On the application of accelerating simulation methods in network analysis

José Incera
ENST Bretagne
2 Rue de la Châtaigneraie
35512 Cesson Sévigné, France
+33-299 12 70 46
FAX: +33-299 12 70 30
Jose.Incera@enst-bretagne.fr

Gerardo Rubino
ENST Bretagne and IRISA
Campus Beaulieu
35042 Rennes Cedex, France
+33-299 84 72 96
FAX: +33-299 84 71 71
Rubino@irisa.fr

Nicolas Stier
IRISA
Campus Beaulieu
35042 Rennes Cedex, France
+33-299 84 72 96
FAX: +33-299 84 71 71
nicost@dc.uba.ar

Abstract

When evaluating quantitative aspects of communication networks using simulation, one of the main difficulties to face is the often considerable computing power required. In some specific mathematical frameworks, different techniques have been proposed in order to accelerate the simulation process while keeping the same precision in the result. This paper deals with one of them, the so called importance splitting technique, applied to a general discrete event simulator used to estimate low probability values, namely loss probabilities. We present experimental evidence that this approach can be efficient and we point out some of the difficulties that the user can find in following it. We also show that this method can result in accelerating factors even if the estimated probabilities are not very low.

KEY TOPICS: Modeling and simulation of communication systems.

Submitted for the General Conference

On the application of accelerating simulation methods in network analysis

José Incera [†]
ENST Bretagne
2 Rue de la Châtaigneraie
35512 Cesson Sévigné, France
Jose.Incera@enst-bretagne.fr

Gerardo Rubino
ENST Bretagne and IRISA
Campus Beaulieu
35042 Rennes Cedex, France
Gerardo.Rubino@irisa.fr

Nicolas Stier
IRISA
Campus Beaulieu
35042 Rennes Cedex, France
nicost@dc.uba.ar

Abstract

When evaluating quantitative aspects of communication networks using simulation, one of the main difficulties to face is the often considerable computing power required. In some specific mathematical frameworks, different techniques have been proposed in order to accelerate the simulation process while keeping the same precision in the result. This paper deals with one of them, the so called importance splitting technique, applied to a general discrete event simulator used to estimate low probability values, namely loss probabilities. We present experimental evidence that this approach can be efficient and we point out some of the difficulties that the user can find in following it. We also show that this method can result in accelerating factors even if the estimated probabilities are not very low.

1 Introduction

One of the main problems in analyzing the so called "rare events", for instance, in estimating the probability of buffer overflow in some communication networks, is that the standard procedures need a considerable amount of computer effort in order to estimate these numbers with acceptable precision. By standard procedures we mean either the utilization of some general purpose discrete event simulator (or a dedicated one), or the estimation performed by means of a standard Monte Carlo routine, that is, a routine implementing a standard estimator. In both situations, the problem is that, by definition, a rare event needs time to appear. Moreover, we must observe it many times during the simulation in order to obtain a minimal precision in the estimation. For this reason, people working in the Monte Carlo field have developed a complete branch of "accelerated" methods. These methods aim to modify either the problem or the standard estimator, or even implementation aspects, in order to obtain the same precision with less CPU effort (or a more precise one using the same computing time). A well known approach to achieve this objective is the importance sampling technique, where the user modifies the underlying probability measure controlling the dynamics of the model, trying to make the rare event happen more frequently. Another approach is the splitting technique, where the idea is completely different: the simulation process makes copies of itself at appropriate points in time, each copy evolving as a standard Monte Carlo process.

The reason why these techniques have been developed in the Monte Carlo setting and not in the general area of discrete event simulation, is that for them to be efficient, the problems must verify some properties that need well defined mathematical frameworks, typically particular classes of stochastic processes. For instance, many fast simulation techniques of the Monte Carlo type have been proposed for helping the simulation of Markov models. Other methods are designed to estimate measures associated with standard queuing systems, etc.

This paper explores the ability of one of these acceleration methods, the splitting approach, to work in the general context of discrete event simulation. The idea is to see what happens when this technique is applied to the estimation of some low probability associated with a communication system using a generic discrete event simulation program. We show that good accelerations can be obtained in this way, and that the related software engineering problems can be solved in a straightforward manner.

The paper is structured as follows: after an introduction to the importance splitting simulation techniques in section 2, we describe in section 3 the model we use to illustrate the proposed approach. Section 4 analyses our techinique and how it was implemented for this particular model. Finally, section 5 presents simulation results and section 6 proposes some conclusions and directions for future research work.

[†] Supported by the Mexican National Council of Science and Technology (CONACyT) and by the Centre Regional de Œuvres Universitaires et Scolaires (CROUS) de Rennes.

2 Introduction to Splitting

The general framework of this technique is well represented by the problem of estimating low probability values. In order to improve the performance of the standard approach which basically consists of directly simulating the dynamics of the system, many sophisticated estimators have been used for a long time in different scientific fields. One of the most important techniques in this area is sometimes called importance splitting. It can lead to very reduced simulation times and it has the particular property that it is quite simple to implement. In the communications area, variations around this idea are receiving a considerable attention from the research community, in particular after the publication of the Restart method (Repetitive Simulation Trials After Reaching Thresholds) in 1991 [12, 13]. A similar technique was studied later by researchers from IBM and Columbia University under the name of Splitting [4, 5].

As with the importance sampling approach, the splitting techniques try to make a better use of the computer time, avoiding to waste resources simulating the system when the event of interest is not very likely to happen. To do that, the basic idea is to keep the simulator most of the time in states which are "close", in some way, to the event of interest. This helps the simulated paths to hit the rare area of the state space. More specifically, the state space of the underlying stochastic process is partitioned in some appropriate manner, and multiple independent and statistically identical paths are simulated each time a partition is entered. If the partition is well chosen, all the effort is distributed equally over the different levels and the same precision is achieved for all them, which can be shown to lead to optimal gains [4]. This should be opposed to the situation that occurs frequently when a standard estimator is used: since the event of interest is rare, most of the effort is done on states that are far from the "interesting" ones. Therefore, we obtain a very precise estimator for the region we are not interested in and a bad estimator for the relevant one. The main advantage of splitting over importance sampling is that the parameters that control the simulation process are much easier to set. For this reason, it seems more appealing to consider the former for its incorporation in a general simulator.

We want to show here how the splitting approach can be applied to successfully estimate probabilities in the area of network modeling, even if no strong mathematical hypothesis are assumed. We also illustrate how interesting gains can be obtained with a low implementation effort, and we give some insight on how the technique can be applied. We do not exactly use the Restart algorithm, nor Splitting: the variation we implemented is more general because we accept that more than one level can be crossed in one transition and that hits can be produced from all levels. Although this method was found to work very well, more formal studies are still being carried on to obtain general application rules.

3 An application example

Let us consider the problem of having a number of audio sources sharing network resources among them and with other traffic. Real time applications such as voice and video are rather restrictive to the end to end delay and delay variations (*jitter*) they may accept, while they can tolerate some packet losses. However, best effort service based packet switched networks like the Internet, introduce random delay variations in the information they transport. To cope with this variability, the receiver may buffer incoming packets and defer their delivery for a given amount of time called the *play-out delay* or *play back point* [3, 8]. If a packet arrives later than its play-out delay, its information is of no use anymore and the packet is discarded.

For interactive applications, the play-out delay can not be arbitrarily long: a limit of 400 msec is often considered acceptable for interactive voice [8, 10]. While some applications use a fixed duration play-out delay, recent proposals show adaptive mechanisms that cope with the dynamic variations of network delay [2, 8].

For the experiments, we work with the model depicted in figure 1. It represents a two-hop network transporting the voice traffic. Node A represents the access router in some local area network, shared by the voice sources and other traffic. Its link rate is 2.048 MB/sec (1.920 MB/sec effective rate) and it has a buffer size of 128 KB. Node B represents the backbone network with effective link rate of 3.840 MB/sec and a buffer size of 512 KB. Both nodes have a First Come First Served service policy.

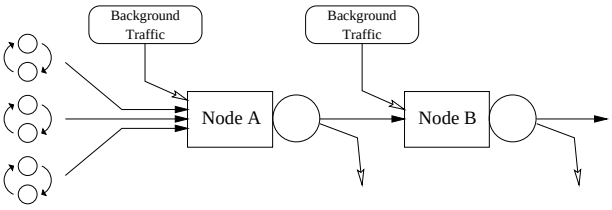

Figure 1: *two-hop network model analyzed.*

Packets generated by the voice sources traverse both nodes before reaching their destination. Background cross-traffic in each node enters the network and leaves it immediately. Background traffic in node A is about 10% of the link capacity while in node B it varies from 60% to 80%: node B is thus the bottleneck node in the model.

We use a popular On-Off model to represent the audio sources [9, 10]. *On* and *Off* periods have mean durations

respectively equal to 0.352 sec and 0.650 sec. During the *On* period, fixed length packets are periodically generated as if they were produced by a 32 KB/sec *adaptive differential pulse code modulation* (ADPCM) codec; packet sizes of 64, 128 and 256 bytes plus 40 bytes of RTP/UDP/IP header overhead are considered in the simulations.

Concerning the exogenous traffic, packet sizes are derived from a discrete distribution that intends to match the mix of "small" and "big" packets observed in various Internet measurements [1, 10, 11]. Packets of size 40, 512, 1500 and 9180 bytes are generated with probabilities 0.5, 0.25, 0.24 and 0.01 respectively. The arrival process is assumed to be Poisson and the mean interarrival rate is varied to provide different network loads.

4 The implemented accelerating technique

In order to estimate probabilities for the model under study, a fast simulation method was incorporated into an available discrete event driven simulator. Our main concerns were the speed of the simulation, the coding effort and the calibration of the method's parameters.

As was stated in the introduction, to estimate with precision the packet loss probability, it is desirable that during the simulation, these losses occur more frequently. In our model, they can be produced due to packet discarding in any of both queues or due to late arrival to the destination.

We present now the essence of the accelerating algorithm: to begin with, thresholds for the chosen splitting variable are fixed. The splitting of paths is going to be triggered when traversing them. The simulation begins normally, as a standard Monte Carlo one. When the splitting variable up-crosses one level, the discrete event simulator acts in a non-standard way: it makes replications of its internal variables and continues the simulation of each copy in parallel. Each one is simulated using the same code but with independent sequences of random numbers [1].

After having simulated every path, the results from all of them are collected and adjusted to reflect the change in the way the simulation was conducted. The variation in the hitting probability and running time depends on the correct placement of the thresholds. An important decision is then, how to structure the sets that will generate the splits, since the obtained gain will depend on this election. Intuitively, this structure, which we will call partition, helps to attract the state of the simulation to the region where more hits (packet losses) are produced. In our case the state space of the model was not infinite but huge, making it ummanegeable. One of our goals was to obtain reasonable gains without an elaborate detailed analysis of the model, for obvious efficiency reasons. Therefore, we decided to consider "simple" partitions, that is, partitions of the state space generated by a set of non overlapping intervals of a specific model's variable. Some of the obvious candidates for this were the occupancy level of each queue and the measured delay of the successive voice packets. Our election was the packet level at the second queue, because in the model this node is the bottleneck of the system. This variable, as a function of time, usually varies more smoothly than the delays and that's why it was preferred. The losses are correlated with the level of the queue because a higher level of the queue makes more probable that a packet arrives late. On the other hand, a small level does not imply that losses cannot occur since large delays can be experimented in the first queue. If we had taken the delay as the control variable, a similar problem would have occurred: even if there are short delays, losses can still happen (for example because the buffer size is small). However in some instances of this problem the second option might make the gains higher.

Let us explain the method with more detail. Each of the n sets in the partition will be named A_i or merely level i. Therefore, for our partition structure, A_i contains the states of the model where the second queue level is between two prefixed thresholds (the limits of the interval). A maximum simulation time T, when a single simulation will stop, is defined. In order to estimate the variance (and thus, to build the confidence interval) of the calculated loss probability, the period $[0, T]$ will be simulated many times. The simulation begins normally with a first path starting from a state belonging to A_0. Eventually, when this path enters the first subset A_1, the simulation state will be saved. Using this saved state, multiple paths are cloned and continued from that point in space and time. These new paths are simulated independently until they finish. They will finish either when they go out from the starting level or when the maximum simulation time is reached. More precisely: N_i paths begin from level i when a former path enters that level from level $i-1$; if one of these new paths hits level $i+1$, it generates N_{i+1} splitted paths; otherwise it finishes when it goes out of the starting level i towards level level $i-1$. All the paths in the simulation share that common behaviour. When a rare event is produced the multiplicative inverse of the cumulated splits applying to that path is added to the probability estimator (see below), and the path continues without any change.

As the simulation could end prematurely —before the scheduled time T— the last path of each split [2] is permitted to continue after going out from its starting level, as is done in Restart. Then the continuation of level i path acts as if

[1] As a software engineering detail, it is worth noting that the different paths are not needed to be run at the same time; for instance our simulator runs them sequentially instead, using recursion.

[2] Or any one since they all have the same properties.

it were a level $i-1$ path. That is, it may split again on entering level i or otherwise it will finish when going out from level $i-1$, unless it were again the last one of those paths.

At this point, it's natural to ask how many copies should be done at each split and in which way the limits of the intervals that form the partition should be picked out. It was derived in [13] that (some of) these techniques are optimal when the probability p_i of being in level $i+1$ conditioned to the model being over level i is close to e^{-2}. If that is not possible for the model, then at least it is convenient that the number of generated paths N_i at each split of level i is close to one divided by p_i. A consequence of this rule is that the number of levels must be $-\frac{1}{2}\ln p$, p being the probability we are estimating, or as near as possible. When the order of magnitude of the probability to be estimated is known, the number of levels can be selected before the simulation. In our case, we did not know it and a pre-simulation was required.

Taking into account the last result and the structure of the partition already established, the number and placement of the actual intervals were chosen in a straightforward way using pre-simulation (for another example of pre-simulation used with Petri Nets models see [6]). We used a divide and conquer algorithm and interpolated the conditional probability of being one level higher for the different values the control variable could take. The algorithm found the optimal partitions of the queue levels for each instance of the model. The number of splits was fixed to 7, the required constant for the optimality.

After having determined all the parameters, the incorporation of the method to the existing discrete event simulator had to be done. We needed to control the new paths, the instants of creation and the terminations. Also, the behaviour when hits were produced had to be modified to reflect the change in the estimator. For the control of the new paths, a component that has the capability of cloning the simulator state at a given time was added. For path control the only change that we did, was adding the splitting logic to the main control loop of the simulator. That logic consists only of detecting when new paths need to be started and when existing paths had to be finished. In the first case, N_i new states are cloned and the simulator is invoked recursively. When the later occurs, statistics are updated and the path is simply discarded.

The estimator used for the loss probability is the quotient between the expected number of losses in a time unit and the expected number of packets emitted in the same period:

$$\hat{p} = \frac{\hat{\Pi}}{\hat{L}}.$$

The numerator counts rare events and for that reason we applied our accelerated simulation technique to estimate it. Because hits can be produced from all the levels, we estimate $\hat{\Pi}$ as the sum of $\hat{\Pi}_k$, where $\hat{\Pi}_k$ denotes the estimator for the number of hits produced from level k. The denominator posses no problems in its estimation and was calculated with standard simulation as the total produced packets over the total simulation time.

To calculate the hits from each level, we must adjust the standard estimator with the number of splits made. More hits than usual were produced, and then the correct adjustment to make it unbiased consists in dividing it by the cumulated number of splits made in levels below k. The formula is:

$$\hat{\Pi}_k = \frac{1}{T}\sum_{i_0=1}^{N_0}\cdots\sum_{i_k=1}^{N_k}\frac{L_{i_0,i_1,\ldots,i_k}}{N_0\ldots N_k},$$

where $L_{i_0,\ldots,i_k}$ is the number of packet losses that occurred in path number $(i_0,\ldots,i_k)$ or 0 if that path did not exist in the simulation.

The number of splits N_i might be made dependent on the path that generated them and this permits us to choose it during the simulation. This in turn, allows N_i to be picked out in a random way in order to get closer to the mentioned recommendation. Frequently this receives the name of randomized splitting. We omit this dependency in the formula for clarity reasons.

The continuations of the last paths make the indexing more complicated because the path indices can be repeated after them. Note that this does not introduce any problem even if two paths with the same index co-exist in the same simulation time.

In fact, in this model it is possible that a transition transports the system more than one level up. This is treated in the same described way: the system detects that the new level was entered and generates and simulates the cloned paths. It can be verified that even with this generalisation, the estimator remains unbiased.

5 Simulation experiments

We consider only the stochastic queuing delay experienced in the network nodes. Other components of the overall delay such as the propagation delay, coding, decoding and packetizing of voice samples and switching delay, are not included explicitly in our models; they are globally encompassed by limiting the play-out delay to values up to 300 msec.

We ran different scenarios varying the number of voice sources, the packet size, the background load and the play-out delay threshold, and collected statistics about the packet loss, end to end delay and delay jitter. Below we present some of our results.

Figure 2 shows the packet loss observed under several conditions: we varied the number of voice sources from 50 to 250 and the background load of node B from 60% to 80% of the link capacity. The payload size was 64 bytes and the play-out delay 200 msec.

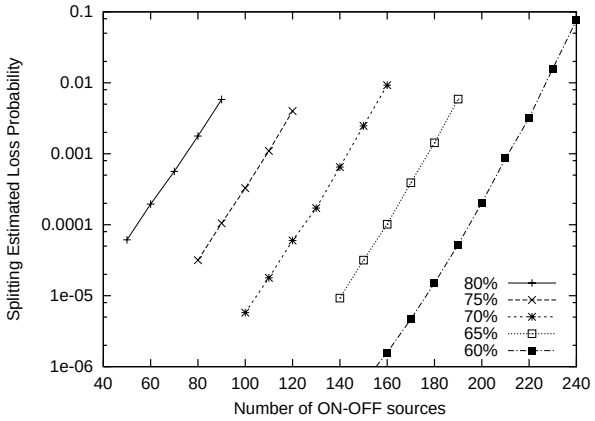

Figure 2: *loss probability vs. number of sources for different intensities of background traffic and a constant precision of 2% in the relative error of the estimator.*

Observe how the packet loss grows exponentially with the number of sources for a given background load (the vertical axis is logarithmic). As expected, we can see how for a given loss target, the number of competing audio sources is inversely proportional to the intensity of the background traffic.

In figure 3, we plot again packet loss against number of voice sources this time for different payload sizes: 64, 128 and 256 bytes. The play-out delay is fixed to 100 msec and the background traffic accounts for 70% of node B's capacity.

We can note that for a given packet loss, the number of competing sources can be increased by augmenting the quantity of coded information stored in a single packet. With a bigger packet size the header overhead is reduced, thus increasing the network efficiency: at 64 bytes of payload the header constitutes 38.5% of overhead, while it is reduced to 13.5% when the payload is 256 bytes. However, a bigger packet size also means a bigger packetizing delay, which in turn increases the overall end to end delay. In defining an optimal packet size, these factors should be weighted.

Figure 4 shows the gains that were obtained using the method for a setting where 60% background load, packet payload of 64 bytes and timeout delay of 200 msec were used. For this chart, the gains were calculated dividing the simulation time needed for a 2% precision using a standard estimator by the accelerated one. At the left of the graph, where the number of sources makes the loss probabilities

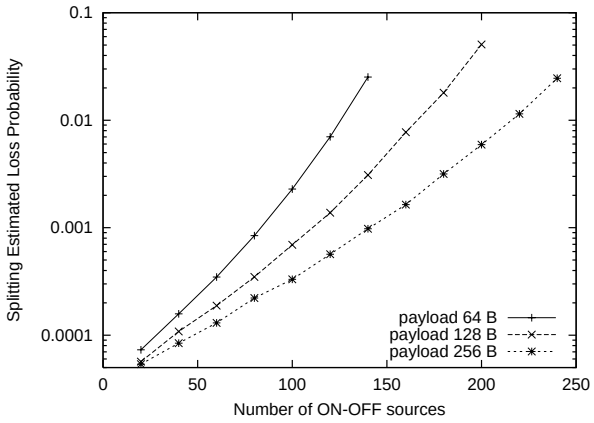

Figure 3: *loss probability vs. number of sources for different packet sizes and a constant precision of 2% in the relative error of the estimator.*

be in the order of 10^{-6}, we have high gains. As expected, when the number of sources grows, the loss probability begins to grow too (see the last series represented in figure 2). and the gain is not as good as before.

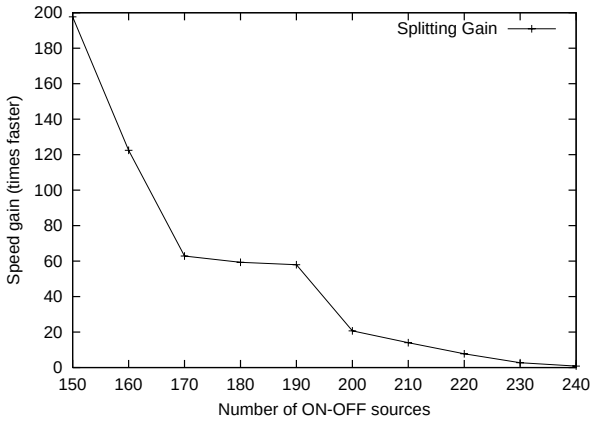

Figure 4: *observed gains due to this acceleration technique for the setting with 60% background load, packet payload size of 64 bytes and timeout delay of 200 msec. The range of the loss probabilities for the points represented in this figure goes from* 10^{-6} *to* 10^{-1}.

Recall that the type of acceleration technique considered in this paper aims to obtain a gain in efficiency in the case of rare events. When those interesting events are not rare, the splitting techniques (and also the importance sampling schemes) are not effective. However this is not a problem, since the standard method is efficient enough to deal with those situations. Nevertheless, we were able to observe good gains even when not very low probabilities arose.

6 Conclusions

The main objective of this paper is to show that some techniques used to accelerate the simulation of a system represented by specific stochastic processes (for instance, Markov models), may also give good results when applied to a general discrete event simulator. These techniques, namely the importance splitting methods, make sense when the event of interest is rare. This leads to difficulties when trying to analyze it using standard techniques due to the often important execution times or sometimes to the fact that the model is untractable.

We give some ideas about how this acceleration techniques can be implemented. The method we used does not need any particular mathematical assumption about the model, hence it can be included into a general discrete event simulator. We point out that there are two main difficulties in this task: the difficulties for calibrating such a tool and an obvious software engineering aspect, because of the characteristics of the splitting approach. The paper shows that even using a simple calibration criteria, important gains can be obtained, even if not so rare interesting events turn out.

Moreover, the example used in the paper illustrates that the method can be useful in important areas such as the forthcoming new mechanisms currently being proposed by the diffserv working group of the IETF, where quite low delays and packet losses are expected.

The two areas of most interest for further research effort are, from our point of view, the software engineering aspects, in order to obtain efficient simulators, and the calibration of the tools, that is, the problem of choosing appropriate values for the parameters that control the performance of this type of estimators. This last task appears to be quite complex in a setting as general as the one considered here. It is probably excessively optimistic to expect to reach optimal values in a very generic framework, but it is of interest to look for reasonable criteria being able to help the designer in concrete classes of problems.

References

[1] J. Bolot. Characterising end-to-end packet delay and loss in the Internet. *J. of High-Speed Networks*, 2(3):305–323, December 1993.

[2] J. Bolot, S. Fosse-Parisis, and D. Towsley. Adaptive FEC-based error control for Internet telephony. In *IEEE Infocom'99*, Anchorage, Alaska, 1999.

[3] D. Clark, S. Shenker, and L. Zhang. Supporting real-time applications in an integrated services packet network: Architecture and mechanism. In *SIGCOMM'92*, pages 14–26, Baltimore, USA, 1992.

[4] P. Glasserman, P. Heidelberger, P. Shahabuddin, and T. Zajic. Multilevel splitting for estimating rare event probabilities. Technical Report RC 20478, IBM, New York, 1996.

[5] P. Glasserman, P. Heidelberger, P. Shahabuddin, and T. Zajic. A large deviations perspective on the efficiency of multilevel splitting. *IEEE Transactions on Automatic Control*, 43(12):1666–1679, 1998.

[6] C. Kelling. A framework for rare event simulation of stochastic petri nets using restart. *Proceedings of the 1996 Winter Simulation Conference*, pages 317–358, 1996.

[7] T. Kushida. The traffic measurement and the empirical studies for the Internet. In *IEEE GLOBECOM 98*, pages 1142–1147, Sydney, Aus, November 1998.

[8] S. Moon, J. Kurose, and D. Towsley. Packet audio playout delay adjustments: Performance bounds and algorithms. *ACM/Springer Multimedia Systems*, 5:17–28, January 1998.

[9] R. Nagarajan, J. Kurose, and D. Towsley. Approximation techniques for computing packet loss in finite-buffered voice multiplexers. *IEEE J. on Select. Areas in Commun.*, 9(3):368–377, April 1991.

[10] R. Puigjaner. Performance modelling of ATM networks. In *XXII Conf. Latinoamericana de Informática*, pages 3–7, June 1996.

[11] K. Thompson, G. Miller, and R. Wilder. Wide-area Internet traffic patterns and characteristics. *IEEE Network*, 11(6):10–13, November 1997.

[12] J. Villén Altamirano and M. Villén Altamirano. Restart: a straightforward method for fast simulation of rare events. In *Proceedings of the 1994 Winter Simulation Conference*, pages 282–289, San Diego, CA, 1994.

[13] J. Villén Altamirano and M. Villén Altamirano. Restart: An efficient and general method for fast simulation of rare events. Technical Report 7, U. Politécnica de Madrid, 1997.