Stochastic Bounds and Histograms for Active Queues Management and Networks Analysis

F. Aït-Salaht[1], H. Castel-Taleb[2], J.M. Fourneau[3], and N. Pekergin[4]

[1] LIP6, Pierre et Marie Curie University UMR7606, Paris, France
farah.ait-salaht@lip6.fr
[2] SAMOVAR, UMR 5157, Télécom Sud Paris, Evry, France
hind.castel@telecom-sudparis.eu
[3] DAVID, Versailles St-Quentin University, Versailles France
jean-michel.fourneau@uvsq.fr
[4] LACL, Paris Est-Créteil University, France
nihal.pekergin@u-pec.fr

Abstract. We present an extension of a methodology based on monotonicity of various networking elements and measurements performed on real networks. Assuming the stationarity of flows, we obtain histograms (distributions) for the arrivals. Unfortunately, these distributions have a large number of values and the numerical analysis is extremely time-consuming. Using the stochastic bounds and the monotonicity of the networking elements, we show how we can obtain, in a very efficient manner, guarantees on performance measures. Here, we present two extensions: the merge element which combine several flows into one, and some Active Queue Management (AQM) mechanisms. This extension allows to study networks with a feed-forward topology.

Keywords: Performance evaluation, histograms, stochastic bounds, queue management

1 Introduction

Measurements are now quite common in networks. But they are relatively difficult to use for performance modeling in an efficient manner. Indeed, the measurements for traffics are extremely huge and this precludes to use them directly in a model. Of course it is still possible to use traces in a simulation, but this is not really an abstract model and we want to be very fast when we solve models and this is not possible with simulations.

One possible solution consists in fitting a complex stochastic process (such as a PH process or a Cox process [8]) from the experimental data and use this parametrized process in a queueing theory model. Here we advocate another solution: the histogram based models. We propose to combine this type of models with stochastic ordering theory to obtain performance guarantees in an efficient manner. Such an approach provides a trade-off between the accuracy of the results and the time complexity of the computations. In the last nine years, Hernández et al. [5–7] have proposed a new performance analysis to obtain buffer occupancy histograms. This new stochastic process called HBSP (Histogram Based Stochastic Process) works directly with small histograms using a set of specific operators on discrete time. The time interval is denoted as a slot. The input traffic is obtained by a heuristic from real traces and it is modeled by a discrete distribution. The arrivals during one time slot are supposed to be identically independently distributed (i.i.d.). The service is supposed to be deterministic, corresponding to the traffic capacity of the link. The buffer is supposed to be finite. Thus, the theoretical model is a $Batch/D/1/K$ queue. In their papers, Hernández et al. do not use the Markovian framework associated with the queue and they develop a numerical algorithm based on the convolution of the distributions. As they named their approach "Histograms", we use the same terminology here. We sometimes write "discrete distributions", which is a more proper term. In this paper, these terms and probability mass function (pmf) are used interchangeably. The analysis proposed by Hernández et al. is only applied to one node because they do not derive properties for the output process of the node. Another problem is that the convergence of their numerical algorithm is not proved. Finally, they use an heuristic to construct reduced histograms from the traces. This is extremely important because their method is fast, but it does not give any guarantees on the results. More precisely, they proceed as follows: they assume the stationarity of the arrivals. Thus, they obtain from the trace, a histogram for the distribution of the number of arrivals during one time slot. But the size of the histogram is too large for a numerical algorithm based on convolution operations. Therefore, they simplify the histogram dividing the space into n sub-intervals (n is a small number) to obtain only n bins (or states) in the histograms. With this method they obtain approximate solutions which can be computed efficiently, if n is small. But there is no guarantee on the quality or the accuracy of the approximations.

To illustrate the approach, we present now a trace used by Hernández et al. and in this work. Figure 1 shows a plot of MAWI traffic trace [11] corresponding to a 1-hour trace of IP traffic of a 150 Mb/s transpacific line (samplepoint-F) for the 9th of January 2007 between 12:00 and 13:00. This traffic trace has an average rate of 109 Mb/s. Using a sampling period of T = 40 ms (25 samples

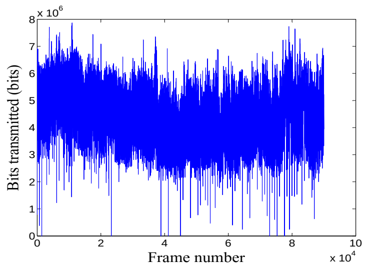

Fig. 1. MAWI traffic trace

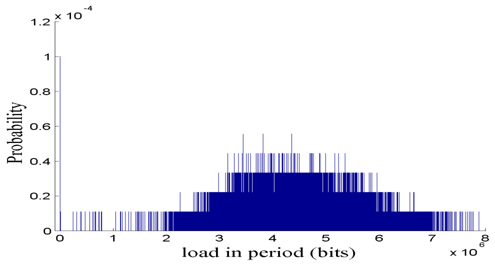

Fig. 2. MAWI arrival load histogram

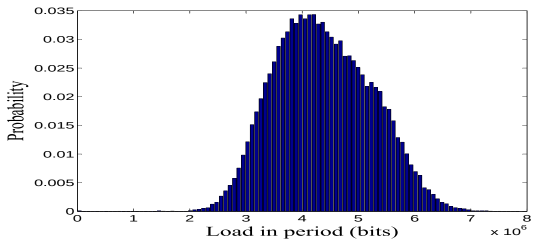

Fig. 3. HBSP approximation of MAWI arrival load histogram with bins=100.

per second), the resulting traffic trace has 90,000 frames (periods) and an average rate of 4.37 Mb per frame, the corresponding histogram is given in Figure 2. The number of bins in this histograms is 80511. Finally, the HBSP approximation with 100 bins is given in Figure 3. The key idea here is the reduction of the number of bins from 80511 bins in the trace to only 100 bins to have the fast numerical analysis.

For our approach, we propose to apply the stochastic bounding method to the histogram based models [3, 2]. The goal is to generate bounding histograms with smaller sizes which can be used to analyze queueing elements with some guarantees on the results. We use the strong stochastic ordering (denoted by $\leq_{st}$) [9]. We have proposed to use the algorithm developed in [4] to obtain optimal lower and upper stochastic bounds of the input histogram. This algorithm allows to control the size of the model and it computes the most accurate bound with respect to a given reward function. The bounding histograms are then used in the state evolution equations to derive bounds for performance measures for a single queue.

An extension of our approach to a queueing network was also investigated. A queueing network is a set of interconnected queues where the departures from one (or more) queue enter one (or more) other queue, according to a specified routing, or leave the system. Here, we focus on queueing networks with finite capacity. We have decomposed the network nodes into: Traffic sources (input flows), Finite capacity queues, Merge elements and Splitters. Monotonicity of networking elements is the key property for our methodology (the formal definition will be given in the paper). In [2] we have proved that some splitters which divide a flow into several sub-flows routing to distinct nodes are also monotone. Therefore, we have generalized the method to networks with a tree topology.

In this paper, we further generalize our methodology in two directions. First, we prove that the merge elements which combine several flows into a global one is also monotone. This first result allows to consider feed-forward networks (i.e. the graph of the networking elements and the links is a Directed Acyclic Graph (DAG)). We use a decomposition approach based on the network topology and the monotonicity allows to obtain approximate results faster than the traditional approach. We remind that the decomposition approach allows us to decompose the network and to study the networking elements in a sequential and greedy manner following the topological ordering associated with the

DAG. This approach gives approximations on performance measures. The use of our methodology in this case aims to accelerate the computational times of this approach with a similar accuracy. Secondly, we study some Active Queue Management mechanisms to extend the modeling applicability of our method.

The technical part of the paper is organized as follows: in Section 2, we describe our methodology: the stochastic comparison of histograms, the reduction of the histogram sizes, the basic queueing model, and the monotonicity. In Section 3, we introduce the routing elements: splitter and merge and we prove that they are monotone. Section 4 is devoted to the AQM mechanisms. Finally in Section 5, we give numerical results for a single node analysis (to compare with HBSP algorithm), and a feed forward network.

2 Methodology for bounds and performances

We briefly introduce a well known ordering, called "strong stochastic ordering", for comparing distributions on $\mathbb{R}$. We show how one can compute the optimal lower bound and upper bound of a given size. The optimality criterion is the expectation of an arbitrary positive and increasing reward chosen by the modeler. We first define the stochastic comparison.

2.1 Stochastic bounds

We refer to Stoyan's book [9] for theoretical issues of the stochastic comparison method. We consider state space $\mathcal{G} = \{1, 2, \ldots, n\}$ endowed with a total order denoted as $\leq$. Let X and Y be two discrete random variables taking values on $\mathcal{G}$, with probability mass functions (pmf in the following) $d2$ and $d1$.

Definition 1. *We can define the strong stochastic ordering by non decreasing functions or by some inequalities involving pmf.*

- **generic definition:** $X \leq_{st} Y \Longleftrightarrow \mathbb{E}f(X) \leq \mathbb{E}f(Y)$, *for all non decreasing functions* $f : \mathcal{G} \to \mathbb{R}^+$ *whenever expectations exist.*
- **probability mass functions**

$$X \leq_{st} Y \Leftrightarrow \forall i,\ 1 \leq i \leq n,\ \sum_{k=i}^{n} d2(k) \leq \sum_{k=i}^{n} d1(k) \tag{1}$$

Note that we use interchangeably $X \leq_{st} Y$ *and* $d2 \leq_{st} d1$*.*

In order to reduce the computation complexity for computing the steady-state distribution, we propose to decrease the number of bins in the histogram. We apply a bounding approach rather than an approximation. Unlike approximation, the bounds allow us to check if QoS requirements are satisfied or not.

More formally, for a given distribution d, defined as a histogram with N bins, we build two bounding distributions $d1$ and $d2$ defined on $n < N$ bins such that $d2 \leq_{st} d \leq_{st} d1$. Moreover, $d1$ and $d2$ are constructed to be the closest distributions with n bins with respect to a given reward function chosen by the modeler. Note that this optimality is not necessary in our approach, but it helps to obtain tight bounds. In [4], three algorithms to construct reduced size bounding distributions have been presented: an optimal algorithm based on dynamic programming with complexity $O(N^2 n)$, a greedy algorithm [4] with complexity $O(NlogN)$ and a linear complexity algorithm. There is no optimality for the last two ones but they are faster. The modeler can use any of them and this gives him the ability to choose between the accuracy and the computation times. In the numerical experiments, we give only results for the optimal one. We emphasize that the important property we need is the construction of a stochastic bound of the experimental distribution extracted from the trace.

In order to get an idea of our stochastic bounding approach, we propose to give an example. We consider the histogram associated to the MAWI traffic trace (see Figure 2) which is defined on 80511 states and we propose to derive bounding distributions $d1$ (stochastic upper bound distribution) and $d2$ (stochastic lower bound distribution) which are defined on reduced size of state space i.e. on $n = 10$ states. For reward function equal to the identity, we computed bounding distributions using the optimal algorithm present in [4]. The cumulative distributions of the different computed histograms are presented in the following figure.

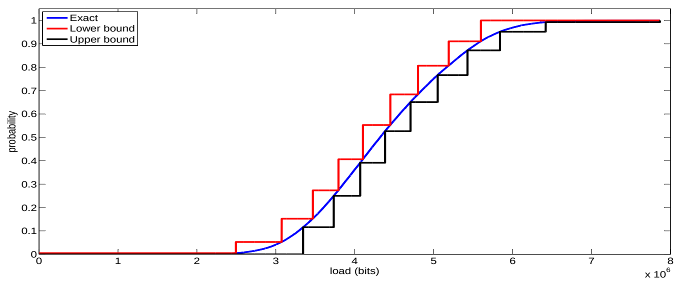

Fig. 4. Cumulative probability distribution (cdf) of computed histograms under MAWI traffic trace.

From this figure, we can clearly observe that our approach allows to define bounds and a coverage of the exact distribution. Moreover, these bounds defined on a reduced state space (i.e. 10 hops instead of 80511 hops) allow to reduce the complexity of numerical computations.

An important practical point is the unit we consider in the model. The traces are measured in bits. To keep the model size reasonable, we convert the values in data units. A data unit is D bits. Typically for the numerical analysis we present here, $D = 1000\,bits$. All the bins in the histograms representing the amount of data are integer multiple of D.

2.2 Stochastic Monotonicity of Networking Elements

The basic networking element is a finite queue associated with one server, a scheduling discipline and an access control. Let B be the buffer size. We assume that the queueing discipline is FCFS and work-conserving. The system evolves in discrete time. The service capacity (the number of data units that can be served during a slot) is constant denoted by C. During each slot, the events occur in this order: arrivals and then service. The buffer-length evolution in the queue is given by a time-homogenous Discrete Time Markov Chain (DTMC) $\{X_n,\, n \geq 0\}$ taking values in a totally ordered state space, $\mathcal{F} = \{0,\, 1,\, 2, \ldots,\, B\}$. The number of data units received during a time slot is independently identically distributed (i.i.d.) random variable A specified by distribution H_1. Therefore, the *evolution equation* of the networking element with finite queue operating with Tail Drop policy [8] is:

$$X_{n+1} = \mathtt{min}(B,\, (X_n + A_n - C)^+), \tag{2}$$

where operator $(X)^+ = \mathtt{max}(X,\, 0)$.

The output of the analysis will be the buffer occupancy denoted by H_3 and defined on state space $\{0 \cdots B\}$ and the departure process given by histogram flow H_5 defined on state space $\{0 \cdots C\}$. For a histogram H, we denote by E^H the set of states. For simplicity H will be considered as a probability vector corresponding to the probabilities for the ordered elements of the set of states E^H.

We now give the main results of [2] about the stochastic monotonicity of the elements. All the proofs are omitted here. At each queuing element, the analysis consists in computing the distributions of H_3 and H_5 or bounds of these distributions knowing the input arrival distribution H_1. For a splitter and a merge node, the analysis consists in computing the output distributions knowing the input distributions, the parameters and the service discipline.

Proposition 1 (Buffer Occupancy distribution, H_3). *The queue-length distribution before the instant of arrivals corresponds to steady state distribution π of the Markov chain.*

Let distribution H_q denote the convolution of distributions H_1 and H_3:

$$H_q = H_3 \otimes H_1.$$

Proposition 2 (Batch Departure, H_5). *The departure histogram H_5 is computed as follows:*

$$\begin{cases} H_5(w) = \quad H_q(w), & \text{if } \; w < C; \\ H_5(C) = \sum_{w \geq C} H_q(w), & \text{otherwise.} \end{cases}$$

Proposition 3 (Losses, H_L). *The distribution of losses under the Tail Drop policy is:*

$$\begin{cases} H_L(k - \mathrm{B}) = \quad H_q(k), & \text{if } \; k > \mathrm{B} + \mathrm{C}; \\ H_L(0) = \sum_{k \leq \mathbf{B}+\mathbf{C}} H_q(k), & \text{otherwise.} \end{cases}$$

Then, the loss probability P_L can be defined as follow: $P_L = \frac{\mathbb{E}[H_L]}{\mathbb{E}[H_1]}$.
We have proved the monotonicity properties of a queue in [2] and we give them without proof.

Definition 2. *A finite capacity queue is H-monotone if the following property holds: If $H_1^a \leq_{st} H_1^b$, then $H_3^a \leq_{st} H_3^b$; $H_5^a \leq_{st} H_5^b$ and $H_L^a \leq_{st} H_L^b$.*

Theorem 1. *A finite capacity queue which is operating with work-conserving FCFS service policy and Tail Drop policy is H-monotone.*

3 Analysis of a network with a DAG topology

In this section, we study network operations involving multiple streams as in [10]. First, we consider the split operation which has already been partially presented in [2]. Then, we introduce the merge operations. We note that the splitters (resp. merge elements) do not have either processing element or queue to store data units. The routing decision in these elements is supposed to be instantaneous.

3.1 Splitter

When the input flow modeled by a distribution H_S crosses a splitter, it is divided into m flows: $H_{S,1}, \ldots, H_{S,m}$. We assume that the batches observed after the splitter are still i.i.d. for each flow. This precludes the representation of Round Robin mechanism which may introduce the non stationarity in the flows.
We define for the split operator the H-monotonicity as follows:

Definition 3. *A splitter is said to be H-monotone, iff*

$$H_S^a \leq_{st} H_S^b \Rightarrow \forall i,\ H_{S,i}^a \leq_{st} H_{S,i}^b.$$

We study two cases of splitter:

- each batch arriving at the splitter is sent completely to one of the output flows. The output is randomly chosen according to a routing probability. This was previously presented in [2].
- the batch is divided into all the outputs according to a distribution for the repartition of the data units. This part is studied in this current paper.

Model of a routing probability. We study a split where all the data units of a batch arriving in the input flow are routed to an output flow with a routing probability. Let $p_i, 1 \leq i \leq m$ (such that $\sum_{i=1}^m p_i = 1$), be the routing probability of the batch to the output flow i of the split. If the set of states of H_S does not include 0, it will be added with probability 0, and the set of states for output flows will be the same as E^{H_S}.

$$E^{H_{S,i}} = \{0\} \cup E^{H_S}, \quad 1 \leq i \leq m.$$

The probability distribution of any output flow i can be computed as follows: $1 \leq \forall i \leq m, \quad H_{S,i}(k) = p_i\ H_S(k), \quad k > 0;$ and $H_{S,i}(0) = 1 - \sum_{k \neq 0} H_{S,i}(k)$.

Example 1. We consider the distribution H given by the set of states, $E^H = \{0, 3, 4, 7, 10\}$ and the corresponding probability vector $H = [0.1, 0.2, 0.4, 0.1, 0.2]$. Assume that the batch is routed on two directions with equal probability. Each of the routed batch has the following distribution: the values are $E^{H_i} = \{0, 3, 4, 7, 10\}$ and the probabilities are $H_i = [0.55, 0.1, 0.2, 0.05, 0.1]$, where $1 \leq i \leq 2$.

For an efficient implementation of histograms, the set of states are constituted of the elements with non null probabilities. However, in the sequel for the proofs, we assume that the histograms are defined on set of states $E^H = \{0, \cdots n\}$ thus, the probability vectors may contain null probabilities.

Theorem 2. *If the batch is routed entirely to a flow according to routing probabilities, then the split is H-monotone.*

Proof: For each flow $i, 1 \leq i \leq m$, we have the following equations:

$$1 \leq \forall l \leq n, \quad \sum_{k=l}^{n} H_{S,i}^{a}(k) = \sum_{k=l}^{n} p_i \ H_S^a(k), \ \ \sum_{k=l}^{n} H_{S,i}^{b}(k) = \sum_{k=l}^{n} p_i \ H_S^b(k).$$

As $H_S^a \leq_{st} H_S^b$, we have $\sum_{k=l}^{n} H_S^a(k) \leq \sum_{k=l}^{n} H_S^b(k)$. Thus, for each flow $i, 1 \leq i \leq m$: $\sum_{k=l}^{n} p_i H_S^a(k) \leq \sum_{k=l}^{n} p_i \ H_S^b(k)$. Which can be also written: $\sum_{k=l}^{n} H_{S,i}^a(k) \leq \sum_{k=l}^{n} H_{S,i}^b(k)$. Which is equivalent to $H_{S,i}^a \leq_{st} H_{S,i}^b$.

Model of a division of the batch and dispacthing among the links. We now assume that the data units are distributed among the m flows. The proportion of data received by each flow is given by the probability p_i which must be understood now as a ratio. Due to this multiplication by p_i, this amount of data can be a non integer amount of data units. Then, we assume that the data units are added with null bits and we obtain an integer number of data units.

$$E^{H_{S,i}} = \{k \mid \mathbb{1}_{\lceil p_i * q \rceil = k}\}_{q \in E^{H_S}}.$$

The probability distribution of any output flow i can be computed as follows: $1 \leq \forall i \leq m, \quad H_{S,i}(k) = \sum_{q \in E^{H_S}, q \neq 0} H_S(q) \mathbb{1}_{\lceil p_i * q \rceil = k}, \ \ \forall k > 0,$ and $H_{S,i}(0) = 1 - \sum_{k \neq 0} H_{S,i}(k)$.

Example 2. Consider the same example, but assume now that the data units are distributed among the flows. We also assume an equal repartition of the flows. The resulting batches have the same distribution: the values are $E^{H_i} = \{0, 2, 4, 5\}$ and the probabilities are $H_i = [0.1, 0.6, 0.1, 0.2]$. Indeed, the probability that the batch size is 2 is the sum of the probability that the initial batch size (before division into two equal batches) was 3 or 4.

Theorem 3. *If the batch is splitted into batches according to dispatching probabilities, then the split is H-monotone.*

Proof: For each flow $i, 1 \leq i \leq m$, we have the following equations:

$$1 \leq \forall\, l \leq n, \qquad \sum_{k=l}^{n} H^a_{S,i}(k) = \sum_{k=l}^{n}\sum_{q=0}^{n} H^a_S(q)\mathbb{1}_{\lceil p_i * q \rceil = k}.$$

After exchanging the summations: $\sum_{k=l}^{n} H^a_{S,i}(k) = \sum_{q=0}^{n} H^a_S(q) \sum_{k=l}^{n} \mathbb{1}_{\lceil p_i*q\rceil = k}$ But $\sum_{k=l}^{n} \mathbb{1}_{\lceil p_i*q\rceil=k} = \mathbb{1}_{q \geq Q_i}$, for some Q_i. Thus, $\sum_{k=l}^{n} H^a_{S,i}(k) = \sum_{q=0}^{n} H^a_S(q)\, \mathbb{1}_{q\geq Q_i} = \sum_{q\geq Q_i} H^a_S(q)$. Since $H^a_S \leq_{st} H^b_S$, due to the st-ordering we get: $\sum_{q\geq Q_i} H^a_S(q) \leq \sum_{q\geq Q_i} H^b_S(q)$. Therefore, $\sum_{k=l}^{n} H^a_{S,i}(k) \leq \sum_{k=l}^{n} H^b_{S,i}(k)$. Thus for all i, $H^a_{S,i} \leq_{st} H^b_{S,i}$.

3.2 Merge

In a merge element, a set of independent flows with distributions $H_{M,i}, 1 \leq i \leq m$ are aggregated to a flow with distribution H_M. We suppose that the links have a finite capacity, where C_i is the capacity of link i. In this subsection, we present the monotonicity properties for the merge elements by means of random variables corresponding to these histograms. Thus, X_i is the random variable with pmf $H_{M,i}$ representing the number of data units of input flow i of the merge element. The merge can be defined as a function as follows:

Definition 4. *A merge is a function* $\mathfrak{m} : \times_{i=1}^{m}\{0,\ldots,C_i\} \to \{0,\ldots,C\}$ *(i.e. the full convolution of m distributions).* $\mathfrak{m}(X_1,\ldots,X_m)$ *represents the state of the output flow of the merge element under independent input flows* X_i*. In fact it is a random variable with pmf* H_M *representing the number of data units leaving the merge element and taking values in* $\{0, 1, \cdots, C\}$ *where* $C \leq \sum_{i=1}^{m} C_i$.

Obviously, for the merge operation, the number of departed data units must be lower than the number of arrived data units.

Definition 5. *The merge is causal if* $\mathfrak{m}(X_1,\ldots,X_m) \leq \sum_{i=1}^{m} X_i$.

We can also define for a merge element the traffic monotonicity as follows:

Definition 6. *A merge element is traffic monotone iff for all couple* $(X_1,\ldots,X_m)$ *and* $(Y_1,\ldots,Y_m)$*, if* $X_k \leq Y_k$*,* $\forall k$*, then* $\mathfrak{m}(X_1,\ldots,X_m) \leq \mathfrak{m}(Y_1,\ldots,Y_m)$.

In the sequel, we consider causal merge elements. The merge operation may have the Tail Drop property which is defined as follows:

Definition 7. *A merge element is said to be Tail Drop iff* $\mathfrak{m}(X_1,\ldots,X_m) = \min(C, \sum_{i=1}^{m} X_i)$.

We study now the monotonicity property of the merge elements.

Definition 8. *A merge element is said to be H-monotone, iff*

$$\forall i,\ H^a_{M,i} \leq_{st} H^b_{M,i} \Rightarrow H^a_M \leq_{st} H^b_M.$$

Theorem 4. *If the merge element is traffic monotone then it is H-monotone.*

Proof: We suppose that $\forall i,\ H^a_{M,i} \leq_{st} H^b_{M,i}$, thus the corresponding random variables are comparable: $\forall i,\ X^a_i \leq_{st} X^b_i$. The traffic monotonicity of the merge element means indeed that the function $\mathfrak{m}$ is an increasing function. Since the output flows H^a_M and H^b_M are defined as increasing functions of comparable independent random variables, they are also comparable (see page 7 of [9]).

Corollary 1. *A merge element operating with Tail Drop (i.e.* $\mathfrak{m}(X_1, \ldots, X_m) = \min(C, \sum_{i=1}^m X_i)$*) is causal and traffic monotone. Therefore, it is H-monotone.*

We now consider loss processes in merge elements. A merge element may delete some data units due to a bandwidth limitation or an access control. First we define the number of data units lost by loss function l which depends on the merge function $\mathfrak{m}$.

Definition 9. *The number of data units lost in a merge element can be defined by a function* $l : \times_{i=1}^m \{0, \ldots, C_i\} \to \{0, \ldots, \sum_{i=1}^m C_i\}$. $l(X_1, \ldots, X_m) = \sum_{i=1}^m X_i - \mathfrak{m}(X_1, \ldots, X_m)$.

Indeed, the number of losses is the difference between the number of data units arrived on the m links (i.e. $\sum_{i=1}^m X_i$) and the number of units accepted by the merge element (i.e. $\mathfrak{m}(X_1, \ldots, X_m)$). The loss distribution can be given as follows, since the arrivals are independent. Let us remark that small letters denote the realizations of the corresponding random variables X_i.

Proposition 4 (Loss Distribution for a merge, H_L).

$$H_L(k) = \sum_{(x_1,\ldots,x_m)} \mathbb{1}_{\sum_{j=1}^m x_j - \mathfrak{m}(x_1,\ldots,x_m) = k} \prod_{i=1}^m H_{M,i}(x_i)$$

Property 1 *If* $C = \sum_i C_i$ *and the merge element is Tail Drop then there are no losses.*

Proof: The element is Tail Drop then, $\mathfrak{m}(X_1, \ldots, X_m) = \min(C, \sum_{i=1}^m X_i)$. But by construction $x_i \leq C_i$. Therefore $\sum_{i=1}^m X_i \leq \sum_{i=1}^m C_i = C$. Thus, there are no losses at the merge element.

Theorem 5. *If the loss function l is increasing, then the distribution of losses in a merge element is monotone: if* $\forall i,\ H^a_{M,i} \leq_{st} H^b_{M,i}$*, then* $H^a_L \leq_{st} H^b_L$.

Proof : The proof is similar to that of Theorem 4. Since H^a_L and H^b_L are defined as increasing functions of comparable input flows, they are comparable.

Property 2 *We consider a Tail Drop, merge element with output capacity C. If* $C < \sum_i C_i$*, the distribution of losses is monotone.*

Proof : The number of data units lost is $l(X_1, \cdots X_m) = \max(0, \sum_{\mathrm{i=1}}^{\mathrm{m}} \mathrm{X_i} - \mathrm{C})$. The monotonicity of the number of data units lost follows from the fact that function l is increasing.

4 Analysis of some AQM mechanisms

The queue presented in section 2 is operated under Tail Drop policy, which is a particular case of AQM (Active Queue Management). Indeed, the data units are accepted in the queue until the queue is full. In this section, we also present some conditions for AQM to be H-monotone in order to derive performance measure bounds. We illustrate this approach with a Random Early Detection mechanism (RED in that follows).

We restrict ourselves to some AQMs where the probabilities of rejection depend on the size of the queue just before the insertion.

Definition 10. *The AQM is immediate if it operates independently and sequentially for each data unit in the batch and if the probabilities of rejection take into account the state of the queue just before the insertion.*

Note that this is a restricted version of AQM. We do not represent some mechanisms like explicit congestion notification. And, in mechanisms like RED, one does not use the instantaneous queue size to compute the acceptation probability, but a moving average of the queue size. Thus, our definition is rather abstracted, but it can be used as a limit or an approximation.

More formally, we define an AQM acceptation by a function $q(X)$ which equals to 1, if the data unit is accepted and 0 if the data unit is rejected when the buffer size is X.

Definition 11. *The AQM is decreasing if function $q(X)$ is not increasing.*

Example 3. The Tail Drop policy used so far is described by the acceptation function: $q(X) = \mathbb{1}_{\{X<B\}}$.

Thus, Tail Drop at the packet level is clearly immediate and decreasing.

Definition 12 (IRED). *The Immediate Random Early Detection policy is an example of AQM. We assume that it operates at data unit level. Contrary to Tail Drop, the acceptation for RED is given with probabilities. Many RED implementations are based on cubic functions or on the following piece-wise linear function to compute the acceptation probabilities:*

- *if* $X \leq \frac{B+C}{2}$: $Prob(q(X)=1) = 1$;
- *if* $\frac{B+C}{2} \leq X < B+C$: $Prob(q(X)=1) = \frac{2(B+C)-2X}{B+C}$;
- *if* $X \geq (B+C)$: $Prob(q(X)=1) = 0$;

Thus, the probability that $q(X) = 1$ decreases with the queue length, X.

We extend to network elements with an AQM the definition for H-monotonicity and we prove some conditions on AQM to be H-monotone.

Definition 13. *The AQM is H-monotone, iff*

$$H_1^a \leq_{st} H_1^b \Rightarrow H_3^a \leq_{st} H_3^b \text{ and } H_L^a \leq_{st} H_L^b$$

We suppose that the queue works with an immediate AQM specified with a decreasing admission function $q(X)$. We denote by X_n the length of the queue at slot n and by $Y_{n,j}$ the length of the queue at slot n after the admission of the jth data unit. We take the same assumptions for the parameters as in the analysis of a queue (Section 2.2), and the maximum arrival batch is denoted by K. The evolution equation of the queue length can be given as follows in the case when arrivals are taken into account before the services.

$$\begin{cases} Y_{n+1,0} &= X_n; \\ Y_{n+1,j+1} &= Y_{n+1,j} + \mathbb{1}_{\{A_n > j \text{ and } q(Y_{n+1,j})=1\}}; \\ X_{n+1} &= (Y_{n+1,K} - C)^+. \end{cases}$$

Theorem 6. *If the AQM is immediate and the acceptation function is decreasing, then the AQM is H-monotone.*

Proof: The proof is based on the sample-path property of the strong stochastic ordering [9]. We prove by induction that the existence of the realizations of the random variables for the evolution of queue length (see Equation 2) satisfy:

$$x_n^a \leq x_n^b, \quad \forall n.$$

We assume that queue lengths are the same for slot 0 and we give the general case for the induction with n. Let suppose that $x_n^a \leq x_n^b$ then by definition $y_{n+1,0}^a \leq y_{n+1,0}^b$. We will prove that if $y_{n+1,j}^a \leq y_{n+1,j}^b$ then $y_{n+1,j+1}^a \leq y_{n+1,j+1}^b$. There are two cases:

1. if $y_{n+1,j}^a < y_{n+1,j}^b$, since the increasing is one by one we are sure that: $y_{n+1,j+1}^a \leq y_{n+1,j+1}^b$.
2. if $y_{n+1,j}^a = y_{n+1,j}^b$, then $q(y_{n+1,j}^a) = q(y_{n+1,j}^b)$, and it follows from the hypothesis that $H_1^a \leq_{st} H_1^b$ then $\mathbb{1}_{A_n^a > j} \leq \mathbb{1}_{A_n^b > j}$. Thus, $y_{n+1,j+1}^a \leq y_{n+1,j+1}^b$.

So, we deduce that: $x_{n+1}^a = y_{n+1,K}^a \leq y_{n+1,K}^b = x_{n+1}^b$. Therefore, we have the stochastic comparison of the queue length evolutions: $X_n^a \leq_{st} X_n^b, \ \forall n$ and for the stationary process: $H_3^a \leq_{st} H_3^b$.
The number of data units lost during slot $n+1$ can be given as:

$$\sum_{j=1}^{K} \mathbb{1}_{\{A_n > j \text{ and } q(Y_{n+1,j})=0\}}.$$

It follows from the above proof that $Y_{n,j}^a \leq_{st} Y_{n,j}^b$. Since the acceptation functions $q()$ are decreasing functions, and $H_1^a \leq_{st} H_1^b$, if the above indicator function is 1 under H_1^a then it is also 1 under H_1^b. Thus, the number of data units lost in each slot and in the limit will be comparable: $H_L^a \leq_{st} H_L^b$.

5 Examples

We consider respectively a node with an IRED mechanism and a network of nodes. For all the experiments, we suppose that the monotonicity property is used for the convergence proof of our method [2] for $\epsilon = 10^{-6}$. the reward function used here is defined by $r(i) = i, \forall\, i \in E^H$. We note that the implementation is performed on Matlab and the experiences were computed on a laptop computer Intel Core I7, 2.53GHz.

5.1 A RED node

We give a simple example to illustrate the impact of our method on single node with IRED mechanism. We consider input histogram $H_1 = [0.10, 0.05, 0.10, 0.10, 0.15, 0.15, 0.10, 0.10, 0.05, 0.10]$ defined on state space $E^{H_1} = \{1, \ldots, 10\}$ and deterministic service $C = 2$. The performance measures (blocking probabilities, average queue length and execution time) are calculated by varying the buffer size from 4 to 30 data units. In Figures 5, 6 and 7, we present the performance measures by using the exact computation and optimal lower bound for bins equal to 3 and 5. In this example we illustrate the lower bounds but the upper bounds can also be calculated.

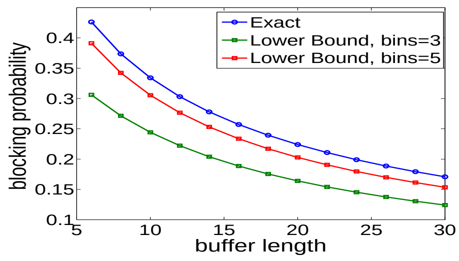

Fig. 5. Results on blocking probabilities.

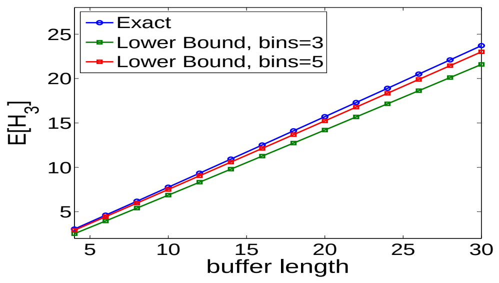

Fig. 6. Results on mean buffer length.

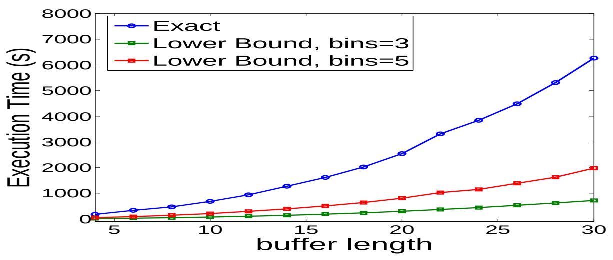

Fig. 7. Execution time (s).

Through these figures, we see that the use of bounding method allows us to obtain accurate results with in reduced execution time. We remark that when the number of bins increases the accuracy of the lower bound is improved.

5.2 A Feed-Forward Network

Unlike HBSP method, our approach can be extended to the study of feed-forward networks as shown in the following example.

We consider a feed-forward network model depicted in Figure 8 with 6 nodes. We take MAWI traffic trace as input arrival histogram (H_1). Each node is a split (resp. merge) element or a finite capacity queue ($\mathrm{B_i} = 10\,\mathrm{Mb}$, $i = 1, 3, 4, 6$). The service for each queue is taken respectively equal to $110\,Mb/s$, $67.5\,Mb/s$, $90\,Mb/s$ and $117.5\,Mb/s$.

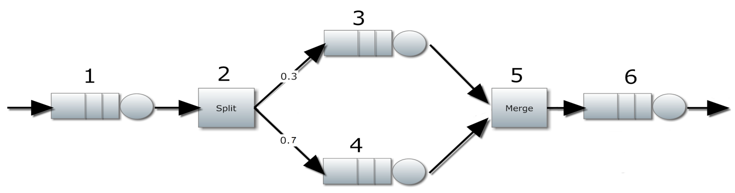

Fig. 8. An example of Feed Forward Network.

Based on the decomposition approach, we compute the performance measures of interest under MAWI real traffic traces (Figure 1) by considering respectively: the whole input distribution (MAWI histogram without reduction) and our stochastic bounding histograms. For this example, we are interested in the queue length distribution (H_3), departure batch distribution (H_5) and loss probabilities (P_L).

		$\mathbb{E}[H_1]$	$\mathbb{E}[H_3]$	$\mathbb{E}[H_5]$	P_L
Queue 1	O. input	4375620	4332130	4352390	0.00530
	$L.b$	4356310	3850300	4339640	0.00382
	$U.b$	4397080	4890670	4364590	0.00739
Queue 2	O. input	1305720	875834	1305030	0.00052
	$L.b$	1300650	863705	1300010	0.00049
	$U.b$	1309460	884739	1308730	0.00055
Queue 3	O. input	3046670	2256310	3037710	0.00293
	$L.b$	3034840	2190650	3026910	0.00260
	$U.b$	3055400	2304630	3045640	0.00318
Queue 4	O. input	4342730	2519100	4327660	0.00346
	$L.b$	4313200	2340670	4301450	0.00272
	$U.b$	4357650	2593470	4341260	0.00375

Table 1. Results for bins=100.

		$\mathbb{E}[H_1]$	$\mathbb{E}[H_3]$	$\mathbb{E}[H_5]$	P_L
Queue 1	O. input	4375620	4332130	4352390	0.00530
	$L.b$	4366450	4099970	4346550	0.00455
	$U.b$	4386860	4622920	4359020	0.00633
Queue 2	O. input	1305720	875834	1305030	0.00052
	$L.b$	1302780	868692	1302120	0.00050
	$U.b$	1307780	880729	1307060	0.00054
Queue 3	O. input	3046670	2256310	3037710	0.00293
	$L.b$	3039820	2217560	3031480	0.00273
	$U.b$	3051480	2282840	3042080	0.00306
Queue 4	O. input	4342730	2519100	4327660	0.00346
	$L.b$	4322810	2398430	4310020	0.00295
	$U.b$	4350010	2555410	4334300	0.00360

Table 2. Results for bins=200.

In Table 1 (resp. Table 2), we give for the four queues of the network, the results obtained when we consider the original input histogram (denoted by O. input) and those computed using our stochastic bounds (denoted by $L.b$ for lower bound and $U.b$ for upper bound) for the number of bins equal to 100 (resp. 200).

From these tables, we remark that the bounds on the results are provided for each intermediate stage (due to the H-monotonicity of the network elements). We can also see that the results provided by our bounds are very accurate, and become very close to the solution obtained with original input histogram, when the number of bins increases. For bins equal to 100 (resp. 200), the execution times of the bounds takes respectively 14.4 s (resp. 22.1 s) for the lower bound and 15.9 s (resp. 25.9 s) for the upper bound, where the resolution of the network using the original input is obtained after longer than three days 314248 s. We can therefore conclude that if we want to use the decomposition approach for DAG network analysis and obtain approximations on performance measures, we can use the proposed method and compute similar results with a relatively small computation complexity.

6 Conclusions

The results developed in this paper are very promising: they allow to mix in an efficient and accurate manner measurements and stochastic modeling to analyse some networks (simple queue, AQM and DAG networks via decomposition approach). As future works, we want to extend our methodology and state some stochastic comparison results in feed-forward networks [1] (and also general topology networks). Note that the approach is not limited to performance evaluation of networks, it can be applied to any problem (reliability, statistical model checking) where we have large measurements and where the model is monotone in some sense.

Acknowledgement: this work was partially supported by grant ANR MARMOTE (ANR-12-MONU-0019) and DIGITEO.

References

1. F. Aït-Salaht, H. Castel Taleb, J. M. Fourneau, T. Mautor, and N. Pekergin. Smoothing the input process in a batch queue. In *IEEE ISCIS 2015*, pages 223–232. Springer, 2015.
2. F. Aït-Salaht, H. Castel Taleb, J. M. Fourneau, and N. Pekergin. A bounding histogram approach for network performance analysis. In *In HPCC, China*, 2013.
3. F. Aït-Salaht, H. Castel Taleb, J. M. Fourneau, and N. Pekergin. Stochastic bounds and histograms for network performance analysis. In *EPEW, Italy, Sept. 2013*, LNCS. Springer, 2013.
4. F. Aït-Salaht, J. Cohen, H. Castel-Taleb, J M. Fourneau, and N. Pekergin. Accuracy vs. complexity: the stochastic bound approach. In *11th International Workshop on Disrete Event Systems*, pages 343–348, 2012.
5. E. Hernández-Orallo and J. Vila-Carbó. Network performance analysis based on histogram workload models. In *MASCOTS*, pages 209–216, 2007.
6. E. Hernández-Orallo and J. Vila-Carbó. Web server performance analysis using histogram workload models. *Computer Networks*, 53(15):2727–2739, 2009.
7. E. Hernández-Orallo and J. Vila-Carbó. Network queue and loss analysis using histogram-based traffic models. *Computer Communications*, 33(2):190–201, 2010.
8. L. Kleinrock. *Queueing Systems*, volume I: Theory. Wiley Interscience, 1975.
9. A. Muller and D. Stoyan. *Comparison Methods for Stochastic Models and Risks.* Wiley, New York, NY, 2002.
10. M. Schleyer. *Discrete Time Analysis of Batch Processes in Material Flow Systems.* Wissenschaftliche Berichte des Institutes für Fördertechnik und Logistiksysteme des Karlsruher Instituts für Technologie. Univ.-Verlag Karlsruhe, 2007.
11. K. Cho Sony and K. Cho. Traffic data repository at the wide project. In *In Proceedings of USENIX 2000 Annual Technical Conference: FREENIX Track*, pages 263–270, 2000.