The Statistically Correct Approach to Distributed Simulation of ATM Networks

C.D. Pham and S. Fdida
Laboratoire MASI
Université Pierre et Marie Curie, 4 place Jussieu 75252 Paris Cedex 05, France. Telephone: (33-1) 44-27-75-12. Fax: (33-1) 44-27-62-86. e-mail : {pham,fdida}@masi.ibp.fr

Abstract

We introduce a statistically correct approach in which events can be processed in partial time order. Upon reception of an outdated message, a process does not always attempt to cancel the bad computations it has performed since the arrival of this message but uses local estimations instead. These local estimations are based on saved historical information. A control should decide whether to use local estimations or the ultimate solution of rolling back that should only be done when the use of estimations may alter too much the results. To validate our approach, a distributed simulation of an ATM network is performed applying this new idea. We focus on the mean buffer length of each switch outputs and find the results very close to those provided by a correct simulation.

Keywords

Distributed simulation, Optimistic scheduling, Local estimations, ATM networks

1 INTRODUCTION

Distributed simulation often requires a physical process of a given system to be simulated by a Logical Process (LP). The interactions between the different physical processes are represented by timestamped messages exchanged between LPs. The problem of synchronization arises since the different LPs may advance at different rate. A time error occurs when a message arrives at a receiving LP and is outdated, or old, according to the local virtual time. Existing mechanisms for parallel simulation must ensure correct synchronization between LPs in order to obtain sequential-like simulations. These mechanisms

fall in two categories: conservative and optimistic. Conservative algorithms avoid all cases of time errors (Chandy,1979). They require that execution of an LP is halted until it is certain that no time errors could happen any more. Unfortunately, this simple approach could lead to a deadlock, so null-messages that carry no physical signification are used to advance artificially the simulation time in order to avoid the deadlock of processes. On the other hand, optimistic algorithms allow LPs to always process available input messages, but implement a rollback mechanism in order to recover from time errors that can now occur (Jefferson,1985). Such mechanisms are required because these traditional approaches aim at preserving causality and so messages have to be processed in increasing timestamp order.

Ensuring causality compels conservative approaches to work with static configurations. Therefore the simulation of large systems is difficult. In addition, a lot of null-messages is necessary for deadlock avoidance when the number of real messages in the system is small. This high null-message/real-message ratio dramatically affects the performance of the simulator. For an optimistic approach the rollback overhead greatly limits the performance of the protocol. Moreover, the well-known instability of the Time Warp cancellation strategy introduces a high risk of cascaded rollbacks. A good survey of conservative and optimistic approaches, and their performance, can be found in (Fujimoto,1990). Briefly, one can improve the performance of these protocols by adding some optimism to the former while limiting the too much of optimism and minimizing the overhead of the rollback mechanism in the latter. However, all these improvements do not call the causality constraints into question. In this work, we investigate a different approach that relaxes the causality constraint and allows inaccuracies to occur.

In the terminology defined in (Reynolds,1988), accuracy requires that events be ultimately processed in increasing timestamp order. If this is not the case then the protocol is said to be inaccurate. Time Warp is accurate because its rollback mechanism corrects the bad computations. In (Theofanos,1984) the author showed for simulation of queuing networks that bad computations do not affect dramatically mean value statistics. Going a step farther leads us to think that in many cases some useful results can be obtained without the total time ordering established by conventional approaches. This paper introduces the inaccurate *statistically correct approach* in which some events can be processed in partial time order without rollbacks. In this approach, upon reception of an outdated message a process does not always attempt to undo the bad computations it has performed since the arrival of this message but uses local estimations instead. A control should decide whether to use local estimations or the ultimate solution of rolling back. The heavy rollback mechanism should only be used when utilization of local estimations may alter too much the results. Thus, the results obtained by a simulation are *statistically correct* when compared to those provided by a conventional approach.

In order to demonstrate our approach, a distributed simulation of two ATM networks configuration, with exponential and ON/OFF traffic sources, is performed applying this new idea. We focus on the mean buffer length of each switch outputs and find the results very close to those provided by a correct simulation.

This paper is organized as follows. The next section details the new approach that introduces the idea of a statistically correct simulation. Section 3 presents the test cases and the computation of local estimations. Section 4 presents the preliminary results. We conclude in section 5 with indications of future directions for this work.

Proceedings of the 6th IFIP Conference on Performance of Computer Networks, pp352-366

2 THE STATISTICALLY CORRECT APPROACH

Now we need to give some basic definitions in order to introduce the statistically correct approach. The term simulation refers to distributed discrete event simulation and we assume that the reader is familiar with Logical Process (LP) and process local virtual time (LVT).

2.1 Definitions

Dfinition 1 (Very-correct) *A very-correct simulation is a simulation in which events are processed by an LP in increased timestamp order with no time errors.*

Dfinition 2 (Correct) *A correct simulation is a simulation in which some time errors have occurred but have all been corrected by a rollback mechanism.*

Dfinition 3 (Statistically correct) *A statistically correct simulation denotes a simulation in which some time errors have occurred but not all of them have been corrected.*

Dfinition 4 (Rollback-free sequence) *Let φ be a finite sequence of n events $< e_i, t_i >$ where t_i is the timestamp of event e_i. A sequence is said to be* rollback-free *if $\forall i, t_{i+1} \geq t_i$. On the other hand, a* k-rollback *sequence is a sequence where there are k t_i such as $t_i < t_{i-1}$. In extension, a* rollback sequence *is a k-rollback sequence where $k \geq 1$.*

A conservative algorithm typically produces very-correct simulations, whereas an optimistic one like Time Warp is more likely to produce correct simulations. Our statistically correct approach is undoubtedly on the optimistic side and can be classified as "very optimistic". Consequently we will compare its behavior to the Time Warp algorithm.

When developing our new statistically correct approach, we find very interesting to give it a tunable behavior. Therefore we wanted to meet the following requirement:

- A statistically correct approach must be able to produce correct and statistically correct simulations.

To meet this requirement, a statistically correct approach must establish a control in order to span the continuum of correctness from correct to statistically correct. Let Φ_c and Φ_s be the controls for obtaining correct and statistically correct simulation with a statistically correct approach. Now we can introduce the notations TW_φ and $ST_\varphi(\Phi)$ to denote the simulation of an LP using respectively the Time Warp protocol and our statistically correct approach applying control Φ, this under the sequence φ. Simulation of an LP under a sequence φ means that the LP successively processes event i of φ, $i = 1..n$. For notation simplicity we will write $a = b$ if the result of simulation a is the same that the result of simulation b. Finally the following statements can be written:

1. if φ is a rollback-free sequence then TW_φ and $ST_\varphi(\Phi)$ are both very-correct simulations whatever the control Φ is. We will have $TW_\varphi = ST_\varphi(\Phi)$.
2. if φ is a rollback sequence then TW_φ and $ST_\varphi(\Phi_c)$ are both correct simulations and we have $TW_\varphi = ST_\varphi(\Phi_c)$. On the other hand, $ST_\varphi(\Phi_s)$ is a statistically correct simulation. In most cases, we will have $TW_\varphi \neq ST_\varphi(\Phi_s)$.

Proceedings of the 6th IFIP Conference on Performance of Computer Networks, pp352-366

For an optimistic approach, rollback is the sole means to obtain a correct simulation. Therefore the purpose of the control Φ is to decide whether to rollback or not, when a time error occurs. It is obvious that Φ_c is the Time Warp's control requiring a rollback at each time error, whereas Φ_s can be based on some assumptions triggering a rollback only on some specific cases.

Since a rollback-free sequence does not generate time errors, simulations using any synchronization protocol, existing or to come, under such a sequence are all very-correct simulations. Statement 1 says that under a rollback-free sequence our statistically correct approach provides results that are identical to those provided by a conventional Time Warp approach, independently of the control. Now if φ is a rollback sequence the Time Warp algorithm produces correct simulations by means of its rollback mechanism. On the other hand, we wanted our statistically correct approach to produce statistically correct simulations depending on the control Φ. This means that upon reception of a bad message the heavy rollback mechanism is not systematically used to cancel the bad computations. Of course in order to obtain coherent results, alternatives to rollback must be found. We propose utilization of local estimations and a different time evolution scheme as described in the next sections.

2.2 Local estimations

The life of a process consists of receiving and processing incoming messages. Processing a message often changes the process state and may generate output messages to other processes. When a bad message m_{past} with timestamp t_{past} arrives at time t_{now}, Time Warp has to roll back. This is done by restoring the process state before t_{past}, by jumping back to time t_{past} and by sending anti-messages to cancel messages produced by optimistic scheduling. Only then all events since t_{past} are re-executed. This is required because the processing of m_{past} may change the evolution of the process making all states and all produced messages since t_{past} erroneous.

There is no way but to roll back if one wants to correct the bad computations propagated by bad produced messages. Now if one accepts the errors introduced by not cancelling produced messages then there is no need to roll back, local estimations can be used instead to avoid the overhead of rolling back. This is the key idea expressed by our statistically correct approach. Estimations should be designed to produce correct output messages resulting from the processing of m_{past} and to re-evaluate the process states from t_{past} to t_{now}.

We propose local estimations based on historical information. Such historical information can be included within the process state and periodically saved in a Time Warp-like fashion. When a past message m_{past} arrives, the process virtual time does not change and saved historical information before t_{past} are used to process m_{past} and to generate output messages. Re-evaluation of saved states is also performed to take into account the changes introduced by m_{past}. The way these estimations are computed is quite application dependent. We give in section 3 the estimations used for the simulation of an ATM network.

2.3 Proposed controls

The main guideline for the statistically correct approach is to reduce the cost of process synchronization while providing useful results. Use of local estimations has the advantage of avoiding rollbacks but can unfortunately guess wrong and alter too much the results. One solution is to establish a control that decides when the process should roll back and when it can use local estimations.

At this point, we must mention that our correctness metric uses the results obtained by a very-correct or correct simulation–using respectively a conservative and an optimistic approach–as a reference. Actually, in deterministic simulations one can obtained identical results from one execution to another if the

sequence of events applied at the input is the same for all executions. So, when all simulations are deterministic, we can define a simple correctness metric by comparing the results provided by our statistically correct approach to those provided by a conventional approach. If both results are identical then we say that maximum of correctness is achieved. In the same manner, a cost metric can be defined by taking the cost of the Time Warp control as a reference.

Therefore, at the uppermost bound of correctness and cost, e.g. $correctness = cost = 1$, we have the Time Warp control Φ_c that provides correct simulations. To obtain "best effort" results, e.g. $correctness \in [0, 1]$, but at a minimum cost, e.g. $cost = 0$, we have:

- *The Null control*, noted Φ_s^{null}. With the null-control a process always uses local estimations and does not use rollback at all.

Between these two end-points, a control should guess when the use of local estimations may give bad results. Let us define the distance of an error due to an incoming message $< e_i, t_i >$ as $d = LVT - t_i$, LVT being the local virtual time of the receiving LP. Some observations suggest to use this distance as a criterion for rolling back. This is motivated by: the more this distance is, (1) the more the estimations may guess wrong because too much changes may have occurred, (2) the more wrong messages have been sent and (3) the more the overhead of state re-evaluation may outpace the overhead of a rollback. This leads us to define the following control:

- *The Distance-Based control*, noted $\Phi_s^{d_0}$. With the distance-based control a process uses the rollback mechanism if the distance d of an error is greater than d_0.

Note that $\Phi_c = \Phi_s^{d_0}$ when $d_0 = 0$. Figure 1 shows the continuum spanned by the statistically correct approach. Under a general sequence φ one can achieved maximum of correctness but also at maximum cost

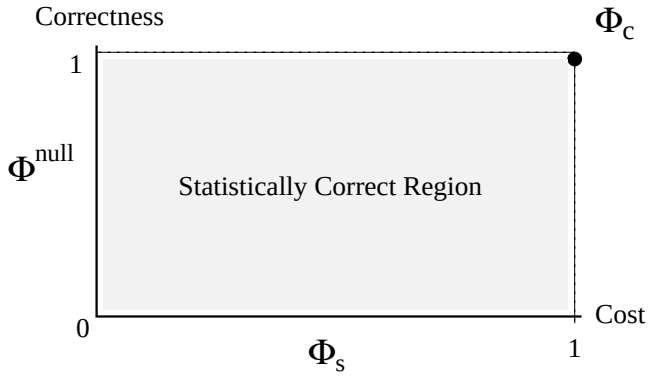

Figure 1 Spanned continuum of correctness from correct to statistically correct for the statistically correct approach.

by using $ST_\varphi(\Phi_c)$. This is equivalent to a Time Warp simulation. On the other hand, $ST_\varphi(\Phi_s)$ provides the possibility to reduce the simulation cost while keeping a reasonable level of accuracy. Simulations on the y-axis always use the null-control. If maximum of correctness is however obtained, then the simulation is a very-correct one–it is equivalent to say that the simulation was run under a rollback-free sequence. Simulations on the x-axis must be rejected since they use a Φ_s control that introduces some cost but

produces no correct results! We believe that practically this should not happen. Now, the shaded region represents simulations using any combination of Φ_s control and φ sequence. Of course one would rather like to obtain simulations in the upper-left corner that achieve maximum of correctness at a minimum cost.

2.4 Local virtual time evolution

In the Time Warp algorithm, a time error causes the process virtual time to jump back to the time of the error in order to run forward again in the right sequence. Now since local estimations are used instead, when a time error occurs the process virtual time need not to jump back. The processing of a past message is done in what we call an *estimation environment*. The first time error blocks the process virtual time at time t_{now} until a message with timestamp t greater than t_{now} is received. Since the virtual time is blocked, subsequent messages with timestamps lesser than t_{now} can also produce time errors. We call this kind of error *secondary time error* as opposed to *first time error*.

Figure 2 shows the possible evolution of virtual time for (a) a conventional approach and (b) the statistically correct approach. In the last case a white point indicates the processing of a past message,

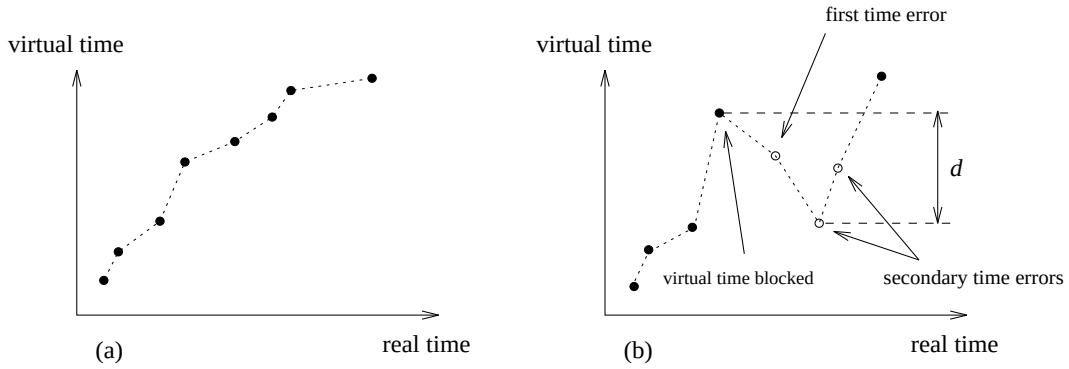

Figure 2 Evolution of local virtual time for (a) a traditional approach and (b) the statistically correct approach.

relatively to t_{now}, in an estimation environment. With the $\Phi_s^{d_0}$ control the distance d of a time error, first or secondary, can not exceed a threshold d_0 while it is unbounded with the Φ_s^{null} control.

3 TEST CASES AND LOCAL ESTIMATIONS

3.1 Test cases

Network configuration a consists of an ATM switch with three traffic sources (figure 3). The switch has output buffers and the following switching strategy: in one time slot each cell that arrives on an input port is switched to the corresponding output. The choice of the output follows some fixed routing probabilities. Only one cell can arrive per time slot on an input but severals cells can be switched to one output in the same time. In this case they are buffered in the output queue. On the other hand, for all outputs

the head cell in the queue is transmitted on the link in each time slot. In this model, the switch has three inputs and two outputs. Traffic sources continually generate cell reception messages $< RECV, t >$ according to a given description. Upon reception of a $< RECV, t >$ message, the switch schedules for itself a $< SEND, t_d >$ event where t_d is the date of departure of the cell. A cell from a given input $i = 1..3$ is switched to an output $j = 1, 2$ with probability p_{ij} and sent out by scheduling at the receiver a $< RECV, t' >$ message. We set for the simulation $p_{ij} = 0.5$ for all i and j. In configuration b a network

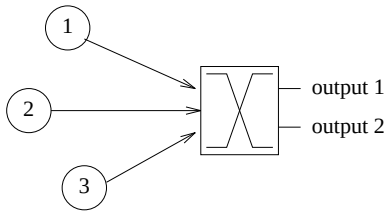

Figure 3 Network configuration a.

of five switches is simulated (figure 4). Only one traffic source per switch is considered that generates cell reception message only up to time $t_{endsource}$ lesser than the simulation time. Since there is no sink process, this limits the number of cells in the network. When a switch a receives a cell from another switch, it sends it to the output $j = 1, 2$ with probability p_{aj}. When traffic sources do not generate cells anymore, the number of cells in the network remains constant and all switches keep sending existing cells to each others until the end of the simulation. We set for the simulation $p_{aj} = 0.5$ for all switches a and output j, and $t_{endsource} = 40$. Propagation delay between two switches takes 2 time slots. Three test

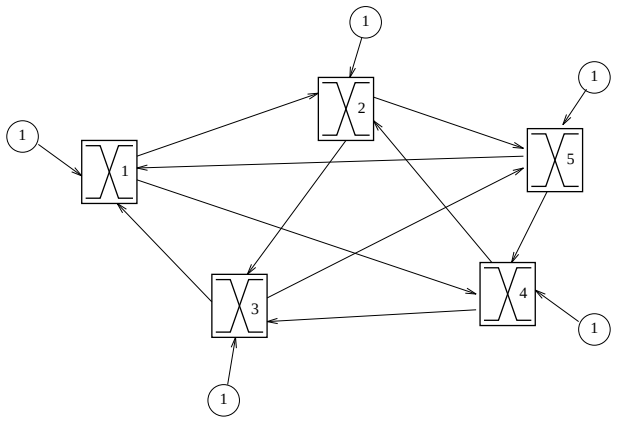

Figure 4 Network configuration b.

cases are considered and listed below:

1. Test case 1 considers the network configuration a with exponential sources. The three traffic sources continually generate cell reception messages. In order to generate at most one cell per time slot on a link, the cell inter-arrival time δ is computed as follows: $\delta = ceil(\tau)$ where *ceil()* is the rounding toward positive infinity function and τ an exponentially distributed variable.

2. Test case 2 considers the network configuration b where all sources are exponentials as described above.
3. Test case 3 considers the network configuration a with three ON/OFF sources. Silent and burst periods are both exponentially distributed. While in a burst period, a source generates a cell at each time slot.

The state of a switch process is defined by a vector $S = (o_1, o_2, last_1, last_2, n_r, n_s)$. o_j and $last_j$ are respectively the number of cells waiting in the output queue j and the *last time* that a cell has been switched to output j. n_r and n_s are respectively the number of received and sent cells. State saving for local estimations purpose is done after every processing of an $< RECV, t >$ event. The processing of self-generated events such as $< SEND, t >$ does not cause the process to save its state. This choice keeps the number of historical information to a minimum.

3.2 Computation of local estimations

Let L_s be the list of saved states and $S_i = (o_1^i, o_2^i, last_1^i, last_2^i, n_r^i, n_s^i)$ be the ith saved state. The timestamp of S_i is noted T_i. Index i increases as T_i increases. Upon reception of an outdated cell reception event $< RECV, t >$, $t < LVT$, we use the following estimation algorithm at the switch process:

1. Gets the first state before time t from L_s. The retrieved state is S_i with $T_i \leq t < T_{i+1}$.
2. Determines the output j and derives the date of departure t_d of this new cell by computing its waiting time. Then schedules an $< SEND, t_d >$ event for itself.
3. Inserts the new state in L_s before the retrieved state S_i (so at index $i+1$; all state indexes are incremented).
4. Re-evaluates all states S_k with $k > i+1$.

Since historical information are saved after every $< RECV, t >$ event the retrieved state is exactly the state the first past message would have find if it was received in sequence. For subsequent bad messages the retrieved state can be a re-evaluated one. We explain steps 2 to 4 in what follows.

Let t_w be the waiting time of the cell before the beginning of its processing at the output j. The date of departure t_d of the cell on output j is computed as follows (time is expressed in time slot):

■ Step 2
$$\begin{cases} \text{if } t = last_j^i \text{ then } t_w = max\{o_j^i - 1, 1\} & (1) \\ \text{else } t_w = max\{o_j^i - 1 - (t - T_i), 0\}. & (2) \\ t_d = t + t_w + 1 + 1. & (3) \end{cases}$$

Note that if $t = last_j^i$ we must have $t = T_i$ and $o_j^i \geq 1$. On the other hand, if $t \neq last_j^i$ then we can have $t = T_i$ but this is not mandatory. Since we assume that several cells can be switched from different inputs to an output queue in one time slot, relation $t = T_i$ expresses the fact that there is more than one cell switched to an output in one time slot. Relation $t = last_j^i$ is more precise since it indicates that at least one cell was previously switched to output j in the same time slot. So if $t = last_j^i$ and $o_j^i > 1$ (eq. 1) the cell must wait for all the cells in front of it in output j at time t minus the one that will already be sent when the cell arrives at the output queue (figure 5a). This because transmission of the head cell of an output also requires one time slot. Now if $o_j^i = 1$ then the cell must only wait for the cell in front of it in output j because both have been switched in the same time slot (figure 5b).

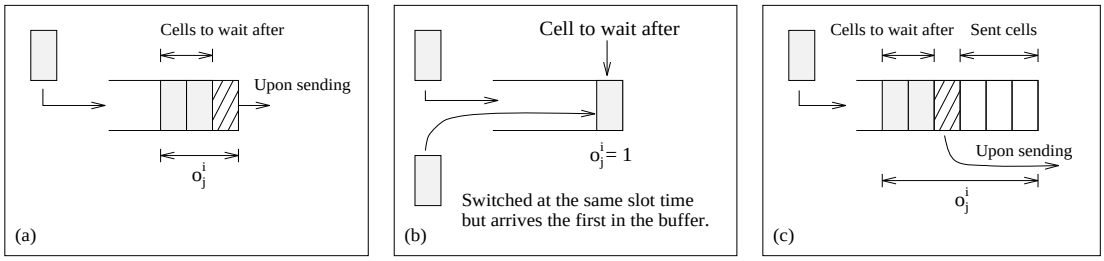

Figure 5 Estimation of the cell waiting time.

If $t \neq last_j^i$ (eq. 2) then the cell must wait for all the cells in output j at time T_i, minus those that were already sent during the interval $t - T_i$, and minus the one that will already be sent when the cell arrives at the output queue (figure 5c). Of course if the output queue is empty then t_w is null. The date of departure t_d is easily obtained by adding to t the switching time (one time slot), the transmission time (one time slot) and the waiting time (eq. 3).

The process state S_{i+1} after the processing of the $< RECV, t >$ message is saved at index $i+1$ and we have :

■ Step 3
$$\begin{cases} o_j^{i+1} = max\{o_j^i - (t - T_i), 0\} + 1. \\ last_j^{i+1} = t. \\ n_r^{i+1} = n_r^i + 1. \\ n_s^{i+1} = n_s^i + min\{o_j^i, t - T_i\}. \end{cases}$$

If $o_j^i < t - T_i$ then all cells that are waiting in output queue j have been sent during the interval $t - T_i$. In this case $o_j^{i+1} = 1$, e.g. the new cell waits alone in the output queue j. On the other hand, if $o_j^i > t - T_i$ then there are some cells left and the new one waits with the other $o_j^i - (t - T_i)$ remaining cells. In the same way, the number of sent cells is incremented by the number of cells that have been sent between t and T_i.

Re-evaluation of all states S_k with $k > i+1$ is done as follows:

■ Step 4
$$\begin{cases} \text{if } T_k - t < o_j^{i+1} \text{ then } o_j^k = o_j^k + 1 \text{ else } o_j^k \text{ is unchanged.} \\ last_j^k \text{ is unchanged.} \\ n_r^k = n_r^k + 1. \\ \text{if } t_d < T_k \text{ then } n_s^k = n_s^k + 1 \text{ else } n_s^k \text{ is unchanged.} \end{cases}$$

Inequality $T_k - t < o_j^{i+1}$ means that the new cell has not been sent yet at time T_k, so o_j^k should be incremented. On the other hand, if $T_k - t \geq o_j^{i+1}$ then the new cell has been sent and o_j^k remains unchanged. In the same manner, if $t_d \leq T_k$ then the new cell has been sent before or at time T_k so n_s^k should be incremented. Note that if $t_d \leq T_k$ then we have $T_k - t \geq o_j^{i+1}$ because both conditions are equivalent.

Table 1 Correct simulation of test 1.

Test case 1	SW1
Buffer 1	0.812
Buffer 2	0.846

Table 2 Statistically correct simulation of test 1.

Test case 1	SW1
First time error	5
Secondary time error	22
Buffer 1	$0.809 \rightarrow 99.63\%$
Buffer 2	$0.841 \rightarrow 99.40\%$

4 PRELIMINARY RESULTS AND DISCUSSION

Simulations are carried out on a network of SparcStations. Implementation is done in C using the PVM package (Geist and al.,1993) for inter-processes messages passing. In order to easily compare the accuracy of the results provided by our statistically correct approach, deterministic sources are used. This means that from one simulation to another, the sources generate the same sequence of cell reception event $< RECV, t >$. However, the number of time errors from one simulation to another can vary due to the randomness of inter-network message passing delay.

In this paper only the null-control that provides "best-effort" results is studied. The simulation time is fixed to 4000 time slots. The results show the number of first and secondary time errors that are detected, and the switch buffer occupancy. For statistically correct results, the percentage of correctness is also shown.

4.1 Results for test 1 and 2

Tables 1 and 2 show the buffer occupancy for test 1 obtained respectively with a correct simulation and a statistically correct simulation with the null-control. Table 3 and table 4 show the results obtained for test 2. Only one simulation result is shown for test 1 but for all simulations that we have performed we found the results very close to the target values obtained by a correct simulation. We can notice that the number of first time errors is small. This is easily explained by the fact that the system is very simple and that the transfer of cells is only unidirectional from the sources to the switch. This lowers the degree of interactions between the network components and then reduces the number of time errors. Good accuracy is then obtained.

For test 2 things do greatly differ since interactions between network components are much more complex. In this scenario a switch a can receive from another switch b a cell that it has originally sent. If meanwhile switch a has processed several cell reception events from the other switches then the risk of a

Table 3 Correct simulation of test 2.

Test case 2	SW1	SW2	SW3	SW4	SW5
Buffer 1	2.584	2.588	2.581	2.572	2.578
Buffer 2	2.578	2.574	2.561	2.558	2.558

Table 4 Statistically correct simulation of test 2.

Test case 2	SW1	SW2	SW3	SW4	SW5
First time error	93	109	113	98	112
Secondary time error	135	157	151	137	148
Buffer 1	2.425	2.374	2.421	2.385	2.463
Buffer 2	2.383	2.436	2.401	2.434	2.395
Correctness	93.84%	91.73%	93.80%	92.72%	95.53%
Correctness	92.43%	94.63%	93.75%	94.04%	93.62%

time error is quite high. When compared to test 1, we do find that the number of first time errors is far greater. However, and it is encouraging, the results obtained have not been dramatically altered.

For both test cases the values for the buffer occupancy provided by our statistically correct algorithm are always less than the values obtained by a correct simulation. This phenomenon was predictable since the error introduced by not roll backing consists of neglecting the possible amount of extra waiting time of previously sent cells. In the light of the results, this error is kept relatively low.

4.2 Results for test 3

Now for test 3 where sources are more sporadic, we have recorded and plotted the evolution of the number of cells in the output queue 1. Figure 6 depicted this evolution for a correct simulation and figure 7 for a statistically correct simulation–always with the null-control. We can notice a missing burst in figure 7. Since record of data is processed on-line, the only explanation is that the switch process has optimistically gone too far in the future. So when the burst arrives out of sequence the process uses local estimations and state re-evaluation to reconstruct it but does not take the burst into account in the recorded data. Tables 5 and 6 show the buffer occupancy for test 3 obtained respectively with a correct simulation and a statistically correct simulation with the null-control. When compared to the previous results for test 1, the number of secondary time errors is much more greater because the burstyness of the sources makes such errors to come in packets. Consequently the errors introduced by neglecting the possible amount of extra waiting time of previously sent cells are greater, making accuracy of the simulation a bit lower. One should note that only the cells sent during the missing burst are concerned. Unfortunately, if the length of the burst is big enough and, if a lot of cells have been scheduled during this burst (figure 8) then accuracy may drop (table 7).

In table 7 we can see that the number of secondary time errors is again much more greater. This

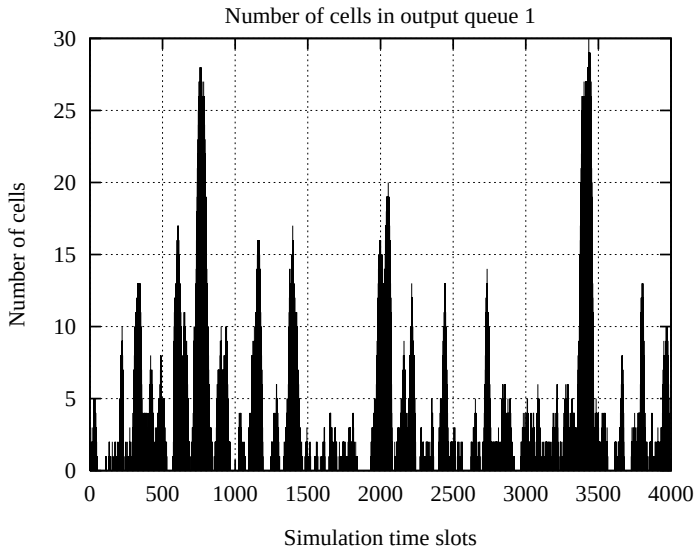

Figure 6 Number of cells in output queue 1—Correct simulation.

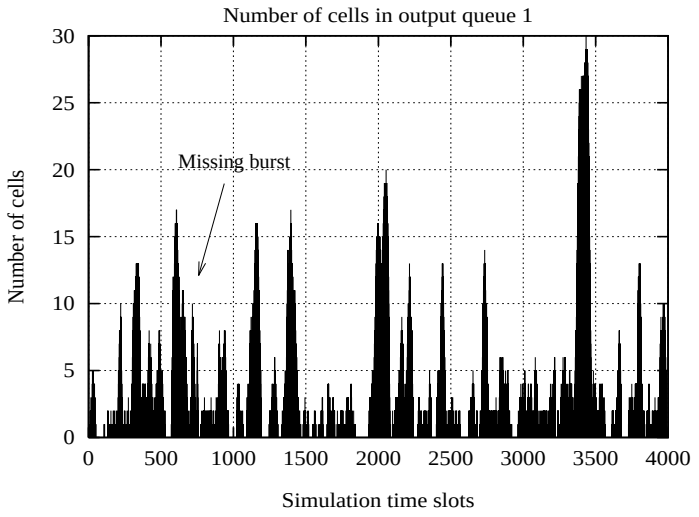

Figure 7 Number of cells in output queue 1–Statistically correct simulation.

test shows the limitations of local estimations since the results obtained differ appreciably from those obtained with a correct simulation. Actually, it is not really the number of time errors that affects the simulation accuracy but the number of cells that have been sent during a skipped burst. These cells have been scheduled to leave the switch too early and, in doing so, they affect the collected statistics.

Proceedings of the 6th IFIP Conference on Performance of Computer Networks, pp352-366

Table 5 Correct simulation of test 3.

Test case 3	SW1
Buffer 1	3.975
Buffer 2	3.938

Table 6 Statistically correct simulation of test 3.

Test case 3	SW1
First time error	2
Secondary time error	123
Buffer 1	3.682 → 92.62%
Buffer 2	3.881 → 98.55%

5 CONCLUSION AND PERSPECTIVES

In this paper, we proposed a new approach called *statistically correct approach* for distributed simulation where total correctness is not always required. This approach provides the possibility to reduce the simulation cost while keeping a reasonable level of accuracy. Early tests with local estimations give favorable results and encourage us to develop this research on. However, we have noticed that the traffic pattern can have some influence on the accuracy of the estimations and further tests should address this point deeper.

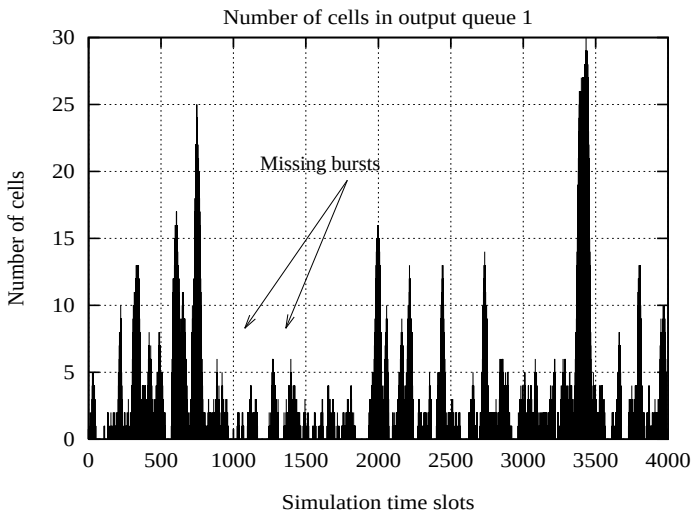

Figure 8 Number of cells in output queue 1–Statistically correct simulation.

Table 7 Another statistically correct simulation of test 3.

Test case 3	SW1
First time error	9
Secondary time error	317
Buffer 1	3.400 → 85.53%
Buffer 2	3.521 → 89.41%

The way estimations are obtained is very application dependent. We are conscious that this scheme gives better results when generated messages and process state do not change very much after the processing of the missing events. This is typically the case for ATM networks. Future works would address the use of this approach to larger network of hundreds of nodes. We believe that in order to make such large scale simulations possible in a reasonable amount of time, one has to relax the synchronization constraints and to accept statistically correct results.

We are in the process of implementing the distance-based control added with some heuristics. A lot of works remains to address performance and suitability of this control under sporadic traffic. In addition, we hope that the important issue of determining the simulation accuracy, without comparing the results, could be easier with the introduction of such a control.

Garbage collection, GVT computation and state saving frequency–what to do when there is no consistent historical information available–are also many interesting problems this approach has to deal about and that have not been discussed in this paper.

REFERENCES

Chandy, K.M. and Misra, J. (1979) Distribution Simulation: A Case Study in Design and Verification of Distributed Programs, *Transactions on Software Engineering*, **5**, 440–452.

Fujimoto, R.M. (1990) Parallel Discrete Event Simulation, *Communication of the ACM*, **10**, 31–53.

Jefferson, D.R. (1985) Virtual Time, *ACM Transaction on Programming Languages and Systems*, **3**, 405–425.

Geist, A. and al. (1993) PVM 3 User's Guide and Reference Manual, May 1993.

Proceedings of the 6th IFIP Conference on Performance of Computer Networks, pp352-366

Reynolds, P.F. (1988) A Spectrum of Options for Parallel Simulation, *Proceedings of the 1988 Winter Simulation Conference*, 325-332.

Theofanos, M. (1984) Distributed Simulation of Queuing Networks, *Master's Thesis*, The University of Virginia, January 1984.

BIOGRAPHY

CongDuc Pham Is a PhD candidate at the MASI Laboratory (Methodologie et Architecture des Systèmes Informatiques), University Pierre et Marie Curie, (Paris, France). He received a Diplôme d'Etude Approfondie (DEA) in computer systems from the University Pierre et Marie Curie in 1993. His research interests focus on parallel discrete event simulation algorithms and their application to the simulation of ATM networks.

Serge Fdida Received the Thèse de 3ème Cycle and Habilitation degrees in computer science from the University Pierre et Marie Curie in 1984 and 1989, respectively. From 1984 to 1989, he was Assistant and then Associate Professor at the University Pierre et Marie Curie. He is currently a Professor at the University René Descartes (Paris, France). His research interests are in the area of modelling and performance evaluation of computer networks and high-speed communication systems architecture and protocols. He is responsible of the "Network and Performance" team of the MASI Laboratory (CNRS). Serge Fdida is the author of many papers in the field and a book on Performance Evaluation. He was the program chairman of the IFIP High Performance Networking '94 Conference. He is a member of ACM, IEEE and AFCET.

Proceedings of the 6th IFIP Conference on Performance of Computer Networks, pp352-366