An efficient method for performance analysis of high speed networks : modeling and hardware emulation

Cyril Labbé*and Jean-Marc Vincent
LMC-IMAG BP53
38041 Grenoble Cedex 9, France
{Cyril.Labbe, Jean-Marc.Vincent}@imag.fr

Abstract

Keywords : Emulation on a Versatile Architecture, Performance Evaluation, High speed Networks, Queueing Networks, Rare Events, Experimentation.

1 Introduction

Integration of services is a crucial issue for the design and the control of high speed networks. The Asynchronous Transfer Mode technology try to ensure a quality of service for various types of applications. Such a network is based on a decomposition of messages in fixed-sized cells which are sent along a virtual circuit between the two end-points. The quality of service should be guaranteed by the network operator. But, the difficulty is to integrate several types of traffics and schedule the communications according to quality criteria for each traffic.

For example, such a network should transmit files without alterations of the contents. This implies a guaranty on the loss rate of packets, typically 10^{-9}. Another application such as teleconferencing or on-line video needs a huge amount of bandwidth to ensure a good throughput, typically $2Mb/s$. Other types of traffic, like audio communications, needs a control on jitter or cells delays variations...

Consequently, the identification of congestions, that generate the most important part of performance degradations has been a topic of intense study. Several approaches could be followed to establish the relation between the input traffic and the real performances [17]. The choice of a methodology depends essentially on the accuracy of the desired estimates.

Mathematical modeling techniques provide general results on the network behavior[22]. They generally underline a canonical model for input traffic such as Bernoulli process or Markov-modulated Bernoulli process and assume some independence properties inside the network [23]. These efficient methods are useful at a high level of abstraction of the model and give realistic orders of magnitudes. It could also be efficient on subsystems studied in isolation. On another hand, asymptotic techniques based on *Large Deviation* establish results on rare events such as losses [16].

Discrete event simulation [4] is an alternative for considering more detailed models. Simulation libraries such as *NS* developed at University of California [1] allows composition of sub-models like links, servers, etc. Performance indexes that are computed concern the general behaviour of the network, utilization, load, mean numbers of cells, response time... But estimation of rare events probabilities increase drastically the simulation duration. Parallel solutions have been proposed in [15], even a linear speedup does not offer a direct evaluation of loss rates. Then, ad hoc variance reduction technique should be implemented to observe and estimate these events.

Obtaining a global and accurate view of the behaviour of the network appears to be very difficult with these approaches. Because discrete event simulation addresses a too large class of models and mathematical models are covering not enough general cases, we have to find a new modelling technique, for which models are reasonably tractable and with a sufficient accuracy. One of the major characteristic of ATM networks is the fixed size of cells. Consequently, the system could appear as a clock driven discrete event system, a step of the clock is simply the emission time of a cell. We have also to remark that such systems are highly parallel, composed by small switches with many queues, links... Then system could be modelled by a huge automaton, parallel product of many elementary “independent” automata that run in a synchronous fashion.

*This work has been done with the financial support of M2000 industry, 4 rue R. Razel 91400 Saclay France

The point of this approach is that such an automaton could easily be emulated on a dedicated circuit. Reconfigurable architectures emulate the automaton and measurements give performance indexes or control parameters. Stiliadis and Varma have already followed this approach for scheduling analysis for ATM switches. Authors focused on the architecture of the emulator and illustrate the methodology on mean behaviour of the system. The context was the *FAST* project at the University of California Santa Cruz[19].

The aim of this paper is to detail this approach based on emulation on a programmable circuit, a versatile architecture. This has been validated by an implementation on a versatile architecture Sim-express. Several network behaviour have already been studied : evaluation of loss rate in multistage interconnection networks [13], modelling of discrete time networks [11], analysis of complex scheduling disciplines [12]. A general presentation of the emulation tool have been done in [14] and [10].

The structure of the paper is the following. The modeling approach is detailed in Section 2. The soft and hard environment is presented in section 3. The last section illustrate the method on a typical problem of performance evaluation for Fair Queueing policy.

2 Modeling Networks protocols in a hardware description language

In this section, the method for modelling network protocols in synthesizable VHDL is described. The method is based on discrete time queueing networks. Different ways of queue building are proposed, showing that the hardware representation depends on the adopted model. The two first subsection presents the random generator and the technique used to produce arrival and departure process. In the third subsection the case of the $././1/k$ queue is considered.

2.1 Traffic generator

The most natural way to supply a random word using hardware resources, is to use a feedback shift register [9, chap. 3] or [21]. The random generator is based on the polynomial $1+X+X^{127}$ and generate a random bit at each clock cycle. The implementation in parallel of such 16 registers allow the generation, every clock cycle of a 16 bits random word w, $0 \leqslant w < 2^{16}$. This word w can be considered as the realization of a uniform pseudo-random variable with integer values in $0, \cdots, 2^{16}-1$.

As we decide to feed our queueing network with random sources, the problem now is to generate, using hardware resources, the uniform random into something more complex. For example, to reproduce a geometric[1] arrival process with an intensity of ρ cells per slots, the pseudo random word w is compared with the threshold $\theta = \rho(2^{16}-1)$. For example with $\theta = 52429$ a cell is emitted if $w \leqslant \theta$ and we obtain an intensity of $\rho \simeq 0.8$.

This method can be used for more complex sources, like for example a On/Off Modulated Bernoulli Process. In this case a state register is updated according to a random word and to the probability of transition. Then an other random word is compared to the probability of emission in the current state. This can be enlarge to other $MMBP$ sources, the discrete time version of a Markov Modulated Poisson Process [6], where two random words are used one for the state transition, one for the emission (figure 1).

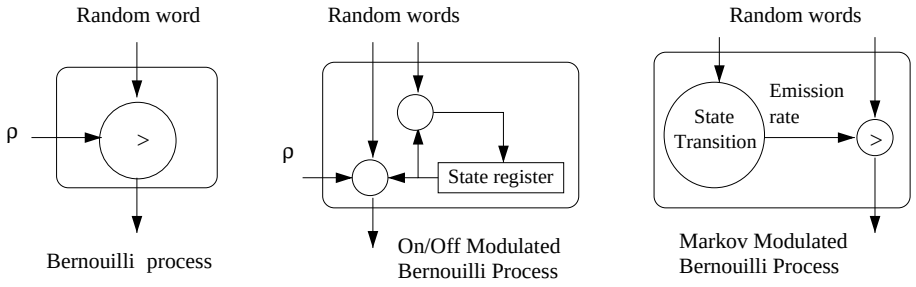

Figure 1: Generalization of random traffic sources.

By this way, a library of traffic generators modules has been developed. The interchangeable modules produce a standard output signal which will be connected to the input pins of the queueing network modules.

[1] also called Bernoulli process

2.2 The ./1/k queue module

If customers are all similar, only a register and some ”logic glue” are needed. The register contains the number of waiting customers and is updated according to input signals :

- the number of new customer arrivals,
- the queue’s capacity,
- the number of served customers.

It is important to note that a slot (a slot is a time sample in discrete time) is equivalent to a clock cycle. Consequently the simulation duration is directly related to the number of clock cycles and is independent of the number of served cells.

To introduce information in customers, real packets carrying information can be used. This information has to be queued. This means that each queue of the network is implemented using a memory corresponding in size to the queue capacity. 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 increase the simulation time –since one emulated slot requires multiple clock cycles– and the hardware size consumption, since each queue needs memory. This represents, however a considerable extension, as packets can be tagged to take into account priority, address, hight level information or time stamps. It is thus possible to study the end-to-end delay and delay variation, or even to study a point to point communication with a background traffic.

2.3 Instrumentation

The purpose of the modelling is to evaluate performance indexes : queues occupation, waiting time, losses. Moreover, it could be very useful to know the impact of traffic parameters on the performances. Then it is necessary to create a certain number of measurement modules. Their purpose is to catch, stored and synthesize informations on the state of the studied system.

For example, to know the loss rate at a queue, it is necessary to count the number of losses that has appear during the simulation, the occupation rate is given by the amount of idle slots. More generally, a large number of performance indexes could be computed from counting specific events.

This counting is done thanks to a register and a simple adder. Counters could be generalized to obtain more complex statistic estimators on durations, distributions, correlations, etc. Figure 2 shows the architecture of the emulator.

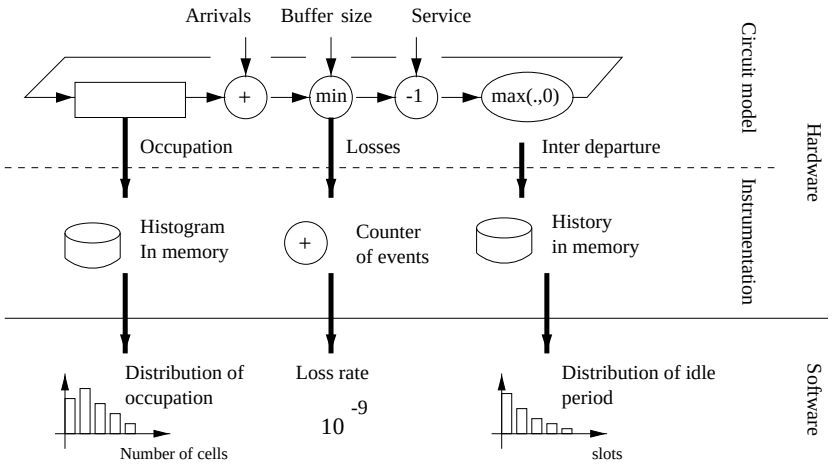

Figure 2: Hardware representation and instrumentation of simple a queue.

3 Hardware and software environment

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

3.1 Architecture

The hardware simulator is the Sim-express machine from Metasystems [5]. The emulator acts like a giant FPGA on which the circuit to be tested and debugged can be mapped. This give to the user the effective use of 500,000 programmable logic gates, 17 Mbytes of memory (single or double port) and an adjustable clock frequency from 1 to 10 Mhz.

All this hardware can be shaped to emulate any digital and synchronous circuit. Memories can be loaded (initialized) 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.

This machine is from the *first generation* (1994). An up to date machine has at least 20 time more logic gates.

3.2 Software environment

The software flow leads to the files required by the emulator to reproduce the functionalities of a circuit. These functionalities 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 detailed above :

- a VHDL (VHSIC Hardware Description Language) description of the chip is used to describe the system in terms of concurrent processes [3].
- 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 [3].
- The Metasystems compiler computes the links for the set of gates available in the machine. This is the routing operation, which results in connecting the gates to each other through the programmable network of the emulator.

Those two last steps are entirely automatic except for customizing the final result with Metasystems memory models.

3.3 Simulation control : visualization and statistic analysis

Emulation is performed using the MEL tool, which loads the emulator with the configuration file, and allows run control, logic analysis, triggering features, and patterns verification. MEL can be driven by procedures written in a C-like code, which is useful for complex simulation.

All the signals or vectors (busses) can be displayed in a waveform window (see Figure 3).

Control of input signals or registers can be done through the monitor window (see Figure 3). Any signal and register value can be displayed and modified.

4 Application to Fair Queueing

4.1 Generalized Processor Sharing and Fair Queueing policies

Various scheduling strategies have been proposed to guarantee an amount of bandwidth among the flows of cells. These policies are based on reservations of users and the service rate for a flow should be proportional to the reservation's value ϕ_i for stream i [20], [8].

In a continuous context, fluid traffic sources and services, the policy *General Processor Sharing* (GPS) guarantees the proportional rates. Unfortunately, ATM networks do not satisfy fluidity assumptions. *Fair Queueing*

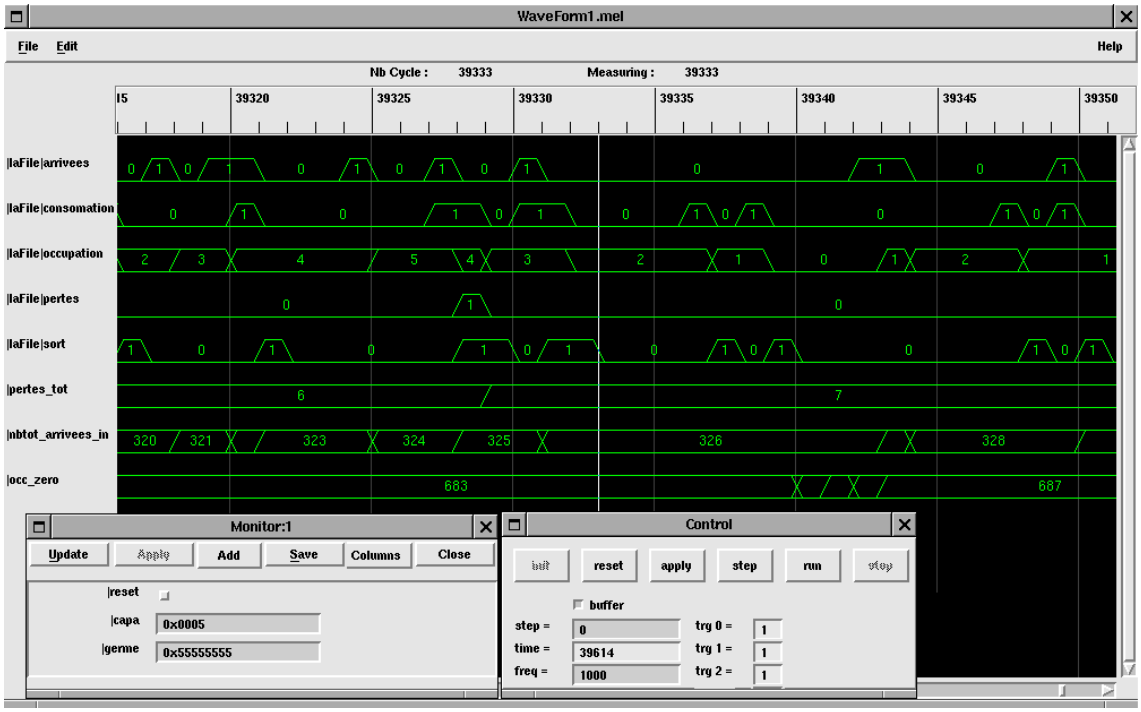

Figure 3: The waveform window display all signals and register values on the last 7000 clock cycles.

policies try to approach the behaviour of GPS [7]. The main idea is to serve cells according to a priority mark, this mark is computed as if a GPS policy were applied. A fair queueing server maintain a table of marks corresponding to the cells in the buffers. The computations of marks is generally based on a *virtual time* which is updated at each slot. This needs particular operators like minimum, maximum, weighted sums, that must be hard-coded on the switch. Consequently, the hardware complexity is highly related with the quality of the scheduling.

These systems seem to be untractable by analytic modelling, this is due to the interleaving of flows and the non "feed-forward" computation of marks. Moreover, if fair conditions are respected, the structure of the output traffic is very difficult to catch.

The aim of this section is to describe a module that emulate a simple *Fair Queueing* protocol and analyze the output traffic. It is shown that the global output traffic is statistically regular, but, for a given flow, perturbations may appear in the distribution of output bursts. This example illustrate the capability of the hardware modelling to analyze complex policies. This work has been applied for the CMS switch developed by CNET-Lannion [12, chap 7].

4.2 Hardware description of a FQ discipline

A typical strategy used in ATM network is *SFF* : the cell which has the *S*mallest virtual *F*inishing time at time t is *F*irst served.

To reduce the number of sums in the computations [2], [18], the virtual time $v(t)$ of this discipline is defined by :

$$v(t) = \ virtual\ finishing\ time\ of\ the\ last\ served\ cell.$$

Theoretical study on fair queueing try to catch worst cases. So, they often used periodic input traffic. To observe more realistic behaviours, random sources has been implemented to make statistics measurements. The generated input is a superposition of N Bernoulli streams of intensity α_i for stream i. The global load of the system is denoted by $\rho = \sum \alpha_i$.

The architecture of the fair queueing module is synthesized in figure 4. It contains sources, waiting queues (one for each input stream), one table of marks corresponding to the first cell in each queue and server. An instrumentation module compute histograms of silent or burst durations for each traffic and for the global stream.

The modules parameters and values are the following :

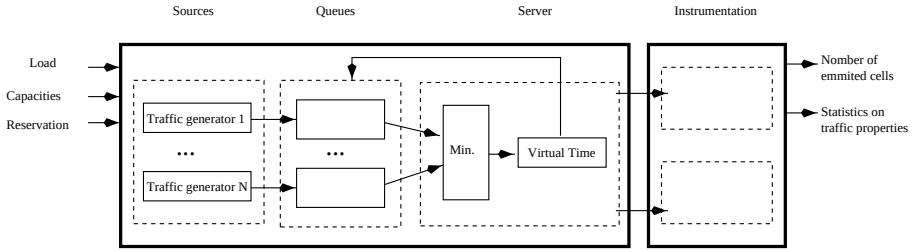

Figure 4: The Fair Queueing module.

Parameter	Experimental value
Number of streams	4
α_i	$0.125, 0.225, (\rho = 0.5, 0.9)$
ϕ_i	$\frac{1}{4}$
Capacity	300 per stream

For the experiments, the slot is 3 clock cycles long, the choice of the cell to be served is obtained by a minimum on the marks and take $log_2(N) + 1 = 3$ clock cycles. The simulation duration is decomposed in a starting period of 10.000 slots to reach the steady state and a measurement period of 10^7 slots. The real simulation time on the machine is about $30s$ for one load.

4.3 Measurements on the silent output periods of the server

We first observe the utilization of the server. The instrumentation module gives histograms on the silent and burst periods of the output traffic. The results of figure gives the empirical distributions for the silent durations (figure 5).

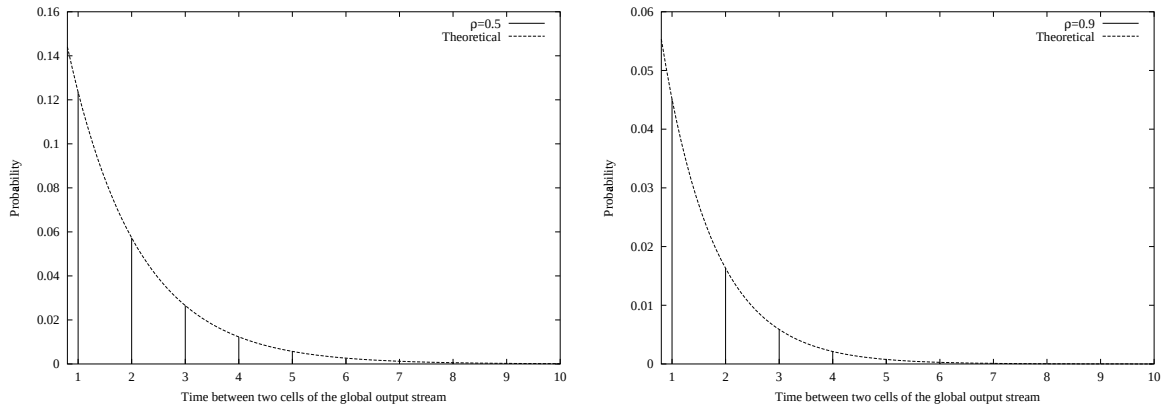

Figure 5: Silent empirical distribution for the output traffic.

Because the scheduling policy is non-idling, the silent periods are geometrically distributed with parameter $\rho' = (1 - \alpha)^N$. A statistical test on the parameter of the distribution and a χ^2 test on the whole distribution with confidence level of 95% shows the adequation between the theoretical results and the empirical measurements. These results could be interpreted as a partial validation of the emulator.

4.4 Measurements on the silent output periods for a specific flow

The previous argument and other measures shows that the structure of the global output traffic looks like the structure of the global input traffic (bulk arrivals are spread on the burst periods). Such a property is not valid for a stream studied in isolation. We consider the output traffic of cells generated by the first source. As in the previous case, histograms on the output periods are given in figure 6.

We first observe that silent periods have not a geometric distribution. The input silent period are geometrically distributed with parameter $1 - \alpha$ and are also plotted on the figure. It is also interesting to note that asymptotically,

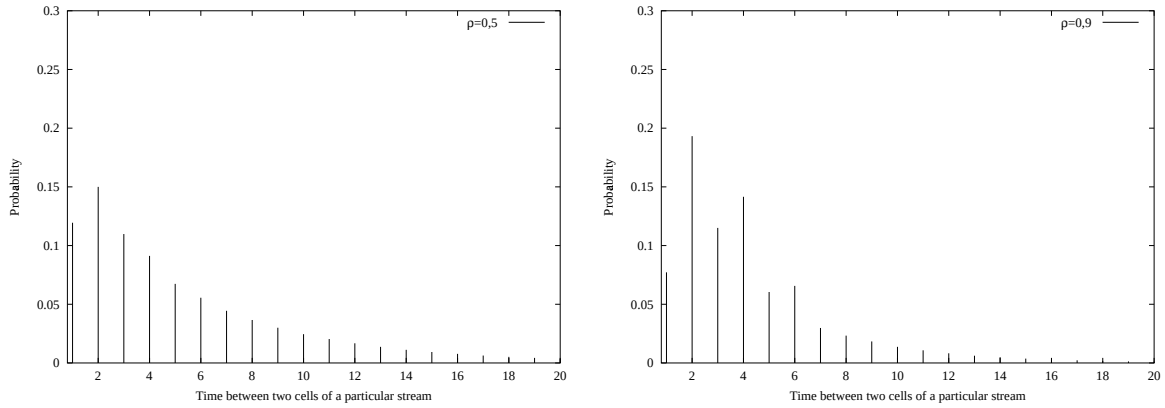

Figure 6: Silent empirical distribution for the global output traffic generated by source 1.

the tail of the empirical distribution converges to the tail of the geometric distribution, which is very interesting for bounding delays.

The fact that the distribution is not unimodal is more surprising. In our opinion, the origin of this phenomenon could be in the algorithm that choose the cell when two cells have the same mark. The implemented version is based on a round-robin policy that alternates cells issued from different traffics.

This implies a traffic perturbation of the output steam that degradates the quality of service, in particular for constraints on jitter. Moreover, the traffic perturbation increases with the load of the system, this match well with our interpretation.

4.5 Buffer occupation

Consider now a buffer associated to one stream. We observe the following distributions 7.

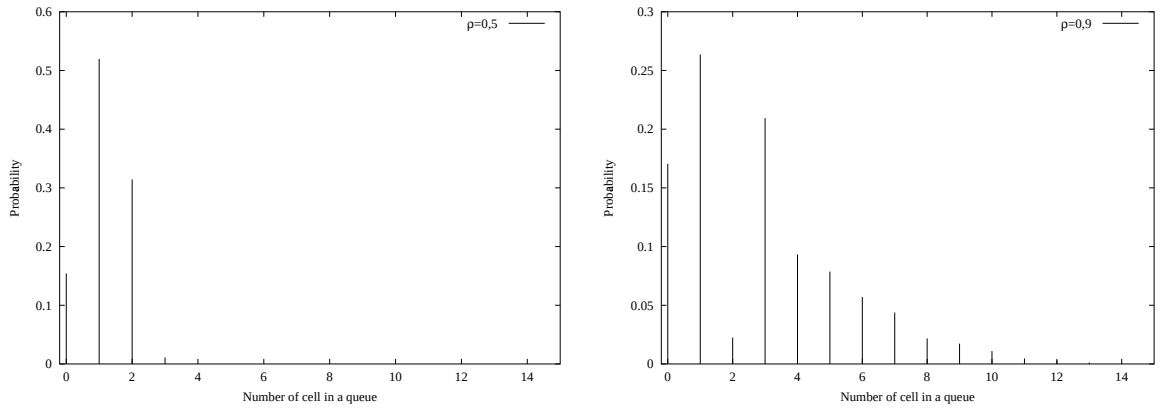

Figure 7: Silent empirical distribution for the output traffic generated by source 1.

Once again, the perturbation clearly appears and is highly related with the global load of the server. Fortunately, it does not affect the tail distribution and consequently the order of magnitude of loss rates should not be affected by the fair queueing policy.

5 Conclusion and extension

In this document, a new methodology for simulation and performance evaluation for high-speed packet switched networks has been presented. This new methodology uses a versatile architecture, configured for the study of

networks protocols. What is important to note is the fact that this architecture is very easy to configure and 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 a Fair Queueing policy. A traffic analyzer has been used to show the perturbation induced by the server policy on the traffic.

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.

6 Acknowledgement

The authors would like to thank P. Vrel for software developments, V.Olive and S.Martin [2] for fruitful support during the course of this work. The authors also thanks the Frederic Reblewski[3] for his technical support.

References

[1] Network Simulator , 1999. University of California, http://www-mash.CS.Berkeley.EDU/ns/.

[2] O. Abuamsha. *Application des m thodes de la comparaison stochastique pour l'analyse des disciplines "Fair-Queueing"*. PhD thesis, PRiSM, Université de Versailles, 1998.

[3] R. Airiau, J-M. Berge, and V. Olive. *Circuit Synthesis with VHDL*. Kluwer Academic Publishers, 1994.

[4] J. Banks and B. Carson, J.and Nelson. *Discrete-Event System Simulation*. Prentice Hall, 1995.

[5] 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)*, pages 801–806, 1996.

[6] W. Fischer and K. Meier-Hellstern. The Markov Modulated Poisson Process (MMPP) cookbook. *Performance Evaluation*, 18(2):149–171, 1993.

[7] A. G. Greenberg and N. Madras. How fair is fair queuing ? *Journal of the ACM*, 39(3):568–598, 1992.

[8] Albert G. Greenberg. Fair Queueing Architectures for High-Speed Networks. In *Proceedings of the Fourth International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, San Jose, California, 1996. IEEE Computer Society TCCA and TCS.

[9] D. Knuth. *The Art of Computer Programming: Semi-Numerical Algorithms*, volume 2. Addison-Wesley, 1973.

[10] C. Labb, S. Martin, and J-M. Vincent. A Reconfigurable Hardware Tool for High Speed Network simulation. In LNCS, editor, *International Conference on Modelling Techniques and Tools for Performance Evaluation, Performance TOOLS'98*, volume 1469, Majorque, 1998.

[11] C. Labb, V. Olive, and J-M. Vincent. Emulation on a versatile architecture for discrete time queueing networks: application to high speed networks. In *International Conference on Telecommunications*, Thessalonica, June 1998.

[12] C. Labbé. *Analyse de Performances pour les rseaux haut dbit : modlisation et mulation sur une architecture reconfigurable*. PhD thesis, Institut National Polytechnique, Grenoble, June 1999.

[13] C. Labbé, F. Reblewski, and J-M. Vincent. Performance Evaluation of High Speed Network Protocol by Emulation on a Versatile Architecture. *RAIRO - Operation Research*, 32(3):253–270, November 1998.

[14] S. Martin, V. Olive, and C. Labb. Mettez un mulateur dans votre simulateur. *Calculateurs Parallles*, 9(1):45–57, 1997.

[15] C. Pham, H. Brunst, and S. Fdida. Conservative simulation of load-balanced routing in a large ATM network model. In *Proceedings of the 12th Workshop on Parallel and Distributed Simulation (PADS-98)*, pages 142–153, Los Alamitos, May 1998. IEEE Computer Society.

[16] A. Schwartz and A Weiss. *Large Deviations for Performance Analysis*. Chapman and Hall, 1995.

[2] from the CNET-Grenoble

[3] from M2000 company

[17] M. Schwartz. *Telecommunication Networks.* Addison-Wesley, 1989.

[18] D. Stiliadis. *Traffic Scheduling in Packet-Switched Networks : Analysis, Design, and Implementation*. PhD thesis, University of California Santa Cruz, 1996.

[19] D. Stiliadis and A. Varma. A Reconfigurable Hardware Approach to Network Simulation. *ACM Transaction on Modeling and Computer Simulation*, 7, 1997.

[20] D. Stiliadis and A. Varma. Efficient fair queuing algorithms for packet-switched networks. *IEEE/ACM Transaction on Networking*, 6, 1998.

[21] S. Tezuka. *Uniform Random Number : Theory and practice*. Kluwer Academic Piblishers, 1995.

[22] J. Walrand and P. Varaiya. *High-Performance Communication Networks.* Morgan Kaufman Publishers, 1996.

[23] M.E. Woodward. *Communication and Computer Networks : Modelling with discrete-time queues*. IEEE Computer Society Press, 1994.