RAIRO. RECHERCHE OPÉRATIONNELLE

C. LABBÉ
F. REBLEWSKI
J.-M. VINCENT

Performance evaluation of high speed network protocols by emulation on a versatile architecture

RAIRO. Recherche opérationnelle, tome 32, n° 3 (1998), p. 253-270

<http://www.numdam.org/item?id=RO_1998__32_3_253_0>

Recherche opérationnelle/Operations Research
(vol. 32, n° 3, 1998, pp. 253-270)

PERFORMANCE EVALUATION OF HIGH SPEED NETWORK PROTOCOLS BY EMULATION ON A VERSATILE ARCHITECTURE (*)

by C. LABBÉ ([1]), F. REBLEWSKI ([2]) and J.-M. VINCENT ([3])

Abstract. – *The goal of this study is to validate a new simulation technique for performance evaluation of discrete time queuing networks. This technique uses a versatile architecture, which has changed from its primary purpose : circuit emulation in the design flow of chips. It is shown that this technique can be used to highlight rare events, such as realistic packet loss probability in high-speed switched networks.* © Elsevier, Paris

Keywords: Emulation on a Versatile Architecture, performance evaluation, high speed networks, queuing networks, rare events.

Résumé. – *Le but de cette étude est de valider une nouvelle technique de simulation pour l'évaluation de performance des réseaux de files d'attente en temps discret. Cette technique consiste à utiliser une machine à architecture reconfigurable en la détournant de son utilisation première : l'émulation de circuit dans la chaîne de conception de systèmes numériques intégrés. On montre que cette technique peut être utilisée pour mettre en évidence des événements rares, comme des taux de pertes réalistes dans les réseaux haut débit.* © Elsevier, Paris

Mots clés : Émulation sur une architecture reconfigurable, évaluation de performance, réseaux haut débit, réseaux de files d'attente, événements rares.

1. INTRODUCTION

Broadband Integrated Services Digital Networks (B-ISDNs) are intended to provide a variety of different teleservices on a single "universal" network. Moreover, such services can have widely differing Quality of Service (QoS) requirements. At the packet (cell) level, this means differences in permissible

(*) This work has been supported by France Telecom (CNET) and M2000.

([1]) M2000, CNET/DTL/ASR, rue du vieux Chêne, BP 98, 38243 Meylan Cedex, France. Cyril.Labbe@cnet.francetelecom.fr

([2]) M2000, Batiment Azur, 4 rue R. Razel, 91400 Saclay, France.

([3]) Laboratoire LMC-IMAG, Domaine Universitaire, BP 53 X, 38041 Grenoble Cedex, France, Jean-Marc.Vincent@imag.fr

Recherche opérationnelle/Operations Research, 0399-0559/98/03/

cell loss and cell transfer delays. A complete specification of performance in packet switching networks involves two components: packet loss and packet delay. End-to-end delay is an important measure of performance as too much delay is equivalent to loss when considering a real time context. This measure of performance depends directly on the switch architecture. That is why investigation on performance evaluation and dimensioning buffered-switching elements are so important. Furthermore, these research activities have a practical impact on architectures and protocols.

Models used for this research are often discrete time queuing networks. This is especially true in the case of ATM (Asynchronous Transfer Mode), where slotted time is quite natural since all the cells have the same size. A slot is the time needed to serve a cell. A realistic packet loss probability is around 10^{-8}-10^{-9}. Such losses are rare events which are difficult to capture by direct computation. Software simulators are too limited to obtain such a probability. For systems with few tenth of thousand states P-simulation can be used [11]. However it is currently possible to compute analytically realistic theoretical loss probabilities on the first stage of a broadband multiplexer and to obtain bounded results on the second stage [20]. However the computation of realistic loss probability for subsequent stages is still an open problem. This is due to the very large number of states of this problem.

The aim of this paper is to validate a new approach, using emulation on a versatile architecture machine for performance evaluation of finite capacities queuing networks in discrete time. It is shown that this technique could be used to hightlight rare events, such as realistic packet loss probability in high-speed switched networks.

Emulation is a widely used technique in the design flow of chips. Programmable hardware is used to reproduce the functionalities of a circuit. Emulation is performed by an emulator, which can be seen as a hardware simulator, since its hardware configuration can be modified to model other circuits ; this is an "all purpose hardware emulator" based on a versatile architecture [9, 7].

Here we will focus on packet loss probability and examine the traffic perturbation induced by stages of multiplexers. A four-by-four multistage ATM switch [18] (see Figure 1) is used as an example throughout the article. If we assume that K is the capacity of queues and q the number of queues under study. The number of states of the system is $(K+2)^q$. That is to say, with five queues of capacity 20, the system has 5.10^6 states. For example in the section 5 a model with 10^{12} states is treated and a loss probability of 10^{-9} had been reached. This ATM switch is modeled by a queuing network

which is emulated by a dedicated architecture on the versatile machine. The choice of architecture is determined by both the structure of the model and the performance parameters under study. Experimental protocols are conducted to collect statistics on losses and traffic characteristics.

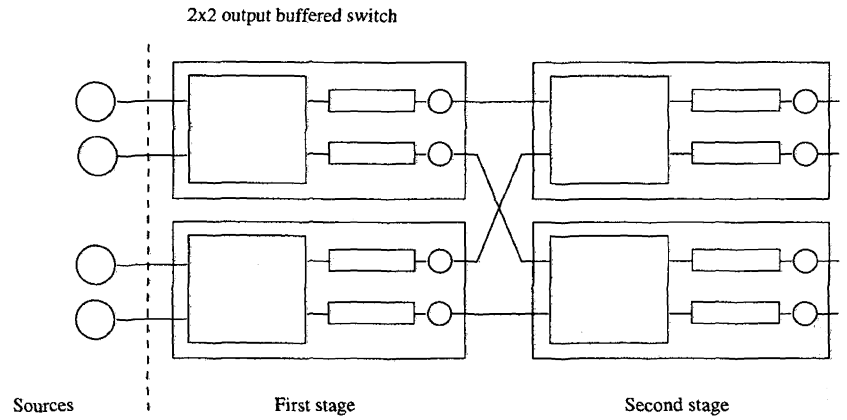

Figure 1. – Four-by-four double stage switch made of two-by-two output buffered switches.

The stucture of the paper is the following. The versatile architecture of the emulator and the software developments used in this study are presented in Section 2. The design method of a queuing network, using this architecture, is discussed in Section 3, using the example of the four-by-four ATM switch. Section 4 presents the *hardware instrumentation* required to obtain information on what happens in the queuing networks. This section also presents the control with respect to statistical assumptions. Section 5 presents a set of experimental results on a four-by-four multistage ATM switch.

2. HARDWARE ARCHITECTURE AND SOFTWARE ENVIRONMENT

This section presents the hardware architecture and the software environment used to emulate queuing networks. The software is used to describe a component modeling the queuing network and the hardware simulator emulates this component.

2.1. Architecture

The hardware simulator is the M500 machine from Metasystems [9, 7], comprising:

- 500,000 programmable logic gates, connected to each other through a programmable network,

- 17 Mbytes of memory, single or double port,
- adjustable clock frequency from 1 to 10 Mhz.

All this hardware can be shaped to emulate any digital and synchronous circuit. The description of a chip is given to the Emulator by configuration files. Memories can be loaded or dumped using ASCII files. The values of the input signals of the emulated component are directly chosen by the user. The clock frequency is 10 Mhz for a very small occupation of the emulator but, under normal conditions, the frequency is usually close to 1 Mhz. The emulator clock is under user control. All signals and register values are available on the last 7000 clock cycles, which is very useful for debugging.

2.2. Software environment

The software flow leads to the files required by the emulator to reproduce the functionalities of a circuit. These functionnalities are described in terms of concurrent processes using the VHDL language. VHDL is an efficient way of obtaining a high level description of a hardware component, which is then translated into gates by the Synopsys synthesis tools. From this representation of the components, the Metasystems compiler produces the data base required by the emulator.

The software flow is presented in Figure 2 and is detailed above:

- a VHDL (VHSIC Hardware Description Language) description of the chip is used to describe the system in terms of concurrent processes [13].
- Synopsys synthesis: this software, provided by Synopsys, translates the VHDL description into combinational logic and registers (logic gates), which are the basic logic elements used in electronic systems [14].
- The Metasystems compiler computes the links for the set of gates available in the machine. This is the routing operation, which results in

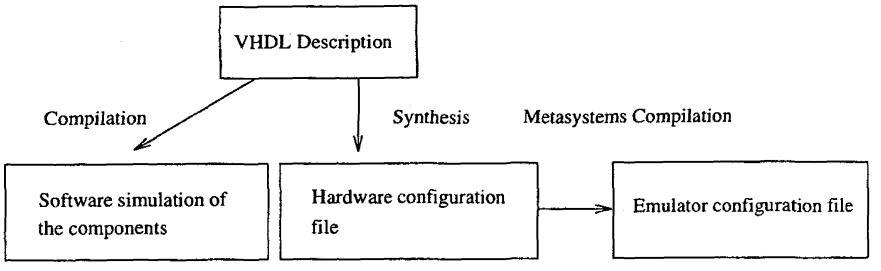

Figure 2. – Software flow used to structure the versatile architecture.

connecting the gates to each other through the programmable network of the emulator.

3. MODELING DISCRETE TIME QUEUES USING THE VHDL HARDWARE DESCRIPTION

Due to this technique one has to rethink the queuing model in hardware design. That is to say, in terms of counters, memories, comparators, state machines... In this section, the method for modeling a queue in synthesizable VHDL is described. Different ways of building a queue are proposed, showing that the hardware representation depends on the adopted model. The simple case of a deterministic server is considered. In the first subsection the updating technique for the number of waiting customers is described. The second subsection presents a traffic supplier to feed a queue and, more often, a queuing network. This is the representation of the arrival process, i.e. generation of hardware sources. It is important to note that an automated translation tool could be implemented. This will allowed to transform the queuing model in a VHDL program, or even in a configuration of the emulator.

3.1. Hardware queue

A hardware representation of a queue depends on various parameters, such as customer or server characteristics.

For example: if customers are all similar, only a register and some "logic glue" are needed (see Figure 3). The register contains the number of waiting customers and is updated according to the number of new customer arrivals, the capacity of the queue and the number of customers served. As this is a discrete time model, there are two options: arrival first (AF) or departure first (DF). What is important to note is that in this case a slot is equivalent to a clock cycle. This is a poor representation since performance parameters, such as mean delay or delay variation are not easily measurable. It is however cheap in terms of hardware occupation, since it requires only register and logic glue.

In high-speed packet switched networks, there may be different levels of priority. In the previous model, the packets are all identical but, in this case, a priority is assigned to each packet. To introduce this into our model, real packets carrying information are used. This information has to be queued in the same way as a real packet. This means that each queue of the network is implemented using a memory corresponding in size to the capacity of

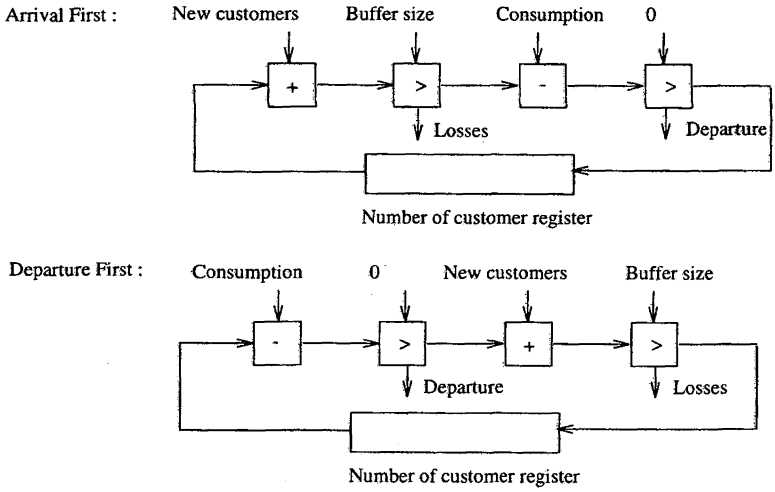

Figure 3. – Hardware representation of a queue. This is a discrete time model (AF or DF) where arrival and departure processes are not fixed. In each case only a register and some logic glue are needed. Arrows give the way of propagation of integer signals.

the queue. However, in the case of multiple arrivals, multiple information has to be stored in one slot. If packets are in conflict, they all have to be stored in the memory in the same time slot. As a result, a slot has to be decomposed into multiple clock cycles. This reduces the simulation speed and increases the time consumption, –since one slot requires multiple clock cycles– and the hardware size consumption, since each queue needs a memory. This represents, however a considerable extension, as packets can be tagged to take into account priority, or to carry dates. It is thus possible to study the end-to-end delay and delay variation, or even to study a single communication with background traffic.

The results presented in Section 5 have been obtained using the first model of a queue with the AF option.

3.2. Traffic supplier

There are different ways of supplying traffic to the network. It is possible to feed the Network with real traffic stored in the memory before beginning the experiment. This is an interesting possibility since the characteristics of real traffic are very hard to reproduce [15]. However, to reach a low loss probability, a very large sample of traffic is required, which is too large to be stored in the emulator memory. Another solution is to emulate traffic using statistical models. The most widely used models are:

- uniform traffic, geometric process: in this model, packets arrive at each input port according to an independent and identically distributed Bernouilli process with parameter p ($0 < p \leq 1$). That is to say a cell arrives at each slot with the probability p. Thus p represents the *input load* or the *arrival rate* at an input port (denoted by ρ). Assuming this model for real traffic leads to an optimistic performance evaluation. In fact, this model does not capture the bursty nature of real time traffic [18].
- Markov Modulated Bernouilli Process (MMBP) [3]: several models have been proposed to describe real traffic, from which *on-off* sources have been widely studied [21, 10]. These sources are characterized by three parameters, λ, μ, r, where λ and μ denote respectively the inverse sojourn times in the off and on state, and r denotes the rate of packet generation in the on state (see Figure 4). The input load ρ is defined as the ratio $r\lambda/(\lambda+\mu)$, and the *burst length* is defined as the average amount of information generated during a single on period of the source, which equals r/μ. Experimental studies have however revealed auto-similarity properties in the behavior of network traffic. Markovian models which capture these properties can be found in [15].

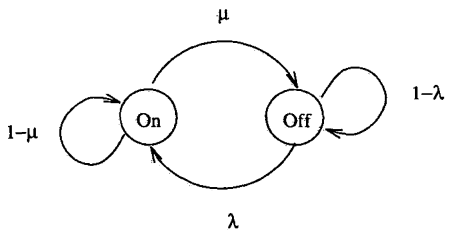

Figure 4. – An *on-off* source; during the *on* state, packets are generated with rate r.

Despite the drawbacks of the uniform traffic model, it has been used to obtain the results of Section 5. Hardware sources are thus built with a memory-based random generator (see appendix A). This generator gives a n bit long random word which goes through a comparator with $\rho * (2^n - 1)$. The result of this comparison is a bit of presence of a packet, with the probability ρ.

The destination of each cell is chosen uniformly from among all output ports and independently of all other packets. For this, each 2×2 switch has a memory-based random generator. This random generator gives a two bit long random word which choose the output for each input port.

This memory-based random generator can also be used to implement a MMBP. A state register has to be used to know the state of the $on - off$

source. The value given by the random generator is then compare to $\lambda * (2^n - 1)$ if the source is in the off state or to $\mu * (2^n - 1)$ if the source is in the *on* state. We are now able to build queues and customer suppliers, but instrumentation is needed to collect results and to capture interesting events. As seen previously (in Section 3), the structure of the component strongly depends on the model, and also on the required performance parameters. An instrumentation component is therefore built to collect important information. When this has been performed, the experiment has to be cleverly controlled. In particular the stop conditions are discussed in the second subsection.

3.3. Collecting results

The information required for the cell loss probability is the number of emitted and rejected packets. There are many different packet loss probabilities to be studied. For example, the loss probability for the whole network, for a single stage, or for a single queue. In terms of hardware, this means one register for the number of incoming cells and another for the number of rejected cells. This is necessary at every point where the cell loss probability has to be measured (see Figure 10).

To study the effect of the queuing network on traffic characteristics, a *traffic analyzer* is required. In fact, it may be desirable to collect statistics, such as the load on a queue, the mean burst length observed on a link, or to study the inter-arrival process. This is also performed by implementation of counters and registers.

All these aspects have to be anticipated and planned during the circuit description. This instrumentation is often more significant than the queuing network in terms of hardware occupation. So there are two different types of hardware implemented. The first is the implementation of the problem, and is composed of queues and sources. The second is the instrumentation depending on the performance measures studied. This type of hardware is composed of counters and registers.

4. INSTRUMENTATION AND CONTROLLING EXPERIMENTATION

4.1. Emulation time duration

This problem is linked to the control of the emulator. The experiment can be stopped when a register is equal to a given value. The register of the number of lost cells was chosen as a trigger mechanism for this purpose. The problem is to choose the most suitable value of this register.

Let X_i be

$$X_i = \begin{cases} 1 \text{ if the i}^{th} \text{ cell is lost} \\ 0 \text{ otherwise} \end{cases}$$

The performance parameter studied here is the loss rate EX_i, an estimator is given by :

$$\overline{X}_n = \frac{1}{n}\sum_{i=0}^{n} I\!I_{\{X_i=1\}}$$

Suppose that, for the analysis of the problem, $\{X_i\}^{i\in I\!N}$ are independent random variables from a common distribution.

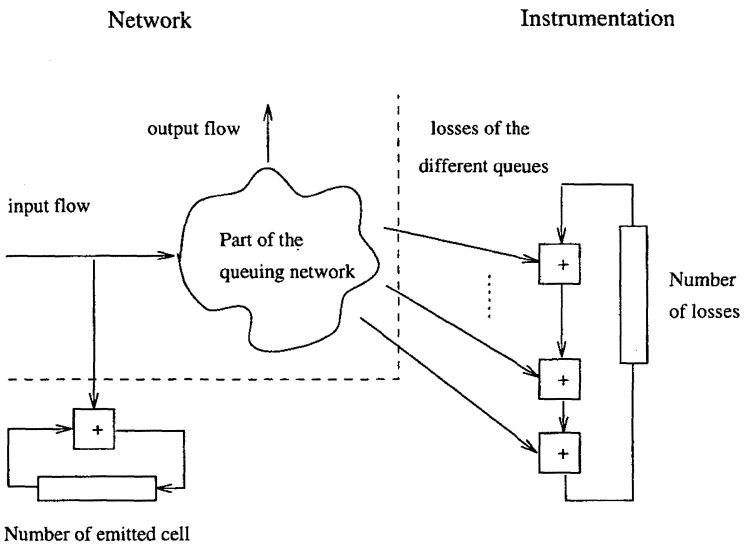

Figure 5. – Hardware instrumentation for collecting information on the network activity. Arrows give the way of propagation of integer signals.

Using the central limit theorem [17], and the standart unbiaised estimator

$$\hat{\sigma}_n^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left(X_i - \overline{X}_n\right)^2$$

the confidence interval $z(\alpha)$ with level α is computed.

In a typical experiment for $\alpha = 0.05$, $n = 10^{10}$ and $10,000$ losses, the confidence interval is 2%, while it is 6.3% with $1,000$ losses.

In fact for bursty traffic, the variables X_i are no longer independent. When a queue is full, the probability of loss ($X_i = 1$) is higher, but *bursts* of losses are still expected to be independent of one another [2]. In practice, to produce accurate results, it was decided to reproduce each experiment several times. Each experiment is stopped when $1,000$ packets have been lost. The values presented in Section 5 are, for each point, the mean of those experiments. The result are given for a confidence interval of 6.3%.

4.2. Cost

We can see that, like every "program", the implementation cost can be mesured in terms of "time" and "space".

Let n be the number of cells to be treated, q the number of operations that have to be done on each cell (in our case this is clearly the number of services required by each cell) and s the number of sources.

The complexity of the problem is then $\mathcal{O}(nq)$, as q operations have to be performed on each cell. This is also the time complexity of a sequential algorithm. In our case, the q operations are made in a pipeline, and the s sources emit in parallel with a rate ρ. This is why the number of clock cycles needed to treat n cells is:

$$\mathcal{O}\left(\frac{n}{s\rho}\right)$$

As an example, for $n = 10^{10}$ cells on the four-by-four switch ($s = 4$), and $\rho = 0.8$, the number of clock cycles required is $3.1\ 10^9$. As the clock frequency is 10^6 hz, 50 minutes are necessary.

The place complexity depends on the complexity of a single queue and the size of the switch (4×4, 8×8 ...), that is to say on the hardware surface needed for the implementation. As seen in Section 3, a queue can be implemented in different ways. Each way has a different cost in terms of hardware occupation. Moreover, the instrumentation often represents a considerable cost, which is why the place complexity is hard to quantify. As an example, four 4×4 switches in parallel only use seven boards of the M500 machine, i.e. 30% of the total capacity.

The most important point to note is that the time needed to evaluate the loss rate on the third stage is no longer than that required to evaluate the loss rate on the second one.

5. APPLICATION TO THE TWO STAGES FOUR-BY-FOUR SWITCH

This section is devoted to the study of a four-by-four switch (see Figure 1). The first subsection concerns the cell loss probability. This leads to the study of the perturbation introduced by the switch on the traffic. In particular, properties of the output traffic are compared to those of the input traffic in the second subsection. The traffic model adopted is the geometric one presented in Section 3.

5.1. Loss probability function of capacities

Figure 6 shows the global packet loss probability, which is the loss probability over all the four-by-four switch. The x axis is the second stage queue capacity K_2 varying from 10 to 50. Each curve corresponds to a different size of first stage buffer ($K_1 = 10, 15, 20, 30$).

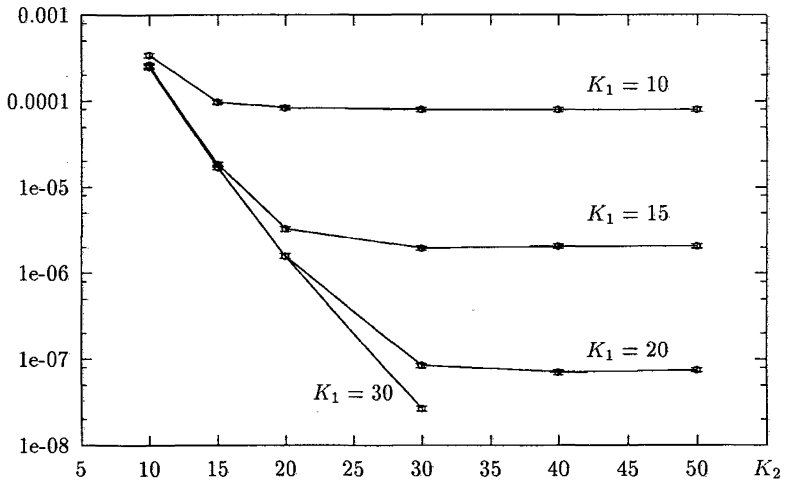

Figure 6. – Loss rate versus capacities of the first (K_1) and second stage (K_2), $\rho = 0.8$. For $K_1 = 30$ and $K_2 > 30$ too few losses have been observed to produce an accurate value of the loss rate.

A plateau can be seen on each curve for high values of K_2. This can be explained by the fact that the second stage is large enough and thus all observed losses arise from the first stage. In this case, increasing the capacity of the second stage will be useless.

On the other hand, for low values of K_2, the curves are all together. In fact the capacity K_1 of the first stage is large with respect to that of the second stage. As a result, all losses arise from the second stage. In this case

increasing the size of the first stage without increasing the capacity of the second will not be useful.

For example, with $K_1 = 20$, the "optimal" configuration, in terms of losses with respect to memory cost, is obtained for $K_2 \simeq 30$. That is to say, if $K_1 = K_2$, the cell loss probability differs on each stage.

The cell loss probability per stage can be seen in Figure 7, where the capacities of stages are equal ($K = K_1 = K_2$). It should be noted that losses are always greater on the second stage. This could be explained by the fact that the first stage induces perturbations in the traffic. In other words the traffic following a buffer stage is more bursty than the one at the entrance.

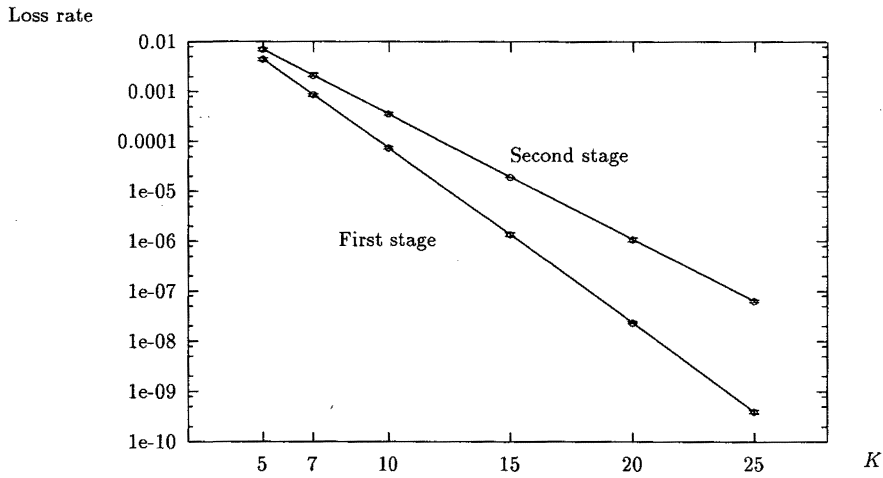

Figure 7. – Loss rate at the first and second stage versus capacity ($K = K_1 = K_2$ **and** $\rho = 0.8$).

This is the reason why a traffic analyzer has been built to characterize these perturbations. Information gathereded by this traffic analyzer is presented in the next subsection.

5.2. Traffic perturbations

Traffic perturbations are shown by a traffic analyzer. The traffic analyzer gives the number of inter-arrival times that lasted j slots at the output of the i^{th} stage. This allows computation of the probability a_j^i of observing an inter-arrival time of j slots at the output of the i^{th} stage. It also gives the number of bursts that lasted j slots at the output of the i^{th} stage. This allows

computation of the probability b_j^i of observing a burst that lasted j slots at the output of the i^{th} stage. As a convention $i = 0$ gives these probabilities at the output of the source stage.

Figure 8 presents the distribution a_j^0 and a_j^2 and Figure 9 presents the distribution b_j^0 and b_j^2 $(0 < j \leq 16)$. As the arrival process is a geometric process of parameter ρ, the theoretical distribution at_j^0 of a_j^0 can be computed:

$$at_j^0 = (1 - \rho) * \rho^{(j-1)}$$

A χ^2 test ([1]) on the observed distribution a_j^0 confirms the accuracy of the traffic supplier. For b_j^0 the theoretical distribution bt_j^0 is given by:

$$bt_j^0 = \rho * (1 - \rho)^{(j-1)}$$

A χ^2 test on the observed distribution also confirms that the traffic observed at the output of the sources is the one expected.

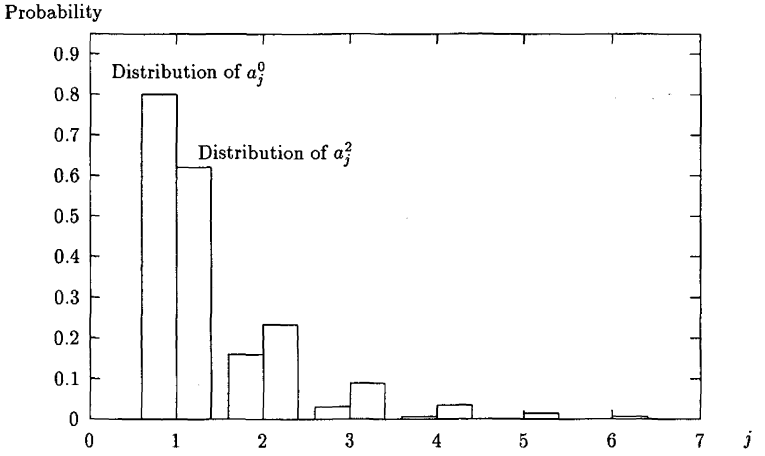

Figure 8. – Probability of an inter-arrival time lasting j slots, at the sources (a_j^0) and at the output of the switch (a_j^2). $\rho = 0.8$ and $K_1 = K_2 = 10$.

Distributions a_j^2 and b_j^2 show the perturbations introduced by the switch on the traffic. From Figure 8, it is clear that the mean inter-arrival time has increased. Figure 9 shows that the number of small bursts has increased,

([1]) All the statistical tests are done with a degree of confidence of 95%.

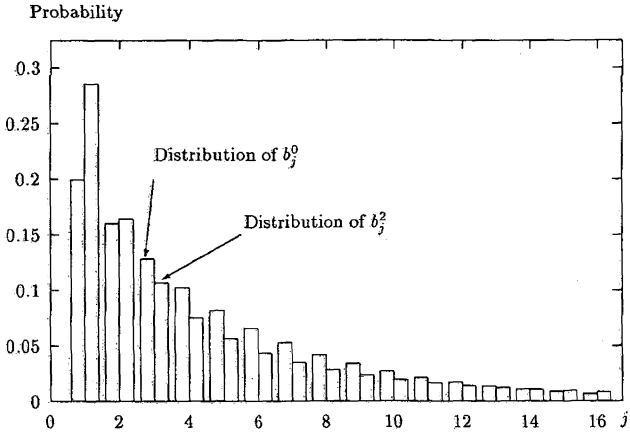

Figure 9. – Probability of a burst lasting j slots, at the sources (b_j^0) and at the output of the switch (b_j^2). $\rho = 0.8$ and $K_1 = K_2 = 10$.

TABLE I

Distribution a_j^i and b_j^i. ($K_1 = K_2 = 10$ and $\rho = 0.8$).

j	b_j^0	b_j^2	a_j^0	a_j^2
1	0.1994	0.2851	0.7998	0.6196
2	0.1602	0.1641	0.1602	0.2323
3	0.1283	0.1069	0.0319	0.0894
4	0.1024	0.0756	0.0063	0.0351
5	0.0819	0.0562	0.0012	0.0139
6	0.0655	0.0433	0.0002	0.0056
7	0.0527	0.0344	$5.3E-5$	0.0022
8	0.0417	0.0280	$9.4E-6$	0.0009
9	0.0335	0.0232	$1.8E-6$	0.0003
10	0.0267	0.0194	$5.0E-7$	0.0001
11	0.0214	0.0165	$3.1E-7$	$6.1E-5$
12	0.0171	0.0141	0	$2.5E-5$
13	0.0137	0.0122	0	$1.1E-5$
14	0.0109	0.0107	0	$3.7E-6$
15	0.0087	0.0094	0	$9.7E-7$
16	0.0069	0.0083	0	$3.2E-7$
> 16	0.0282	0.0918	0	$6.4E-7$

but the number of large bursts has also increased as $b^2_{j>16} = 0.918$ and $b^0_{j>16} = 0.0282$ (see Table I).

Thus the difference of losses on the different stages can be explained by the fact that the mean burst length has been increased by the two stages of the switch.

6. CONCLUSION AND EXTENSION

In this document, a new methodology for simulation of discrete time queuing networks with finite capacities has been presented. This new methodology uses a versatile architecture, configured for maximum efficiency for a given problem. This method has to be compared to software tools which only allow the computation of higher (10^{-5}) cell loss rates. What is important to note is the fact that this architecture is very easy to configure, and that this configuration is very easy to debug. This last point results from the fact that all signals of the configuration are available.

This new approach has been applied to the study of high-speed packet switched networks. This has allowed for simulation of realistic cell loss probabilities $(10^{-8}, 10^{-9})$ at all stages of a multistage ATM switch. Using this thechnology, it is thus possible to highlight rare events with a high degree of accuracy.

A traffic analyzer has also been presented. This traffic analyzer shows the perturbation induced by a multiplexer on the traffic and shows why traffic modeling is so important.

For further studies, the model has to be extended to real cells carrying information. This extension will allow studies on end-to-end delay and delay variation (jitter). It is also possible to introduce priority levels for packets, and new buffering strategies, such as partial sharing.

More generally, this type of machine could be used to emulate numerous types of network protocols, and also to solve different discrete time queuing network problems.

7. ACKNOWLEDGEMENT

The authors would like to thank V. Olive and S. Martin ([2]) for fruitful support during the course of this work. The authors also thanks the anonymous referees for their valuable comments.

([2]) from the CNET-Grenoble.

APPENDIX

A. PSEUDO RANDOM GENERATOR

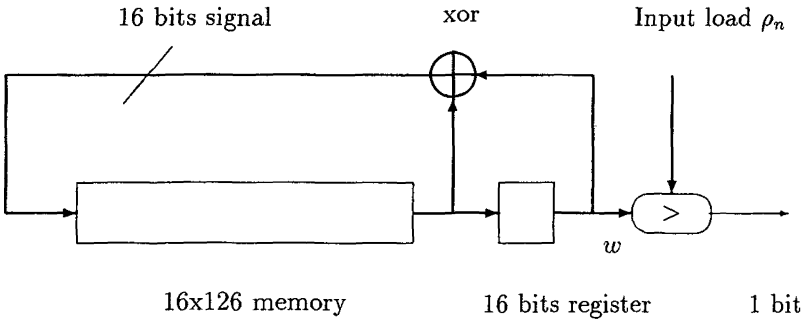

Figure 10. – The memory based random generator.

The random generator is based on the polynomial $1 + X + X^{127}$ (see [18]). A double part memory is used as a shift register. Every clock cycle a 16 bits random word w is generated $\left(0 \leq w \leq 2^{16} - 1\right)$. To reproduce the geometric process describe in section 3 this word is compared with

$$\rho_n = \lfloor \rho * 2^{16} - 1 \rfloor$$

Where $\lfloor . \rfloor$ is the lower integer part. For example with $\rho = 0.8$ we have $\rho_n = 52429$. A cell is emitted if $w \leq \rho_n$.

Here is a part of the VHDL code for this random generator:

```
– A double port memory is used. Addresses are 7 bit words.
  signal AddRead, AddWrite : std-ulogic-vector(6 downto 0);
– Addresses as integer.
  signal NumWrite, NumRead : integer range 0 to 125;
– 16 bits words read or write in memory.
  signal ToBeWrite, Read : std-ulogic-vector(15 downto 0);
– A 16 bits register at the output
  signal regX0 : std-ulogic-vector(15 downto 0);
  – Type Conversion.
    AddRead <= std-ulogic-vector(Conv-unsigned(NumRead,7));
    AddWrt <= std-ulogicvector(Convunsigned(NumWrite,7));
    if CK'event and CK='1' then
        – xor operator.
        ToBeWrite <= Read xor regX0;
        – Shift
```

```
        NumRead <= (NumRead +1) Mod 2**Tad;
        NumWrite <= (NumWrite+1) Mod 2**Tad;
        regX0 <= Read;
        -- Comparator.
        if ( conv-integer(unsigned(regX0))< rhon ) then
          res<=1;
        else
          res<=0;
        end if;
    end if;
```

REFERENCES

1. A. Gravey and G. Hébuterne, Simultaneity in discrete-time single server queues with Bernouilli inputs, *Performance Evaluation North-Holland*, 1992, *14*, pp. 123–131.
2. B. Sericola, F. Guillemin, G. Rubino and A. Simonian, Transient characteristics of an $M/M/\infty$ system applied to statistical multiplexxing on an ATM link. Publication interne 874, IRISA, October 1994.
3. W. Fischer and K. Meier-Hellstern, The Markov-modulated Poisson process (MMPP) cookbook, *Performance Evaluation North-Holland*, 1993, *18*, 2, pp. 149–171.
4. E. Gelenbe and G. Pujolle, *Introduction to Queueing Networks*, John Wiley & Sons Limited, 1987.
5. G. Hébuterne, *Écoulement du trafic dans les autocommutateurs*, Masson, Institut National de Télécommunications, 1985.
6. F. Hübner, Discrete-time analysis of the busy and idle period distributions of a finite-capacity ATM multiplexer with periodic input, *Performance evaluation North-Holland*, 1994, *21*, 1-2, pp. 23–36.
7. L. Burgun, F. Reblewski, G. Fenelon, J. Barbier and O. Lepape, Serial Fault Emulation. In *Proceedings of the 33rd Design Automation Conference 1996 (DAC 96)*, Metasystems, France, 1996, pp. 801–806.
8. J. Pellaumail, Majoration des retards dans les réseaux ATM, *Rairo recherche opérationnelle*, 1996, *30*, pp. 51–64.
9. L. Burgun and F. Reblewski, Première Génération d'Emulateurs Matériels Metasystems. In *Quatrième symposium sur les architectures nouvelles de machines*, Metasystems, France, 1996.
10. G. Meempat, G. Ramamurthy and B. Sengupta, A new performance measure for statistical multiplexing : perspective of the individual user, *Performance evaluation North-Holland*, 1996, *25*, pp. 59–80.
11. J. Pellaumail, *Graphes, Simulation, L-matrices, application aux files d'attente*, Hermes, 1992.
12. G. Pujolle, Commutateurs ATM: Classification et Architecture, *Technique et Science Informatique*, 1992, *11*, 1, pp. 11–29.
13. R. Airiau, J.-M. Berge and V. Olive, *VHDL du langage à la modélisation*, Presses polytechniques et universitaires romandes, France Telecom, 1990.
14. R. Airiau, J.-M. Berge and V. Olive, *Circuit Synthesis with VHDL*, Kluwer Academic Publishers, France Telecom, 1994.

15. S. ROBERT and J.-Y. LE BOUDEC, Can Self-Similar Traffic Be Modeled by Markovian Processes?, *Lecture Notes in Computer Science*, 1996, 1044.
16. J. ROBERTS and F. GUILLEMIN, Jitter in ATM networks and its impact on peak rate enforcement, *Performance Evaluation North-Holland*, 1992, *16*, 1-3, pp. 35–48.
17. S. M. ROSS, *A Course in Simulation*, Mamillan Publishing Company, University of california, Berkeley, 1991.
18. R. Y. AWDEH and H. T. MOUFTAH, Survey of ATM switch architectures, *Lecture Notes in Computer Science*, 1995, *27*, pp. 1567–1613.
19. S. TEZUKA, *Uniform Random Numberd: Theory and practice*, Kluwer Academic Piblishers, IBM Japan, 1995.
20. L. TRUFFET, *Méthodes de Calcul de Bornes Stochastiques sur des Modèles de Systèmes et de Réseaux*, PhD thesis, Université Paris VI, 1995.
21. Y. XIONG, B. STEYAERT and H. BRUNEEL, An ATM statistical multiplexer with on /off sources and spacing: numerical and analytical performance studies, *Performance evaluation North-Holland*, 1994, *21*, 1-2, pp. 37–58.