EFFICIENT PARALLEL SIMULATION WITH THE STATISTICAL SYNCHRONIZATION METHOD

Gábor Lencse
Department of Telecommunications
Technical University of Budapest
Sztoczek utca 2
H-1111 Budapest, Hungary
E-mail: Gabor.Lencse@hit.bme.hu

KEYWORDS

parallel discrete-event simulation, communication networks, performance analysis, statistical synchronization, speed-up

ABSTRACT

The transient behavior, accuracy and efficiency of the Statistical Synchronization Method (SSM) are studied in a practically important case. An existing high-speed network (two interconnected FDDI Rings) is simulated accurately. The topology, cable lengths and the offered load is taken from the real system. The results of the simulation with SSM are compared to the results of a reference simulation without SSM. The execution times of distributed and non-distributed simulations using SSM are compared to the execution time of the reference simulation (without SSM). The questions of implementation of parallel simulation with the help of SSM are also discussed. We conclude that the simulation with SSM is a bit less accurate but it facilitates a very efficient parallel simulation, which can be easily implemented and provides an excellent speedup.

INTRODUCTION

Discrete event simulation is a powerful method in the performance analysis of communication networks, digital circuits and computer systems. The simulation of large and complex systems requires a large amount of memory and computing power that is often available only on a supercomputer. Efforts were made to use clusters of workstations or multiprocessor systems instead of supercomputers, as this would be much more cost effective. The conventional synchronization methods for parallel simulation (e.g., conservative, optimistic) (Fujimoto 1990) use event-by-event synchronization and they are unfortunately not applicable to all cases, or do not provide the desirable speedup. The conservative method is efficient only if certain strict conditions are met. The most popular optimistic method "Time Warp" (Jefferson 1987) often produces excessive rollbacks and inter-processor communication.

The Statistical Synchronization Method (SSM) (Pongor 1992) is a promising alternative to the conventional methods. As with other parallel simulation methods, the model is divided into segments that typically execute on separate processors; but unlike other methods, SSM does not exchange individual messages between the segments but rather the statistical characteristics of the message flow. Actual messages are regenerated from the statistics at the receiving side. Further explanation will be given later.

SSM claims to be less sensitive to communication delay and it requires less network bandwidth than event-by-event methods. Nevertheless, it is not accurate in the sense that an event that occurred in one segment of the system does not have an immediate influence on another segment. For this reason, the method cannot be applied in some simulations, for example in the case of digital circuits but remains feasible in other classes of simulation such as the performance estimation of communication systems.

The transient behavior and accuracy of SSM were already demonstrated in an earlier paper (Lencse 1997). This paper deals with the questions of parallelisation using SSM and with the efficiency of SSM. As SSM used for distributed simulation in this paper is slightly different from SSM in the above mentioned paper, some of the earlier experiments were repeated to check the transient behavior and accuracy of the new SSM.

The simulated system is an FDDI network. First, a very accurate non-parallel simulation of two interconnected FDDI rings is done to set up a reference. One ring consists of several FDDI stations interconnected by wiring concentrators and the other one is a smaller ring of FDDI stations. The topology and the cable lengths are taken from a real system. The load is produced by measurements. Second, SSM is used between the two rings and the results are compared to the case without using SSM. Third, the simulation is executed in parallel by one processor: the simulation of the two rings is done by cooperating processes on the same processor; SSM is used between the two rings; the local virtual time of the two parts of the modell is syncronized by loose synchronisation (see later). Fourth, the simulation is executed in parallel by two processors: the simulation of each rings runs on its own processor; SSM and loose synchronisation is used between the two rings. The execution times are measured in the 4 cases and are compared.

This topic was identified as being of importance in the efficient parallelisation of event-driven discrete event simulation facilitating rapid and easy parallel implementation.

The remainder of this paper is organized as follows: first, a brief introduction to SSM is given, then the simulated system is described, next the simulation model is defined, afterwards the questions of parallelisation are considered and finally, the simulation results are presented and discussed.

THE STATISTICAL SYNCHRONISATION METHOD

A short summary of the Statistical Synchronization Method is given here.

Similarly to other parallel discrete event simulation methods, the model to be simulated -- which is more or less a precise representation of a real system -- is divided into segments, where the segments usually describe the behavior of functional units of the real system. The communication of the segments can be represented by sending and receiving various messages. For SSM, each segment is equipped with one or more input and output interfaces. The messages generated in a given segment and to be processed in a different segment are not transmitted there but the *output interfaces* (OIF) collect statistical data of them. *The input interfaces* (IIF) generate messages for the segments according to the statistical characteristics of the messages collected by the proper output interfaces. (See *Fig. 1.)*

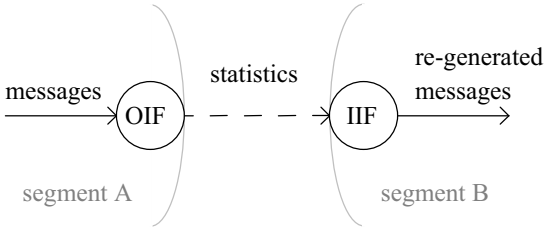

Fig. 1. An OIF - IIF pair

The segments with their input and output interfaces can be simulated separately on separate processors, giving statistically correct results. The events in one segment have not the same effect in other segments as in the original model, so the results collected during SSM are not exact. The precision depends on the partitioning of the model, on the accuracy of statistics collection and regeneration, and on the frequency of the statistics exchange among the processors.

Advantages of SSM

SSM has the following advantages compared to the other PDES (Parallel Discrete Event Simulation) methods:

- requires less network bandwidth
- tolerates communication delay better
- can be easily implemented
- requires less support from the simulation kernel
- may produce better speedup

A feasible approach can be that one implements his simulation as a uni-processor version first. After verification, he replaces the message paths on the segment boundaries with statistical interface pairs. He may run the simulation fast on a cluster of workstations and produce results that are probably less accurate than those that can be achieved without SSM but they can be produced much faster and are probably applicable for tuning the model on the basis of them. The final results are to be verified with the usual DES verification methods.

THE SIMULATED SYSTEM

To examine the characteristics of the SSM in a practically important case, a widely used communication network: Fiber Distributed Data Interface was chosen. FDDI is a 100-Mbps fiber optic network standard. (ANSI X3.139, 1987) It has a dual ring topology that can be extended by wiring concentrators. The so-called Timed-Token access protocol is applied for media access control purposes.

The aim of this simulation study was to examine SSM in a realistic simulation, so all possible efforts were made to use a simulation model that is very close to an existing network.

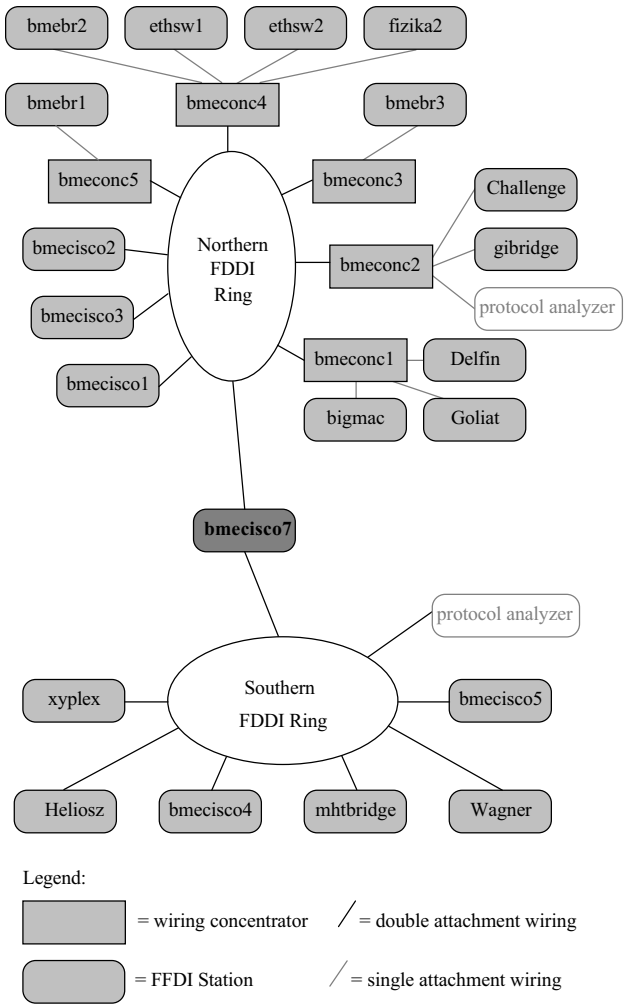

Fig. 2. FDDI backbone of the Technical University of Budapest

The FDDI backbone of the Technical University of Budapest was found to be appropriate. It consists of two rings: The Northern Ring is a university-wide network and consists of 15 FDDI stations interconnected by 5 wiring concentrators. The Southern Ring is the backbone of the Faculty of Electrical Engineering and Informatics, and being smaller ring of 7 FDDI stations. The topology of the network and the cable lengths were taken from the real system. *Fig. 2.* shows the topology of the two interconnected rings.

The two rings are interconnected by the *bmecisco7* router.

The load used in the simulation model came from measurements taken on the real rings. By using a protocol

analyzer, the first 32 octets of all the packets were copied from the ring and the packet lengths, arrival times, as well as the addresses of the source and destination stations were stored.

From the above data, one can produce a very detailed traffic matrix $T=[t_{ij}]$ where *i* and *j* stands for the source and destination stations, respectively and all the t_{ij} elements are two-dimensional distributions of the packets from $station_i$ to $station_j$, the dimensions being packet length and inter-arrival time.

THE SIMULATION MODEL

What is Modeled from the FDDI Network?

The parts of the FDDI standard that were found to be irrelevant concerning our simulation are not modeled. From the dual ring topology, only the primary ring is modeled because the secondary ring is not used during normal operation. All the FDDI stations contain a *Media Access Control* (MAC) entity. Its normal operation (Timed-Token Ring Protocol) is simulated precisely. All the ring initialization and ring recovery mechanisms are omitted. At the beginning of the simulation, the *operative value* of the *Target Token Rotation Time* (*T_Opr*) is set in all the stations and a token is inserted into the ring by a chosen station. No errors are modeled during the operation as no errors were observed during the measurements. The exact fiber lengths between stations were taken from measurement logbooks. The value of *station latency* was taken from (MIL 3, 1996). The value of T_Opr was queried from FDDI stations. The effect of the wiring concentrators is modeled by a constant delay.

The Simulation Environment

The simulation was performed using the OMNeT++ discrete-event simulator. It was developed by András Varga at the Technical University of Budapest, and is freely available for academic purposes. Readers interested in the simulator should refer to the OMNeT++ Home Page (Varga, 1997).

OMNeT++ makes it possible to run the segments of the simulated system on different computers. It uses PVM for the communication of the segments. All the segments are simulated separately using their own *Future Event Set*s (also called *event queue*), and their *Local Virtual Time*s (LVT) are NOT synchronized, unless the user* explicitly asks for it.

Load Modeling

As it was mentioned before, the $T=[t_{ij}]$ traffic matrix was derived from the measurements on the FDDI rings. This matrix was observed at given points of the rings where the protocol analyzer was inserted. T is not identical with the $D=[d_{ij}]$ demand matrix, where d_{ij} expresses the two-dimensional (length and inter-arrival time) distribution of send requests arriving from the outside world at $station_i$ (as source station) to be sent to $station_j$. We need to use D for load generation but only T can be easily measured. In (Lencse 1997) we have already shown why one can use T for load generation instead of D, though they are not identical matrices.

As the average utilization of the observed FDDI network is below 5%, a factor called *Load Multiplier* was introduced to simulate higher load. The inter-arrival time is divided by this factor so the load becomes Load Multiplier times more. By changing the value of this factor, the load of the system can be easily modified, while the nature of the traffic remains the same.

Traffic Between the Rings

The two rings are interconnected by a router, which has one port in both rings. In the simulation model called "*wired*", the two router-ports are interconnected by 2 wires. All the packets destined to the router in one ring will be captured by the router port and sent to the router port connected to the other ring. In the model, that port has to select a destination station randomly for the packet in the other ring, because the original routing information was not captured during the traffic measurement. The selection is done on statistical basis and the algorithm can be found in (Lencse, 1997).

In the simulation models called "*SSM-C*" and "*SSM-T*" the two router ports operate in the same way but they are interconnected through statistical interfaces.

Statistical Interfaces

The statistical interfaces of the SSM consist of output interface (OIF) and input interface (IIF) pairs. An OIF captures the messages, collects statistics about them, and if certain conditions are met it sends the statistics to the corresponding IIF. The IIF generates messages on the basis of the statistics it received from the corresponding OIF.

The condition that triggers the OIF to send the collected statistics should be chosen by the user. In the earlier paper (Lencse 1997), the counter driven approach was used. This is called *SSM-C*. In SSM-C the transmission of the statistics is controlled by the so called *update threshold* (UT) parameter. The OIF counts the captured messages and if the counter reaches UT, the OIF sends the collected statistics to the appropriate IIF and also restarts its own statistics collection. In this way, when the OIF sends its statistics they are already based on enough observations to achieve the required accuracy; but this method may produce long transient if the rate of message arrival to the OIF is low.

We have already mentioned that OMNeT++ supports the synchronization of the LVT's of the segments if the virtual time of the required synchronization is known in advance. Unfortunately, it is not known in advance, at what LVT an OIF will finish collecting UT number of samples. For this reason, the time-driven approach is used in this paper. This is called *SSM-T*. In SSM-T the OIF sends its statistics after an *update interval* (UI) LVT has been elapsed. Using this second method, the LVT of the statistics sending is predictable but the accuracy of the statistics is not ensured. For the latter, the OIF does not delete its statistics at the time of statistics sending automatically. The user must explicitly specify, when the statistics collection should be restarted.

* Throughout this paper, under "*user*" we mean the person who implements the simulation.

During the simulation, the length and the inter-arrival time of the messages are observed by the OIF's and the collected statistics (histograms) are sent to the appropriate IIF's. The IIF's re-generate the traffic on the basis of the statistics.

QUESTIONS OF PARALLELISATION

Partitioning of the Model

Partitioning is always an important question in parallel discrete-event simulation. The question is even more critical if we use SSM between the segments. SSM may be used only at points, where not the individual messages but only the statistical characteristics of the message flow are important. In our simulation model, there is only one point where this condition is satisfied: the connection between the two rings. The insertion of a statistical interface pair into an FDDI ring would result in the violation of the media access control protocol of the ring.

Diverse Local Virtual Times

SSM eliminates the need for event-by-event synchronization of the segments. In this way, if the system is simulated distributed (in multiple processes, either on the same processor, or on multiple ones) then each segment has its own event queue, and the LVT's of the segments may be different. (Pongor 1994) This approach uses SSM-C, the counter-driven version of SSM.

Loose Synchronization

It is many times expected, that the LVT's of the segments meet at certain points of time. For example, if a parameter is changed in segment A at t_0, its effect should reach segment B not earlier than t_0 and not later than $t_0+\Delta t$, where Δt is an allowed time interval for the transient. This requirement can be satisfied by the following construction:

Loose synchronization between segment A and segment B is defined formally as follows:

Let t_1, t_2, t_3, ... t_i, ... t_n be synchronization points of time. Let t_A and t_B denote the LVT's of segment A and segment B, respectively.
Segment A and segment B are loosely synchronized if:

$$((t_A < t_i) \Rightarrow (t_B \le t_i)) \wedge ((t_A > t_i) \Rightarrow (t_B \ge t_i)), \quad i = 1, 2, \ldots, n.$$

The loose synchronization of the two segments means that none of them may leave any synchronization point of time (t_i) until the other one reached it.

In t_i the segments A and B may exchange the statistics they have collected before t_i and they may use the new ones in the $[t_i, t_{i+1}]$ time interval. With the appropriate choice of the t_i synchronization points of time, it is ensured that the effect of a change in segment A in t_0 will reach segment B at the earliest at t_0 and at the latest at $t_0+\Delta t$. In this paper, we use loose synchronization where $t_i=i*UI$, UI being the update interval. This approach uses SSM-T, the time driven version of SSM. The great advantage of the loose synchronization and SSM-T over the event by event synchronization is that the segments may be simulated independently between the t_i synchronization points of time.

The simulation tool, OMNeT++ supports the loose synchronization in the following way. If *Segment A* wants to send a message to *Segment B* at LVT t_0 then Segment A should ask the simulator not to let the LVT of Segment B pass t_0 until it received a message from Segment A. In this way, the causality is ensured and no rollback is necessary. This is a so-called "uni-directional" synchronization. If both segments ask for synchronization for the same LVT, then the LVT's of the two segments will really meet.

SIMULATION RESULTS

Types of Transients Caused by the SSM

The aim of our first simulation experiments was to examine the transient behavior of SSM. First, to set up a reference, the two rings were interconnected with wires. Second, the wires were replaced with the statistical interfaces of SSM-C. Third, SSM-T was used between the two rings. With the help of the Load Multiplier factor, the offered load of the Northern Ring was raised by a factor of 2 at t_1=1 (LVT) and it was set back to its original value at t_2=2s. The statistics collection of the output interfaces of SSM-T was restarted at t_1 and t_2. We observed the utilisation in both rings and calculated the average in each 25 ms (LVT) window and plotted as a function of time.

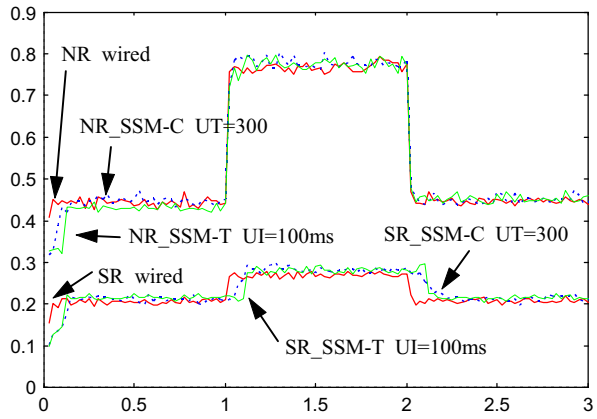

Fig. 3. Transient behaviour: the load changes

Fig. 3. shows the average of 20 simulation runs with different seeds for the random number generator. The averaging of the results of multiple runs was necessary because there were too many fluctuations -- the 25 ms window size could not be increased because the transients would have disappeared.

The curve titled "NR_wired" shows the utilization of the Northern ring in the case when the two rings are interconnected by wires and serves as a reference. "NR_SSM-C UT=300" and "NR_SSM-T UI=100ms" show the utilization of the Northern Ring in the case when using SSM-C and SSM-T, respectively. The meanings of the curve titles are similar for the Southern Ring (SR).

Two types of transients can be observed in the graph. The *initial transient* is caused by the statistical interfaces only: a certain proportion of the load of the rings comes from the other ring, and the OIF - IIF pair needs some time to collect statistics. This time (and in this way the length of the initial transient)

depends on the rate of packets arriving to the OIF in the case of SSM-C. This difference can be observed in the graph: the Southern Ring has longer initial transient then the Northern Ring. In the case of SSM-T the length of the initial transient depends on the UI parameter only and for this reason it is of equal length for both rings. When the load of the rings is changed, the statistical interfaces need some time to collect new statistics, in this way if there is a transient in the simulated system, SSM produces further transient. Let us see the graph again: at t_1 NR_SSM-C and NR_SSM-T rise with NR_wired, because the load from the Southern Ring to the Northern Ring does not change. After t_1 the IIF of the Southern Ring still generates packets according to the old statistics, so SR_SSM-C and SR_SSM-T do not raise until the IIF gets new statistics. The situation is similar after t_2 but the transient of the SSM-C is longer than after t_1 because the OIF sends the statistics after the arrivial of every UT-th packets and after t_2 the load in the Northern Ring is significantly less the after t_1. Again, the length of the transient caused by SSM-T is equal to UI.

A drawback of SSM-T is that the OIF can not automatically delete its statistics after sending them because it has to be ensured that the statistics collected by the OIF are based on enough observations. In the present implementation of the OIF of SSM-T it is the responsibility of the user to explicitly specify when the statistics collection of the OIF has to be restarted. As it can be seen from the graph the UI=100ms is large enough ("NR_SSM-C UT=300" rises earlier than "NR_SSM-T UI=100ms") to collect the necessary number of observations. The needed time interval for the collection of the required number of observations is not known in general bacause it depends on the arrivial rate of the messages. Of course, if the arrival rate is measured in advance, an appropriate UI can be choosen and then the OIF may delete the old statistics right after sending them.

Accuracy of the Results

As it was examined in (Lencse 1997), SSM-C produces acceptable result accuracy. SSM-T is even better, as its statistics are based on more and more observations as time elapses because the OIF does not delete the old statistics. If the user tunes the value of the UI and asks for the deletion of the statistics after every sending of the statistics to the appropriate IIF, then the result accuracy will depend on the choice of UI.

Fig. 4. shows the utilisation of the rings for a longer period. The value of the average utilisation in the (1s, 5s) interval can be seen in brackets next to the titles of the curves.

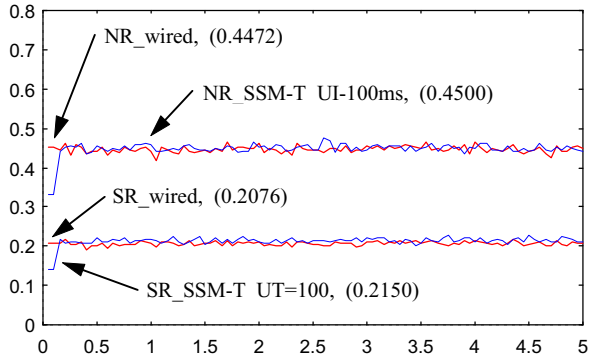

Fig. 4. Utilisation of the FDDI rings with and without SSM-T

Efficiency of SSM

The main aim of our research is to speed up the simulation with SSM. The following 4 simulation models were executed until 2s virtual time.

1. The two rings are interconnected by wires; the simulation of the whole network is done by one process; there is one event queue; the LVT's of the two rings are common.

2. The two rings are interconnected by ***statistical interfaces of type SSM-T***; the simulation of the whole network is done by one process; there is one event queue; the LVT's of the two rings are common.

3. The two rings are interconnected by statistical interfaces of type SSM-T; the ***simulation of the rings is done by two separate processes having their own event queue; the LVT's of the two rings are loosely synchronized***. The two processes are executed by the same processor, ***using PVM for communication with each other***.

4. The two rings are interconnected by statistical interfaces of type SSM-T; the simulation of the each ring is done by a separate process having their own event queue; the LVT's of the two rings are loosely synchronized. ***Each process is executed by its own processor***, using PVM for communication with each other.

Of course, the results produced by the execution of the 2, 3 and 4 models are completely the same, only the execution times differ.

All the experiments were executed on PCs with 100MHz Intel Pentium processors, 16MB RAM, 512KB cache. In model 4, the two computers were interconnected by a 150 Mbps ATM network. The processes ran under Linux operating system (kernel version 2.0.25) and were compiled by gcc v2.7 with the "-O3" flag on (full optimization). The PVM version was 3.3.

The execution times were measured by the UNIX "time" command; the so called wall clock times were taken into consideration. (Wall clock time means the physically elapsed time in the real world from the start to the termination of a process.)

Table 1. shows the results. As it can be seen, neither the insertion of the SSM-T interfaces (model 2) nor the loose

synchronization with the necessary inter-process communication (model 3) caused significant overhead compared to the reference simulation (model 1). The results of model 4 show 1.91 and 1.86 speedups compared to the results of model 1 for UI=100ms and UI=10ms, respectively. Model 4 was executed for UI=1ms too and its execution time was 5 minutes and 33 seconds resulting in a speedup of 1.75. This is also good but UI=1ms probably will not be used in practice being too small for statistics collection as the typical value of the TTRT (Target Token Rotation Time) of FDDI is in the order of 10ms.

Other Observations

During the simulation experiments two interesting observations were made:

The Future Event Set of the OMNeT++ simulator is stored in a (binary) heap. When linked list was used instead of heap, the following observation was made. First, the network was simulated in one process using SSM-T. Afterwards, it was simulated in two processes running on the same processor. In the second case the execution of the simulation required 23% less wall clock time than in the first case. It sounds surprising but the explanation is very simple. In the first case the average size of the FES was 79. In the distributed case the average sizes of the FES's were 55 and 24 for the Northern Ring and for the Southern Ring, respectively. The number of all the simulation events to schedule was nearly the same in the two cases but in the second case the linked lists (storing the FES) were shorter so it required fewer instructions to handle them. (Especially the number of the key comparisons dominates.) When heap was used for implementing the FES, no such kind of speedup was experienced. This observation is in accordance with the theoretical results in (Lencse 1995).

The other observation is the negative correlation between the load of the FDDI rings and the execution time of the simulation. When the load of a ring is small, the stations are repeating the token in the majority of the time, as they do not have packets to transmit. The token is short, so the average increment of the virtual time per event is very low. The simulation of an empty FDDI ring requires more events (and in this way more execution time) than a heavily loaded one. For this reason the FDDI models used in commercial simulators detect the idle state of a ring and use token extraction and retransmission mechanisms (MIL 3 1996).

CONCLUSION

The transient behavior and accuracy of the Statistical Synchronization Method were tested and compared to the traditional event-by-event synchronization. The selected simulated system was an FDDI network. It was found that the results produced by using SSM-T are close to the results of the traditional simulation used as reference.

A simple parallelisation method was proposed with the loose synchronization and SSM-T. Using this method, a speedup of 1.91 or 1.86 (depending on the UI parameter of SSM-T) was achieved in the simulation of two interconnected FDDI rings by two networked computers.

It was found that in some cases the parallel simulation produces speedup even if the processes simulating the parts of the system execute on a single processor.

We conclude that SSM-T and loose synchronization makes it possible to implement efficient parallel simulation of large systems on clusters of workstations or multiprocessor systems.

Model no.	UI = 100 ms	UI = 10 ms
1	9:42	
2	9:43	9:46
3	9:45	9:50
4	5:04	5:13

Table 1. Execution times of simulation models in [min:sec]

REFERENCES

ANSI X3.139. 1987. Fiber Distributed Data Interface (FDDI) Token Ring Media Access Control (MAC). American National Standards Institute, New York, NY

Fujimoto, R. M. 1990. "Parallel Discrete Event Simulation." Communications of the ACM 33, no 10, 31-53

Jefferson, D; B. Beckman; F. Wieland; L. Blume; M. DiLoreto; P. Hontalas; P. Laroche; K. Sturdevant; J. Tupman; V. Warren; J. Vedel; H. Younger and S. Bellenot. 1987. "Distributed Simulation and the Time Warp Operating System." Proceedings of the 12th SIGOPS - Symposium on Operating System Principles, pp. 73-93.

Lencse, G. 1995. "Investigation of Event Set Algorithms" Proceedings of the 1995 European Simulation Multiconference (ESM 95) (Prague, Czech Republic, June 5-7). SCS Europe, pp. 821-825.

Lencse, G. 1997. "Efficient simulation of large systems - Transient Behaviour and Accuracy" Proceedings of the 1997 European Simulation Symposium (ESS 97) (Passau, Germany, Okt. 19-23). SCS Europe, pp. 660-665.

MIL 3. 1996. OPNET Example Models Manual, Release 3. (Chapter FDDI) MIL 3, Inc.

Pongor, Gy. 1992. "Statistical Synchronization: a Different Approach of Parallel Discrete Event Simulation". Proceedings of the 1992 European Simulation Symposium (ESS 92) (The Blockhaus, Dresden, Germany, Nov. 5-8). SCS Europe, pp. 125-129.

Pongor, Gy. 1994. "Multiple Virtual Times in Parallel Discrete Event Simulation". Proceedings of the Parallel Processing Wokshop (Technical Univ. of Budapest, Budapest, Hungary, Febr. 10-11). Request paper from its author by E-mail: Pongor@hit.bme.hu

Varga, A. 1997. The OMNeT++ Home Page http://www.hit.bme.hu/phd/vargaa/omnetpp.htm

BIOGRAPHY

Gábor Lencse was born in Gyôr, Hungary, in 1970. He received his M.S. in electrical engineering and computer systems from the Technical University of Budapest in 1994. He is currently pursuing his Ph. D. at the same university. The area of his research is computer architectures and parallel processing. He is interested in (parallel) discrete event simulation.