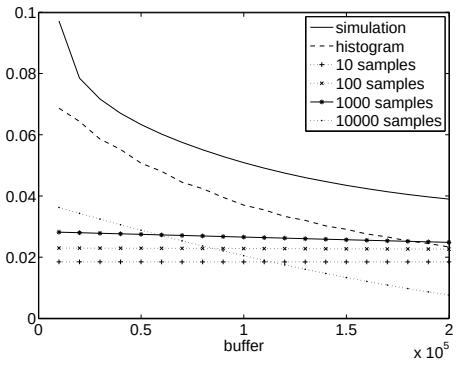


(a) R = 9Mb/s Infinite buffer (b) R = 7Mb/s b = 0.2Mb (c) CLR R = 7Mb/s b = 0.2Mb

Fig. A.13. Classes and precision.

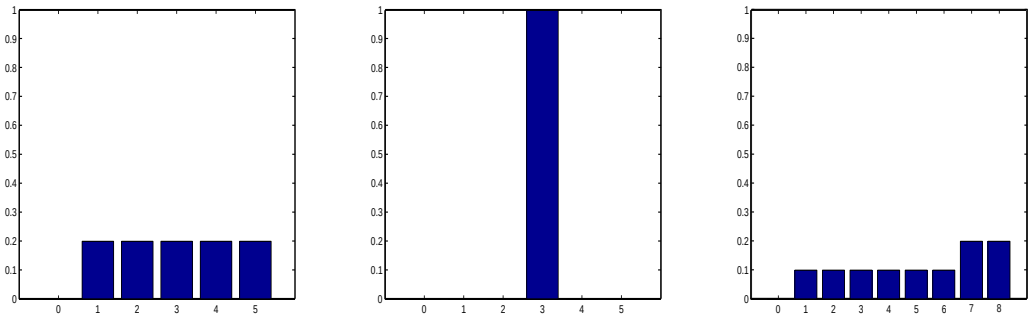


(a) MD1K results r = 700000bps b = 200000bps



(b) CLR r = 800000bps b = 400000bps

Fig. A.14. Comparison with other methods.



(a) Workload $\mathcal{A}_0$ (b) Workload $\mathcal{A}_1$ (c) Interference of $\mathcal{A}_1$ on $\mathcal{A}^0$

Fig. A.15. Example of the interferences method

List of Tables

Table A.1
Sequence values

k	$\mathcal{I}_k$	$\mathcal{B}_k$	$\mathcal{I}_k - \mathcal{B}_k$	k	$\mathcal{I}_k$	$\mathcal{B}_k$	$\mathcal{I}_k - \mathcal{B}_k$
1	2.85	0.45	2.4	16	4.6136	1.7774	2.8362
2	3.3	0.735	2.565	17	4.6274	1.7886	2.8387
3	3.585	0.9491	2.6358	18	4.6386	1.7977	2.8408
4	3.7991	1.1187	2.6804	19	4.6477	1.8051	2.8425
5	3.9687	1.2542	2.7144	20	4.6551	1.8112	2.8439
6	4.1042	1.3636	2.7406	21	4.6612	1.8161	2.8450
7	4.2136	1.4522	2.7613	22	4.6661	1.8201	2.8460
8	4.3022	1.5242	2.7780	23	4.6701	1.8233	2.8467
9	4.3742	1.5826	2.7915	24	4.6733	1.8260	2.8473
10	4.4326	1.6302	2.8024	25	4.6760	1.8281	2.8478
11	4.4802	1.6689	2.8113	26	4.6781	1.8298	2.8482
12	4.5189	1.7003	2.8185	27	4.6798	1.8313	2.8485
13	4.5503	1.7259	2.8244	28	4.6813	1.8324	2.8488
14	4.5759	1.7467	2.8291	29	4.6824	1.8334	2.8490
15	4.5967	1.7636	2.8330	30	4.6834	1.8341	2.8492