Proceedings of the 1996 Winter Simulation Conference
ed. J. M. Charnes, D. J. Morrice, D. T. Brunner, and J. J. Swain

PERFORMANCE COMPARISON OF HIGH LEVEL ALGEBRAIC NETS DISTRIBUTED SIMULATION PROTOCOLS

Karim Djemame
Mohamed Bettaz

Computing Science Institute
University of Constantine
Constantine 25000, ALGERIA

Dennis C.Gilles
Lewis M.Mackenzie

Computing Science Department
University of Glasgow
Glasgow G12 8QQ, Scotland, UK

ABSTRACT

This paper addresses the problem of developing distributed simulation techniques to analyze ECATNets. ECATNets (Extended Concurrent Algebraic Term Nets) are a kind of High-Level Algebraic Nets used for specifying various aspects of distributed and parallel systems. Their most distinctive feature is that their semantics is defined in terms of rewriting logic. The conservative and optimistic approaches of Distributed Discrete Event Simulation (DDES) are used as the starting point to discuss a simulation framework for studying the behaviour of ECATNet models. The ECATNet model to be simulated is partitioned into several connected subnets. The various subnets are simulated in parallel by several Logical Processes. Next we develop two distributed simulation protocols to execute discrete-event simulations of ECATNets.

1 INTRODUCTION

Petri nets (Murata 1989) are an important graphical and mathematical tool applicable to many systems. They are a promising tool for describing and studying information processing systems that are characterized as being concurrent, asynchronous, distributed, parallel, nondeterministic and/or stochastic. ECATNets are a kind of High-Level Algebraic Nets. They are proposed as a way for specification, modeling and validation of applications from the area of communication networks, computer designs and other complex systems (Bettaz et al. 1993a, Bettaz et al. 1993b). They are built around a combination of three formalisms. The first two formalisms constitute a net/data model, and are used for defining the syntax of the system, in other terms to capture its structure. The net model, which is a kind of advanced Petri net (Jensen and Rozenberg 1991), is used to describe the process architecture of the system; the data model, which is an algebraic formalism (Ehrig and Mahr 1985), is used for specifying the data structures of the system. The third formalism, which is a rewriting logic (Meseguer 1992), is used for defining the semantics of the system, or in other words to describe its behaviour. According to this logic, the system behaviour may be explained by formal reasoning. Transforming our logic into a rewriting system (Bettaz and Mehemmel 1993) may be used for rapid prototyping and automatic proving of a system under design. However, the achieved models had two drawbacks: the occultation of the problem of time and a bad exploitation of the parallelism inherent to the studied models. In a previous paper (Djemame and Bettaz 1995), we presented an approach for the treatment of time in ECATNets, by introducing in each transition a firing delay to perform the operations "remove/deposit tokens". Simulation of these models can not only perform their validation, but can evaluate their performances as well. Simulation can also be amenable to parallel execution in order to exploit the inherent parallelism of these models. It is worth mentioning that we are dealing with the parallelism at two levels: the inter-module level, where the parallelism is achieved by partitioning the "initial" models w.r.t. a "separation of concern" strategy; the intra-module level, where the detection of the parallelism is permitted by the use of our rewriting logic.

In Distributed Discrete Event Simulation (DDES), the system being modeled is viewed as being composed of some number of Physical Processes (PPs) that interact at various points in simulated time. The simulator is constructed as a set of Logical Processes (or simply LPs), one per Physical Process. Logical Processes exchange timestamped event messages to interact. However, relationship between events may exist, so concurrent execution of these events must be synchronized, otherwise *causality errors* can occur. Distributed Discrete Event Simulation falls into two categories (Fujimoto 1990): *conservative* and *optimistic*. In the following, we consider synchronization protocols based on the Chandy-Misra scheme (CM) described by (Misra 1986) (conservative), and on Time Warp described by (Jefferson 1985) (optimistic, based on the *virtual time* paradigm). The conservative mechanisms strictly avoid the possibility of any causality error ever occurring by forcing LPs to block as long as there is the possibility for receiving messages with lower timestamp. The optimistic mechanisms use a detection and recovery approach, this means that causality errors are detected and a rollback mechanism is invoked to recover. Recently, (Chiola and Ferscha 1995) proposed a new DDES protocol called *probabilistic*, a performance efficient compromise between the two classical approaches.

The contributions of distributed simulation in the area of Petri nets and reported in the literature include (Ammar and Deng 1991, Nicol and Roy 1991, Thomas and Zahorjan 1991, Chiola and Ferscha 1993). They all deal with Timed and/or Stochastic nets. In (Chiola and Ferscha 1993), Petri net structural analysis is exploited for the efficient implementation of DDES techniques using both approaches: *conservative* and *optimistic*. To the best of our knowledge, no attention has been given to high-level Petri nets distributed simulation. We are only aware of the work in progress by Schöf on parallel simulation of THOR nets (Timed Hierarchical Object-Related Nets) (Schöf et al. 1995). It is worth mentioning that concerning simulation techniques for high-level nets with arc inscriptions, the enabling test and the firing operations are substantially more complex.

This paper is organized in the following way: in section 2, we review some basic notions about ECATNets, and illustrate their use by a complete example. Section 3 describes synchronization protocols based on the conservative and optimistic approaches for ECATNets distributed simulation. Section 4 presents some empirical results derived from the implementation of DDES strategies for ECATNet models. Some concluding remarks and perspectives of the research for future developments are given in section 5.

2 ALGEBRAIC TERM NETS

2.1 Basic Concepts

The graphical representation of a generic ECATNet is given by Figure 1. **IC**, **DT** and **CT** are multisets of (equivalence classes of) terms. The terms are defined by an algebraic specification of an abstract data type given by the user. **TC** (Transition Condition) is a boolean expression which may contain variables occurring in IC (Input Condition), DT (Destroyed Tokens) and CT (Created Tokens). Each place is associated with a capacity **C(p)** defined as a multiset of closed (equivalence classes of) terms. The marking **M(p)** of a place p of the net, which is itself a multiset of closed

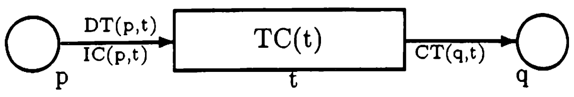

Figure 1: A Generic ECATNet

terms, is defined w.r.t. the capacity (which may be infinite).

A transition t is fireable when various conditions are simultaneously true. The first condition is that every IC(p,t) for each input place p is enabled. The second condition is that TC(t) is true. Finally the addition of CT(q,t) to each output place q must not result in q exceeding its capacity when this capacity is finite. When t is fired, DT(p,t) is removed from the input place p and simultaneously CT(q,t) is added to the output place q. Transition firing and its conditions are expressed by rewrite rules which are strongly depending on the form of the syntactic notation used for representing IC. Those rewrite rules (metarules) together with a set of deduction rules define a rewriting logic which gives the semantics of the net. They act as a parallelizing compiler which tries to find sequences of "code" which may be executed in parallel. Examples on concrete instantiations and practical use of our metarules may be found in (Bettaz et al. 1992, Bettaz 1993, Bettaz and Maouche 1993, Bettaz et al. 1994).

2.2 Aspect of Time

In (Djemame and Bettaz 1995) we have chosen to introduce in each transition a (marking) related rate. This will lead to take into account a firing delay "d" to perform the operations "remove/deposit tokens". This choice allows to preserve the incremental approach used for defining ECATNets and preserves the semantic framework defined in terms of rewriting logic. Thus, *firing times* as defined for Timed Petri nets are offered leading to modeling activities duration. Note that there will be two kinds of transitions. *Timed* transitions are specified by annotating each timed transition t_i with a firing rate λ_i. Let $\lambda : T \rightarrow \mathcal{R}$ assigns firing delays λ_i to T-elements $t_i \in T$. Zero delays are associated with transitions that are called *immediate*. ECATNets enriched with temporal specification are then suitable to discrete simulation. This is an important step in their quantitative performance evaluation.

2.3 An Example of Modeling with ECATNets

This example, borrowed from (Bettaz 1993), deals with the behaviour of the Ethernet transmitting station. It comprises four modules, each module is specified by an ECATNet model. The first module deals with the functions of formatting and transmitting starting. The second module is relative to the functions of transmission with success and acknowledgement. The third module is relative to the retransmission function. The fourth module treats essentially the functions of collision handling and acknowledgement (Figures 2-5).

The transmitter station transmits one frame at a time. The user is not allowed to request the transmission of a new frame before receiving the ackowledgement of the previous one. The formatting function starts when a token of type "d,s,data" is deposited in place FROM_USER. This token is considered as a primitive transferred from the user layer to the MAC layer for requesting the transmission of data "data", from a source "s" to a destination "d". The frame "d.s.data.fcs" is then deposited as soon as it is composed in a transmission register (TRANS_REG). The formatting function is consisting of the concatenation of sequences of bits corresponding to the addresses "d" and "s", to the data "data", and to the error control sequence "fcs" previously computed. On the other hand, the MAC layer is listening to the medium (CARRIER_SENSE) in order to avoid any collision occurring with a current transmission. The place CARRIER_SENSE is an interface between the MAC layer and the physical layer. When the medium becomes free (a token "false" is present in place CARRIER_SENSE), it waits a certain amount of time corresponding to the inter-frame spacing delay. Then, considering that the transmission may terminate with success (deposit of a token "false" in place BUSY_CHANNEL and a token "true" in place SUC_TRANS), it takes possession of the medium (CHANNEL_ACCESS) and the transmission starts (deposit of a token "true" in INIT_TRANS).

3 A DISTRIBUTED SIMULATION FRAMEWORK FOR ECATNETS

For each LP, we identify three components: the work partition assigned to it, its communication behaviour and its simulation engine are the constituent parts of the distributed simulation framework.

3.1 Net Partitioning Performance Impact

Starting from the "separation of concern" strategy and according to the conservative and the optimistic mechanisms, details about the protocols developed using such partitioning are found in (Djemame and Bettaz

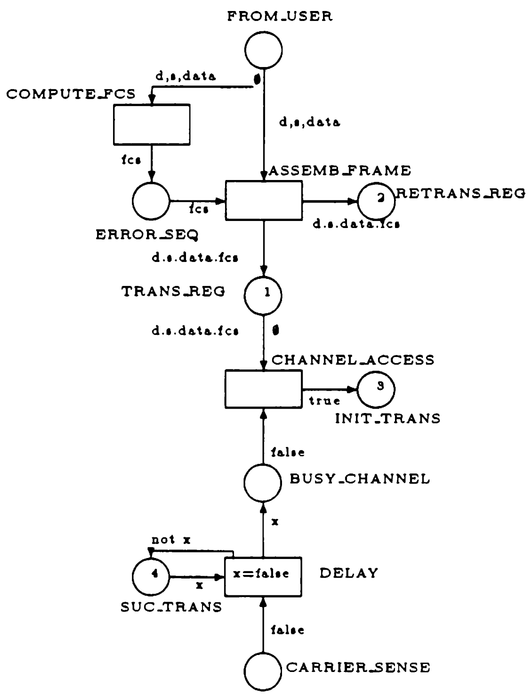

Figure 2: Starting of Transmission Module

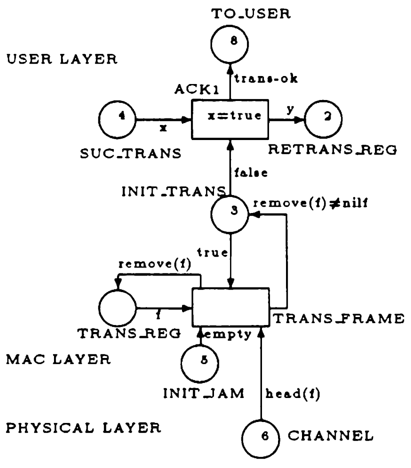

Figure 3: Transmission With Success Module

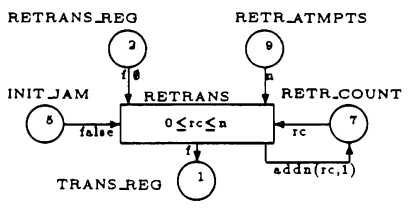

Figure 4: Retransmission Module

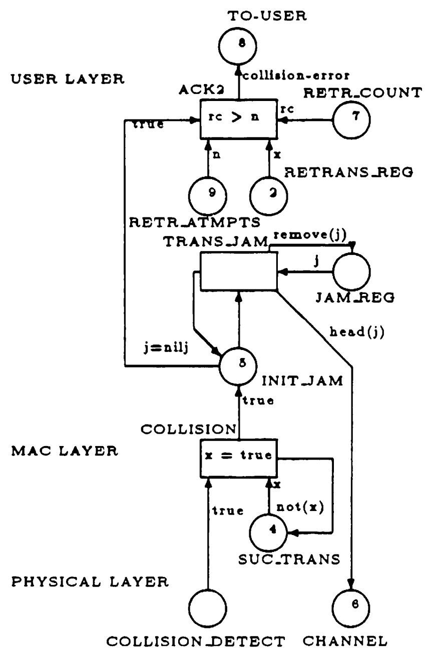

Figure 5: Collision Handling Module

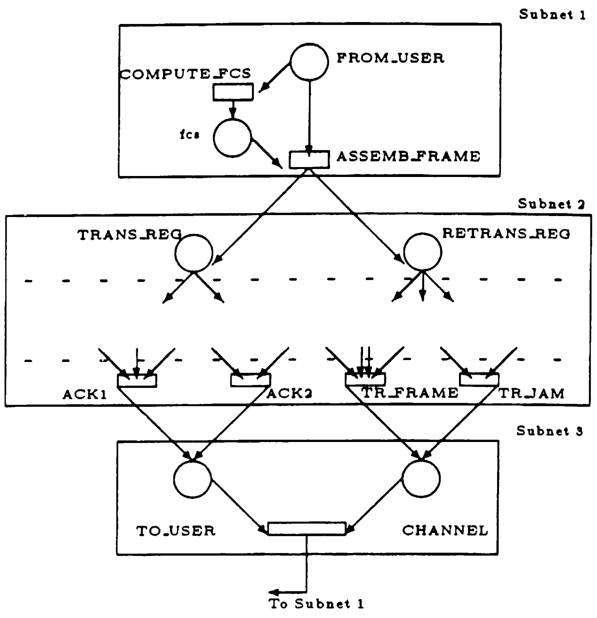

Figure 6: ECATNet′Partition

1995). However, it has been seen that for performance reasons the generality of a partition should be limited in such a way that conflicting transitions together with all their input places should always reside in the same LP (Nicol and Roy 1991, Chiola and Ferscha 1993). Such partitioning avoids the implementation of a distributed conflict resolution algorithm and minimises overhead.

Separation of Subnets The partitioning has to be related to the firing rule : an LP should be a set of transitions along with their input places such that local information is sufficient to decide upon the enabling and firing of any transition. If P, T and A are respectively the set of places, the set of transitions and the set of arcs of the "intial" ECATNet model, the partition is a set of n subnets such that:
$ECATNet_i = (P_i, T_i, A_i, \lambda_i)$ where $\cup\, P_i$ = P, $\cup\, T_i$ = T, $A_i \subset (P_i \times T_i) \cup (T_i \times P_i)$, i=1..n.

λ_i is the set of rates of local transitions in T_i. We did not consider the multisets of terms IC, DT, CT and the boolean expression TC. They are defined as in the "initial" model and appear in the graphical representation of the subnets.
Definitions A place $p_i \in P_i$ in LP_i is said to be a member of the set of *input places* (IP_i) of LP_i if there exists a transition $t_j \notin LP_i$ which p_i is an output place. A transition $t_i \in T_i$ in LP_i is said to be a member of the set of *output transitions* (OT_i) of LP_i if there exists a place $p_j \notin LP_i$ which t_i is an input transition. A *communication arc* is an arc connecting a place (transition) in LP_i to a transition (place) in LP_j.
For each *output transition* t in subnet i, define :
$P_{t_{out}}$ = list of places to indicate which (input) places are related to t and the kind of relation that exists (CT).
Example Let's take the example of section 2.3, the modular specification of the Ethernet transmitting station. The partition is a set of three subnets, each subnet is simulated by an LP. Places and transitions are partitioned among the subnets as follows (Figure 6):
Subnet 1: 2 places, 2 transitions;
Subnet 2: 11 places, 8 transitions;
Subnet 3: 2 places, 1 transition;

TRANS_REQ and TO_USER are input places in $subnet_2$ and $subnet_3$ respectively. In $subnet_1$, ASSEMB_FRAME is an output transition. (ASSEMB_FRAME, TRANS_REQ), (ASSEMB_FRAME, RETRANS_REQ), (TRANS_FRAME, CHANNEL) are communication arcs.

3.2 The Communication Interface

The decomposition of ECATNets requires an interface among ECATNet partitions preserving the behavioural semantics of the whole ECATNet. Such an interface has to be implemented by an appropriate protocol among the partitions according to the simulation strategy. If there is a communication arc from an *output transition* t to an *input place* p, there exists a unidirectional channel between them. This is motivated by (output) transitions which have to interact with their (input) places for sending tokens when these transitions fire. We can map the set of arcs ($T_k \times P_l$) interconnecting different subnets to the channels of the communication interface. We define I_k = (*CHANNELS*,m) of *subnet*$_k$ to be the communication interface with $CHANNELS = \bigcup_{i,j} ch_{i,j}$ where $ch_{i,j}$ = (LP_i,LP_j) is a set of directed channel from LP_i to LP_j corresponding to the arcs $(t_k,p_l) \in (T_k \times P_l)$ carrying messages of type m.

3.3 Chandy-Misra's Strategy

The simulation engine implements the simulation strategy. The simulation of events is performed in *virtual time* according to their causality. The data structures according to the conservative approach, a local virtual time (LVT) (representing an accumulated value of firing time in LP_i), a list of events (EVL) ordered by time of occurrence, input queues (IQ) (one queue per each input channel, which collects recently arrived messages ordered by time), and output queues (OQ) (one queue per output channel, which keeps messages to send, ordered by time) have to be maintained.

3.3.1 Lookahead Computation

Loohahead is required in conservative mechanisms to avoid deadlock situation (Misra 1986). It is the process' ability to predict what will happen, or more importantly, what will not happen in the simulated time as regards to its behaviour and when next it may affect other processes. It prevents incorrect computations from propagating too far ahead into the simulated time. Lookahead provides a "window" such that all events with timestamps in this window can be executed safely and without further communication between LPs. To highlight lookahead that exists in Petri nets simulation, if a transition starts firing at *Tsim*, an LP can predict exactly when the tokens generated by this firing are deposited (*Tsim+d*, where *d* is the firing delay associated with t). Lookahead comes directly from the ECATNet structure, and is the increment firing time of succeeding timed transitions in the LP plus the firing time of the transition of the output border of the LP. The value of lookahead can be established for a pair of transitions in each subnet by a static analysis of the subnet's structure.

3.3.2 Types of Messages

The form of the messages exchanged between LPs is (Type_message, source, destination, timestamp, type_token). *Source* and *destination* are either a place (in LP_i) or a transition (in LP_j). Type_token is the abstract data type of the token moved among subnets. The causality of events is preserved over all LPs by sending timestamped token messages of type **Tokens_Deposited**(t,p,TT,CT) in non-decreasing order. This message is carrying Created Tokens when t in LP_i fires leading to a deposit of tokens in place p in LP_j. **Tokens_Deposited** (t,p,TT, Null) is a Null message which is sent for synchronization purpose. It is a promise not to send a new message timestamped earlier than TT.

3.3.3 LP's Behaviour

The conservative approach allows only the processing of safe events, firing of transitions up to LVT for which the LP has been guaranteed not to receive messages with smaller timestamps. The behaviour of the conservative simulator is to process the first event of EVL if there is no token message in one of the IQ_is with smaller timestamp, or to process the token message with the minimum token time in IQs. Each input queue IQ_i has a clock CC_i associated with it that is equal to either the timestamp of the message at the head of the queue if the queue contains a message, or the timestamp of the last received message if the queue is empty. The LP blocks as soon as the minimum timestamp of messages in IQs is not larger than the occurrence time of the first event in EVL (if IQ_i becomes empty, the value of CC_i is changed to 0). The firing of a transition t is as follows: if t $\in$ (OT_i) in LP_i, then a bf Tokens_Deposited message is generated and inserted in the corresponding output queue (OQ). If t has an output place in LP_i, it schedules an event **End_Firing** of t. A Null message is also deposited for every output border transition in the corresponding OQ.

3.4 Time Warp Strategy

In order to simulate an ECATNet partition, the data structures of an LP we maintain with an optimitic simulator according to Time Warp are an input queue (IQ) which collects recently arrived messages (positive and negative), an output queue (OQP) which contains the *positive* messages to send, an output queue (OQN) which contains the *negative* copies of the messages recently sent (*antimessages* for unsending the originals), and an event stack (ES) which records all state variables such that a past state can be reconstructed in case of a rollback. The form of the entries is (t_i,LVT,M) where t_i is the transition that has fired at time LVT yielding a new marking M.

3.4.1 Types of Messages

Tokens_Deposited(t,p,TT,CT(t,p)) is a message carrying Created Tokens when t $\in OT_i$ fires leading to a deposit of tokens in place p in LP_j. The timestamp of this message is the accumulated firing time of transition t. **Tokens_Cancelled** is used in the rollback mechanism needed for synchonization. In this message, TT indicates which message should be cancelled.

The simulation engine's main task is not only to synchronize the LPs simulating the various subnets by controlling the timestamp (TT) of each message and the local virtual time (LVT), but also to implement the functions of the communication arcs. An LP processes messages in the input queue (IQ) by checking the sign (positive or negative) and the timestamp of each one (these messages are ordered by TT, the head of the queue corresponds to the smallest TT). Messages with timestamp > LVT are inserted in IQ. In case of a positive message, it is inserted in timestamp order, otherwise (the sign is negative) it annihilates the positive message in IQ previously sent. If the message is a *straggler* (timestamp of the message < LVT), the LP must roll back and restore a valid state. As for the conservative simulator, the processing of the first event in EVL or the first message in IQ generates either new events in EVL or output messages.

3.4.2 Message Cancellation

When an LP rolls back, it first inserts the straggler message into IQ and updates LVT. The state at (new) time LVT is restored. If rollback is applied with *aggressive cancellation*, all messages in OQN with token time > LVT are annihilated by removing them from OQN and sending them. All incorrect computation is undone by poping out all the records prematurely pushed in ES. The simulator can also apply *lazy cancellation*. In the case reevaluation yields exactly the same positive messages as already sent before, the new positive message is not resent. This will prevent unnecessary message transfers as well as possibly new rollbacks in other LPs.

3.4.3 Global Virtual Time

The *Global Virtual Time* (GVT) is considered as the virtual clock for the system as a whole. The knowledge of (GVT) reduces past state savings in ES. Any message in an input or output queue whose virtual time is < GVT can be discarded.

4 SIMULATION RESULTS

In the absence of a parallel computer, our parallel programs have been implemented in a network of (Sun Sparc) workstations. All our code is written in C plus MPI (Message Passing Interface) (MPI 1995). We have decided to run our initial tests for distributed simulation on discrete-event ECATNet model of the Ethernet transmitting station. The simulation has been running for 10,000 simulated time units. We have considered the simulator's implementations of the ECATNet partitioning in a parametrization: $\lambda_{DELAY} = 1.0$, probability of occurrence of a collision = 0.5, and N = 1, 2, 3, 4 processors respectively. Process 0 is a special process whose task is to start the simulation by assigning LP_i, i=1..3 to a dedicated processor and to compute GVT.

We faced two primary problems with the conservative approach. The first problem is related to cyclic models. A Null message sent out by one Logical Process could possibly circulate through a series of other Logical Processes and arrive back at the original sender at the time it was sent (eg. a Null message generated after transition DELAY in LP_2 fires). In some cases, exponential firing times are used, and because these have a minimum delay of zero, models must be modified for use on distributed simulation. To cope with this situation, a firing transition identifier was introduced in each Null message generated by a timed transition. If the Null message arrives back at the original sender at the time it was sent, it is simply discarded. Deadlock is avoided because there are no cycles in which the collective timestamp increment of messages traversing these cycles is 0. The second problem is related to the number of Null messages exchanged between LPs. Since a large number of transitions in LP_2 are immediate, there is no need to generate new Null messages when these transitions fire if there is no timed transition among the succeeding transitions up to the output border, i.e the accumulated firing time is 0, and this does not change lookahead.

With Time Warp, we faced the situation where a straggler positive message changes the marking in LP_2 but does not cause the enabling of any new event in the past. In such case, LP_2 does not have to rollback. The simplified mechanism which has been used to recover was an appropriate insertion of firings made on ES, and the top of ES was copied considering a potential change in the marking.

The results show that communication time between LPs is quite important. LP_1 and LP_3 contain one single input transition in the output border, whereas LP_2 contains four. LP_2 is the process with the largest event processing time because of its large number of transitions (8) and places (11) leading to an important number of events to schedule. The performances of the conservative simulator (Chandy-Misra's approach with deadlock avoidance) and the optimistic simulator

Table 1: LPs Execution Profiles (CM Approach, Number of Processors = 4)

Logical Process	Simulation Steps	Positve Messages	Null Messages	Time Evt. Processing	Time Communication	Time Blocking	Time Term. Protocol
1	20001	20002	10001	11.442	347.310	290.862	0.010
2	75000	20000	10001	66.773	282.757	213.071	0.027
3	20000	10000	10001	03.496	356.383	326.377	0.010

Table 2: LP_2 Execution Profile (CM Approach)

Number Proc.	Simulation Steps	Positve Messages	Null Messages	Time Evt. Processing	Time Communication	Time Blocking	Time Termin. Protocol
1	75000	20000	10001	79.703	819.843	162.407	1.067
2	75000	20000	10001	49.680	378.169	91.005	0.089
3	75000	20000	10001	83.855	281.944	200.557	0.410
4	75000	20000	10001	66.773	282.757	213.071	0.027

Table 3: LP_2 Execution Profile (TW, Lazy Cancellation)

Number Proc.	Simul. Steps	Number Rollbacks	Nb.Messages Pos./Neg.	Time Evt. Process.	Time Communic.	Time Rollback	Time Blocking	Time Term. Protocol
1	100001	5001	29997	123.722	861.796	2.385	0.461	1.008
2	85074	9950	20004	68.803	318.494	2.389	110.952	0.049
3	85125	9910	20002	73.184	281.583	2.357	219.943	0.054
4	85048	9970	20004	47.406	255.075	1.577	210.614	0.014

Table 4: Processing Time, CM vs TW

Nb.Proc.	Proc.Time (CM)	Proc.Time (TW)
1	15 mn 28 s 680	17 mn 49 s 944
2	07 mn 35 s 695	07 mn 15 s 698
3	06 mn 42 s 482	06 mn 46 s 962
4	06 mn 19 s 871	05 mn 40 s 175

(Time Warp, lazy cancellation) are shown in Tables 1-4. We have observed speedup of 2.4 and 3.1 using 3 and 4 processors respectively for the Ethernet transmitting station ECATNet model using our simulator. Speedup is reported by comparison with the distributed simulation code running on a single processor (Figure 7).

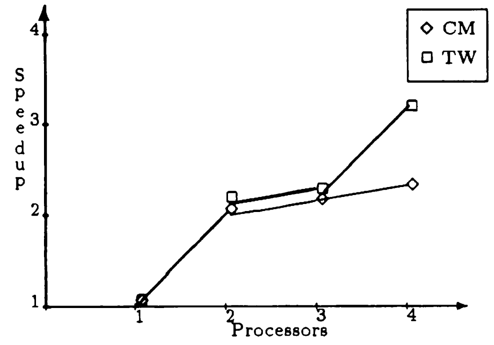

Figure 7: Speedup (CM vs TW)

5 CONCLUSION

Compared with other attempts on parallel or distributed simulation of Petri nets, the methods presented in this paper differ in at least one of the following points: ECATNets are high-level algebraic nets, the state of the net is distributed, and the simulation techniques have to respect timed transitions. The protocols we proposed are expected to improve the ability of efficiently simulate the behaviour of systems modelized by ECATNets over a period of time. An analysis of the structure of the ECATNet model had to be taken into account so that transitions sharing input places are always assigned to the same LP.

The design of a high-level algebraic net concurrent simulator is under investigation. Distributed Simulation is based on spatial decomposition whereas Concurrent Simulation (Jones et al. 1989) is based on temporal decomposition, which actually could be an alternative approach to the application of multiple processors to discrete event simulation. Our objective is a comparable study in order to clearly assess the performance of both approaches (Concurrent versus Distributed).

ACKNOWLEDGMENTS

K.Djemame is supported by a grant from the Algerian Ministry of Higher Education and Scientific Research, and the British Council under the Anglo-Algerian SATELLITE Programme.

REFERENCES

Ammar H. and Deng S. 1991. Parallel Simulation of Petri Nets using Spatial Decomposition. In *Proceedings of the IEEE International Symposium on Circuits and Systems*, pages 826-829, Singapore, jun.1991.

Bettaz M., Maouche M., Soualmi M. and Boukebeche M. 1992. Using ECATNets for Specifying Communication Software in the OSI Framework. In *Proceedings of ICCI'92*, pages 410-413. IEEE.

Bettaz M. 1993. Specification Hautement Compacte et Modulaire de l'ETHERNET: la Station Emettrice. In *Proceedings of CFIP'93*, Montreal, HERMES, Paris. (in french).

Bettaz M. and Maouche M. 1993. How to Specify Non Determinism and True Concurrency with Algebraic Term Nets. *Lecture Notes in Computer Science*, 655:164-180. Springer-Verlag.

Bettaz M. and Mehemmel A. 1993. Modeling and Proving of Truly Concurrent Systems with CATNets. In *Proceedings of Euromicro Workshop on Parallel and Distributed Processing*, pages 265-272.

Bettaz M., Maouche M., Soualmi M. and Boukebeche M. 1993a. Compact Modeling and Rapid Prototyping of Communication Software with ECATNets: a Case Study. *Simulation Series*, 25(1), 149-154. SCS and IEEE.

Bettaz M., Maouche M., Soualmi M. and Boukebeche M. 1993b. Protocol Specification using ECATNets. *Networking and Distributed Computing*, 3(1):7-35. Hermes, Paris.

Bettaz M., Maouche M., Soualmi M. and Boukebeche M. 1994. On Reusing ATNet Modules in Protocol Specification. *JSS*, 27(2):119-128, nov 1994.

Chiola G. and Ferscha A. 1993. Distributed Simulation of Timed Petri nets: Exploiting the Net Structure to Obtain Efficiency. *Lecture Notes in Computer Science*, 691:146-165. Springer-Verlag.

Chiola G. and Ferscha A. 1995. Performance Comparable Design of Efficient Synchronization Protocols for Distributed Simulation. In *Proceedings of MASCOTS'95*, pages 59-65, Durham, North Carolina, jan 1995. IEEE.

Djemame K. and Bettaz M. 1995. On the Parallel Simulation of ECATNets. Technical Report, Computing Science Institute, University of Constantine, Algeria, jun 1995.

Ehrig H. and Mahr B. 1985. Fundamentals of Algebraic Specifications. *EATCS Monographs on Theoretical Computer Science*, Springer-Verlag.

Fujimoto R. 1990. Parallel Discrete Event Simulation. *Communications of the ACM*, 33(10):31-53, oct 1990.

Jefferson D.R. 1985. Virtual Time. *ACM Transactions on Programming Languages and Systems*, 7(3):404-425, jul 1985.

Jensen K. and Rozenberg G.(Ed.). 1991. *High-Level Petri Nets*. Springer-Verlag, Berlin.

Jones D.W., Chou C.C., Renk D. and Bruell S.C. 1989. Experience with Concurrent Simulation. In *Proceedings of the 1989 Winter Simulation Conference*, pages 756-764, MacNair E.A., Musselman K.J., P.Heidelberger (Ed.).

Meseguer J. 1992. Conditional Rewriting Logic as a Unified Model of Concurrency. *Theoretical Computer Science*, 96, pages 73-155.

Misra J. 1986. Distributed Discrete Event Simulation. *ACM Computing Surveys*, 18(1):39-65, mar 1986.

Message Passing Interface Forum. 1995. MPI: A Message-Passing Interface Standard. Technical Report CS-93-214, University of Tennessee, jun 1995.

Murata T. 1989. Petri Nets: Properties, Analysis and Applications. *Proceedings of the IEEE*, 77(4):541-580, apr 1989.

Nicol D. and Roy S. 1991. Parallel Simulation of Timed Petri Nets. In *Proceedings of the 1991 Winter Simulation Conference*, pages 574-583, B.Nelson, D.Kelton, G.Clark (Ed.).

Schöf S., Sonnenschein M. and Wieting R. 1995. Efficient Simulation of THOR Nets. *Lecture Notes in Computer Science*, 935:412-431. Springer-Verlag.

Thomas G.S. and Zahorjan J. 1991. Parallel Simulation of Performance Petri Nets: Extending the Domain of Parallel Simulation. In *Proceedings of the 1991 Winter Simulation Conference*, pages 564–573, B.Nelson, D.Kelton, G.Clark (Ed.).

AUTHOR BIOGRAPHIES

KARIM DJEMAME completed an M.S. in computer science at the University of Constantine, Algeria, in 1991. He is currently a Ph.D. student on leave at the University of Glasgow.

MOHAMED BETTAZ is a professor in the institute of computing science at the university of Constantine, Algeria. His research interests include introducing formal specifications in the life cycle of software engineering. He is a member of IFIP.

DENNIS C.GILLES is an emeritus professor of computing science at the University of Glasgow.

LEWIS M.MACKENZIE is a lecturer in the department of computing science at the University of Glasgow. His research interests emphasize high performance networks and scalable parallel architectures.