Evaluations of the New Synchronization Method for the Parallel Simulations of Wireless Networks

Sławomir Nowak, Agnieszka Debudaj-Grabysz, and Mateusz Nowak

***Abstract*—The article describes the implementation of the new synchronization method for parallel discrete event simulation and presents the experimental results obtained from parallel environment. In the proposed method the time steps are introduced and the synchronization occurs only at the end of a time step to achieve reduction of messages, exchanged between local processes of simulation. The method was implemented in C++, using the MPI library. The studies have shown that it is possible, under certain conditions and using appropriate hardware architecture, the significant acceleration of distributed simulations.**

***Index Terms*—Distributed simulations, wireless network simulation, synchronization.**

I. Introduction

With increasing complexity of simulation models and scenarios, the demands on computational resources also significantly increase. This problem is particularly important for the simulation of wireless networks, due to the complexity of the physical layer, shared medium and especially interferences.

The efficient and scalable simulations assume simplified models especially of the PHY layer and/or using the parallel simulations [1]. Parallel *discrete event simulations* (DES) use several processors, cores or hosts to achieve a considerable speedup. The simulation's scenario is divided in a number of logical processes of simulation (LPs), each of them having its own clock (LVT, *local virtual time*) and executing a part of the scenario. The LPs must be synchronized to ensure consecutive processing of events in accordance with time stamps and avoid the *causuality errors* (where an incoming event's timestamp is less than LVT of the given LP) [2].

There are two main concepts to handle event synchronization: *conservative algorithms* and *optimistic* ones. In the conservative algorithms, by avoiding violating the local causality constraint, a causality errors never happen. The basic and most popular conservative synchronization method is the Chandy/Misra/Bryant [3] algorithm, where LPs exchange control messages called *null messages*, containing lower timestamp bound of given LP's future messages.

In optimistic approach the causality errors may occur, but they are recovered using a rollback mechanism, which restores the correct state of LP. As withdrawal action taken by given LP, also events sent in roll backed period must be cancelled. This leads to rollback on another LPs. The drawback is the necessity of storing last objects states. Rollback operation is usually time-costly, and links with high memory overhead. The representative optimistic synchronization algorithm is the Time Warp [4].

In [5] a new method of synchronization was proposed to improve performance of parallel simulation of wireless communication, with potential maintaining the accuracy of detailed model of the PHY/L2 layers. The method in general represents an optimistic synchronization with time windows (e.g. [6], [7]), and was adapted (by introducing simulation *time steps*) to specific of wireless communication with contention based media access, as in 802.11 based wireless networks

The parallel simulator presented in the paper combines benefits of conservative and optimistic synchronization methods, and thanks to a number of improvements a significant reduction in the number synchronization of messages was achieved.

In the following sections the context of the parallel simulation of wireless networks is presented, then the short description of the methods, the implementation, the results of experiments in distributed environment. At the end of the article we summarize and present the conclusion.

II. Parallel Simulation of Wireless Networks

The maintaining of physical layer accuracy in distributed simulations of wireless networks is a difficult issue, mostly because of the shared type of the medium and the resulting intensity of communication between objects.

To model the complexity of the wireless physical layer it is required, among others, to represent each data frame by at least two events: the beginning (FRAME_START) and end of the transmission (FRAME_END), separately for each simulation object using a shared medium. Additional events are used to model the changing conditions of the medium during transmission (e.g. change of signal power).

When receiving a FRAME_END event, the interference for that frame is evaluated, to determine which signals interfere with the transmission of the frame. The correct frames can be passed to upper layers. This set of interfering signals can be large for large scale simulations [1].

Some of the simulation tools support parallel simulation (e.g., PDNS – distributed version of NS-2 [8], GTNetS [9] or OMNeT++ [10]) in the case of complex simulation models (e.g. INET for TCP/IP and some models of wireless network simulations) but using parallelization to evaluate complex scenario is not possible or is very difficult. Also dedicated

Manuscript received 05 December 2012; revised January 30, 2013.

Sławomir Nowak and Mateusz Nowak are with the Institute of Theoretical and Applied Informatics of PAS, 44-100 Gliwice, ul. Bałtycka 5, Poland (e-mail: emanuel@iitis.pl, mateusz@iitis.pl).

Agnieszka Debudaj-Grabysz is with the Silesian University of Technology, Institute of Informatics (e-mail: Agnieszka.Debudaj-Grabysz@polsl.pl).

DOI: 10.7763/IJCTE.2013.V5.778

simulation tools are available, for parallel simulation of wireless and mobile networks, e.g. MoVeS [11], GloMoSim [12], and related works in [13]-[15].

The presented method allows to increase the simulation performance (and lower the memory usage per single node) by using multi-core and multi-node servers. In the presented study a simplified model of 802.11 was used, but the increase in the complexity of the model is planned as future work.

III. Time Stepped Synchronization Method

In the distributed simulation the scenario is partitioned into a number of LP (*logical processes*). Each LP handles events, arranged by increasing timestamps, and cooperate with the others in accordance using specified synchronization method.

In the proposed method the *time steps* are introduced and the synchronization occurs only at the end of a time step. During the time step each LP operates as an independent, sequential DES. Events created and directed to local simulations objects are stored in *local future events set* (L-FES), and LP processes events with timestamps limited to boundaries of the current time step. The advantage of this approach is the full simulation performance by each LP during a given time step (it corresponds with the optimistic synchronization).

Events directed to simulation's objects, managed by other LPs, are stored in structure called *external future events set* (E-FES) and exchanged during the synchronization phase. External events received by the LP are inserted into the L-FES at the end of the synchronization phase.

Each frame transmission is simulated by two events, related to start of transmission (FRAME_START event) and end of transmission (FRAME_END). The obvious consequence of delayed exchange of external events is the number of causality errors (depends on the length of time step) and the possibility of starting the transmission of the frame, which should not be started due to busy channel.

As external events are delayed, the LP sending such erroneous frame discovers the fact that the channel is busy just at the beginning of next time step (after receiving delayed FRAME_START event). Therefore the FRAME_CANCEL is introduced, to handle such a situation.

The optimistic synchronization assumes *rollback* operations and so called anti-messages (to cancel particular events) to restore correct state of objects and re-simulation of cancelled time-period. Compared to the optimistic synchronization the proposed synchronization method does not require anti-messages, but only single FRAME_CANCEL event. Cancelling frame event does not imply subsequent withdrawal of next events, neither locally nor in other LPs.

For the method to work properly it is necessary to impose additional boundaries on time, in which causality errors can occur and implement appropriate causality error handling at the level of simulation objects. It is possible to restore the synchronization with the single FRAME_CANCEL event when the time step is limited to:

$$\max T_k < \frac{1}{2}\min(t_{FRAME\_END} - t_{FRAME\_START})$$

where T_k is the length of time step k, and t_{FRAME_START} and t_{FRAME_END} means the timestamps of events related to respectively beginning and end of a frame transmission. In other words, the length of a single time step must be less than half the time of minimum frame transmission time in a given scenario.

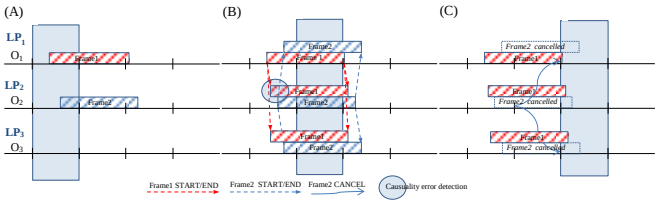

Fig. 1. The causality error in the synchronization method with the time step (A, -B, C: subsequent time steps). (A) Object 1 (O1) detects free channel and send Frame1. O2 detects free channel, and sent Frame2, (B) Obj2 has detected the causality error, cause channel was not free for Frame2 transmission FRAME_CANCEL of Frame2 is sent. O3 has detected both (1 and 2) frames, (C) O1 and O3 have received event FRAME_CANCEL before the FRAME_END of Frame2 will occur and restore the correct state of objects [5].

Adoption of the length of time step consistent with (1) ensures that the transmission of the frame, started in time step k will be delivered to the destination objects in step $k+2$ (or later). Therefore, the causality error may occur in the specific situation (Fig.1): The object O_2 located in LP_2 in time step k at time t_2 checks the status of the wireless medium, and states that it can start sending a frame, so it sends a FRAME_START event. On the next time step $k+1$, the O_2 object is receiving a delayed FRAME_START event from simulation object O_1, located in the LP_1. The timestamp of the event $t_1 < t_2$. This would allow the object to withdraw the state associated with the errorneus frame without generating further events.

This solution involves the need to remember the state of the object (within the time of one time step). The objects must be able to withdraw its state before receiving a FRAME_START event (with restoration of counters, timers, collisions indicators etc.). This is clearly an implementation overhead, and is associated with the implementation of object-level simulation and may be supported only slightly by a kernel of simulator.

IV. Implementation

The method described above has been previously tested in terms of accuracy using the pseudo-parallel implementation [5]. In this paper fully parallel implementation is described as well as the obtained results.

Our algorithm was implemented in C++, using an MPICH2 library for message-passing communication scheme (MPI, *Message Passing Interface*) and gcc as the compiler. MPI is used to exchange the E-FES content (which is the STL array) between processes [16].

Exchanging data comprises two stages: the first - where the information about the number of items sent by each LP is transmitted; the second - where the content of E-FES is distributed. It should be stressed that the amount of data to be sent during the second stage need not be equal at each LPs.

The MPI functions: *MPI_Allgather* and *MPI_Allgatherv* turned out to be the best tools for the first and the second stage, respectively. Both of them are efficiently implemented functions of collective communication.

The most important due to the memory usage and computational of simulations are structures responsible for the interference analysis.

The main structure is the STL map, which stores information about the currently transmitted frames. Each of the transmitted frame structure, in turn, stores information about frames, which interfere with the frame. These structures are required to transfer at the end of the frame to analyze whether the frame was seen as correct and can be forwarded to the upper layers.

Each L2 object a set of frames maintain own Instance of such data structures. For this reason in case of more objects and the broadcast nature of the wireless transmission the overhead associated with operation with threes structures is significant. Distribution of these operations is also a major source of efficiency for distributed simulations.

In addition to above, the mechanism of so called *connectors* objects was also implemented. The connector corresponds to the available transmission channel and intermediates between local and external objects in the transmission of frames for given LP. Simulation objects send frames events directly to the appropriate connector, which performs the following:

1) Frame addressed to local objects is multiplied and inserted into L-FES as FRAME_START and FRAME_END events.
2) Frame addressed to external objects are stored in E-FES in the form of single bulk event, and broadcasted to others LPs during synchronization phase.
3) Bulk event that arrived from external object is sent locally in the same way as in case a).

In our implementation it is assumed that the exchange of messages between LPs in the synchronization phase occurs only between connectors' objects. Exchanging events between individual objects significantly slows down the process of synchronization and the performance of simulation.

In the current implementation, one connector object is assigned to each LP. It is also possible to define multiple connectors, corresponding to a number of communication channels, which will useful to examine more complex protocols (eg, MIMO, planned as future works).

V. Simulation Model and Scenario

The simplified model of communication in a wireless network was adopted (described in details in [17]), since the primary objective of the study was to evaluate the methods for synchronization. This model, however, remains consistent with the *Distributed Coordination Function* (DCF) of the IEEE 802.11-2007 standard [18], [19]. A few mechanisms modeled according the standard are used: *carrier sensing* (CS), *NAV timers* (NAV), *inter-frame spacing* (IFS) and *exponential back-off* (EBO).

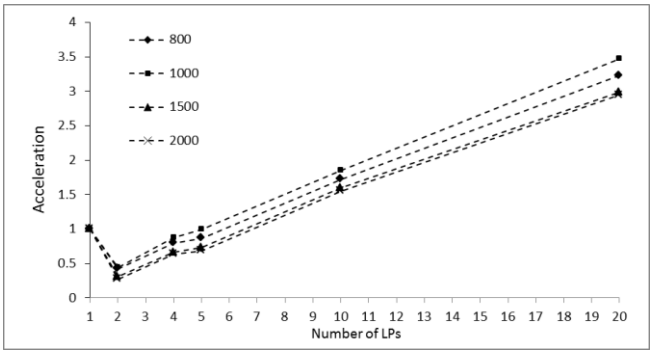

Fig. 2. The acceleration as a function of LPs number (architecture A only).

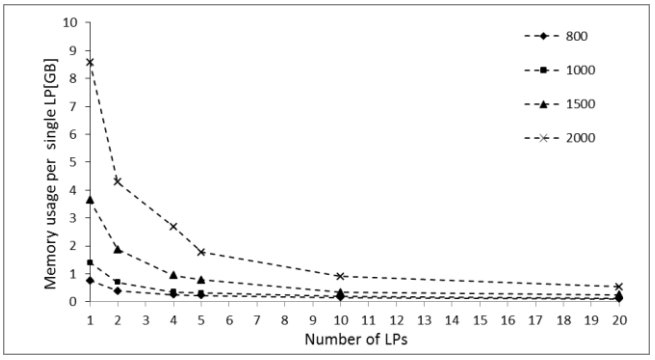

Fig. 3. The memory usage (per single LP) as a function of LPs number (architecture A and B).

Nodes only exchange frames without using acknowledgements (ACK), what reflects a multicast transmission. The RTS/CTS mechanism is not used – it has been poorly adopted by the ISP industry due to its performance and is rarely used nowadays.

The simulation scenario includes *n* hosts, acting as client-server pairs (*n/2* pairs). The client requests a file transfer from a dedicated server and the server responds by sending data. At the application layer a protocol similar to the TFTP protocol [20] is used, using UDP datagrams.

Client objects send requests at random time intervals with uniform distribution (0, 1) s. The transmission rate is set to 1Mbps. The distance (and delay) between any two hosts on the network is fixed.

VI. Experiments

Experiments were carried out on cluster of 4 core-machines (each node is 2×2 core AMD Opteron processor, 8 GB of RAM server) – the *first* architecture (A) and on a 24 core-server (2×12 core AMD Opteron processors, 64 GB of RAM) – the *second* architecture (B).

Numerical data were obtained by running the simulator for scenarios embracing 800, 1000, 1500 and 2000 objects communicating in pairs. The execution time, as well as memory usage was tested.

The speedup obtained after 100000 synchronization points (represents 10s of real time) is presented in Fig. 2. For the number of LP of 10 and more we observe speedup of the simulation (the B architecture). On the other hand, for smaller numbers of LP, synchronization and inter-core communication overhead is too great to gain the benefit of parallelization. However, the profits from distributing the internal data structures among working processors/cores

cannot be ignored. The greater number of LPs, the smaller amount of RAM is needed per a single core (Fig.3). It is especially important for architectures similar to the first one, where it was impossible to run the simulation for 2000 objects with the number of LPs lesser than 4. At that time memory requirements exceeded available 8GB per a node. The described phenomenon is advantageous for number of LPs between 2 and 10. With a bigger number of LPs there is a need to extend the structures for rollback of the simulation, thus the memory requirements are not reduced so much.

For the number of simulating objects lesser than 100 (which is not presented in the figures) we observed no gain as long as time and memory requirements are concerned. The gain from distribution of data structures among cores is too small to compensate for the communication overhead. Additionally, some data structures must be kept indivisibly on each LP and in case of the small number of objects there are too many of them as compared to the portion of structures that can be distributed to observe decrease of memory requirements.

Another undesirable speedup phenomenon is observed for our first architecture. When the number of LPs is greater than 10, then the computing time surges. As was expected the communication between nodes is much slower than communication inside nodes.

The obtained results make us conclude that our simulator is advantageous in terms of speedup on the machines with the architecture of the second type (multi-core server with common memory). For the first type (computing cluster) our simulator may be applied to overcome the limitation of memory available on a single node by collective usage of memory of several nodes.

VII. Conclusions

In the article the implementation of the new method of synchronization as well as and some experimental results were described.

Using a set of solutions (grouping simulation events into a single MPI message, exchanging messages at regular intervals at the end of the time window (synchronization phase), duplication of messages directed to the group of objects at destination's LPs etc.) we achieved a significant reduction in the number synchronization of messages.

The method combines the benefits of optimistic synchronization (no blank messages/null messages and frequent synchronization) while reducing its drawbacks. With the limitation on the length of the synchronization time window, the method prevents the need to send anti-messages by subsequent nodes. As the result there is no need to send anti-messages. So the total overhead of the inter-processes communication has been reduced. It is particularly important in simulations of wireless networks which is a complex problem to simulate, both the computational complexity and memory usage.

The obtained experimental results show that it is possible, under certain conditions and using appropriate hardware architecture (large number of cores), the significant acceleration of distributed simulations of wireless networks.

The additional advantage of the method is the distribution of resources demanded by a model, which reduces memory usage for individual LP.

The authors are aware that the presented model and scenario is simple and largely unrealistic (simple 802.11 protocol, many nodes competing for the link on the relatively small area). At this stage the goal was to examine mainly the synchronization method, for intense interprocess communication.

In future work we will focus on the development of more detailed simulation models of PHY layer and L2 protocols. When developing the model, we expect the growth of computational complexity, so the results of the distributed simulation should which should have a positive effect on the acceleration (in compare to the single node case).

References

[1] E. Ben Hamida, G. Chelius, and J. M. Gorce, "Impact of the physical layer modeling on the accuracy and scalability of wireless network," *Simulation*, vol. 85, pp. 574-588, 2009.

[2] R. Fujimoto, "Parallel and distributed simulation systems," *John Wiley and Sons*, Inc, 2000.

[3] K. Chandy and J. Misra, "Distributed Simulation, a case study in design and verification of distributed programs," *IEEE Transactions on Software Engineering*, vol. 5, no. 5, pp. 440-452, 1979.

[4] D. Jefferson and H. Sowizral, "Fast concurrent simulation using the time warp mechanism," presented at *SCS Conf. Distributed Simulation*, San Diego, CA; (USA); Jan. 1985.

[5] S. Nowak, M. Nowak, and P. Foremski, "New synchronization method for the parallel simulations of wireless networks," *NEW2AN/ruSMART 2011, LNCS 6869*, Springer-Verlag Berlin Heidelberg, pp. 405-416, 2011.

[6] L. Sokol, B. K. Stucky, and V. S. Hwang, "MTW: a control mechanism for parallel discrete simulation," in *Proc. the International Conference on Parallel Processing-ICPP*, pp. 250-254, 1989.

[7] J. S. Steinman, "Breathing time warp," in *Proc. the Seventh Workshop on Parallel and Distributed Simulation*, PADS '93, ACM, New York, pp. 109-118, 1993.

[8] The network simulator. [Online]. Available: http://www.isi.edu/nsnam/ns/.

[9] GTNstS homepage. [Online]. Available: http://www.ece.gatech.edu/research/labs/MANIACS/GTNetS

[10] OMNeT++ [Online]. Available: http://inet.omnetpp.org/.

[11] L. Bononi, M. Felice, G. D. Angelo, M. Bracuto, and L. Donatiello, "MoVES: A framework for parallel and distributed simulation of wireless vehicular ad hoc networks," *Computer Networks*, vol. 52, pp. 155-179, 2008.

[12] X. Zeng, R. Bagrodia, and M. Gerla, "A library for parallel simulation of large-scale wireless networks," in *Proc. the 12th Workshop on Parallel and Distributed Simulation (PADS'98)*, pp. 154-161, 1998.

[13] M. Nowak and S. Nowak, "Parallel simulations with modified INET simulation package," *Theoretical and Applied Informatics*, vol. 19, no. 2, pp. 147-156, 2007.

[14] P. Peschlow, A. Voss, and P. Martini, "Good news for parallel wireless network simulations," in *Proc. the 12th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pp. 134-142, 2009.

[15] H. K. Wu, Ch. Chiang, V. Jha, M. Grela, and R. Bagrodia, "Parallel simulation environment for mobile wireless networks," in *Proc. Winter Simulation Conference*, pp. 605-612,1996.

[16] W. Gropp, E. Lusk, and A. Skjellum, "Using MPI: portable parallel programming with the message-passing interface," MIT Press, ISBN: 0262571323, 1999.

[17] M. Nowak and S. Nowak, "Synchronisation concept for distributed simulation of networks with packet loss," *Studia Informatice*, vol. 30, no. 1, pp. 81-89, 2009.

[18] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Standard 802, Part 11, 2007.

[19] *Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, IEEE Standard 802, Part 11, Amendment 5, 2009.
[20] K. Sollins, *The TFTP Protocol (Revision 2), IETF RFC 1350*, 1992.

Sławomir Nowak was born in 1974, received MSc degrees in Computer Science from Silesian Tchnical University in Gliwice in 2000 and Ph.D. in Computer Science form Institute of Theoretical and Applied Informatics of Polish Academy od Science (IITiS PAN, www.iitis.pl) in 2005. He is presently working as lecturer in IITIS PAN Gliwice and Academy of Business in Dąbrowa Górnicza. Authored many research papers in International/ national Journals and conferences in the field of computer network simulations (based on DES concept).

Agnieszka Debudaj-Grabysz was born in 1973, received MSc degree in Computer Science from Silesian University of Technology in Gliwice, Poland, in 1996, and PhD in Computer Science from the same University, in 2007. In 2005 she got an HPC-Europa Transnational Access scholarship for a short term stay in High Performance Computing Center Stuttgart (HLRS). Currently she is an adjunct professor in the Institute of Computer Science of her alma mater. Her research interests focus on concurrent programming, in particular in applications to solve optimization problems.

Mateusz Nowak was born in 1972, received MSc degree in Computer Science from Silesian University of Technology in Gliwice, Poland, in 1996, and PhD in Computer Science from Institute of Theoretical and Applied Informatics of Polish Academy od Sciences (IITiS PAN, www.iitis.pl) in 2006. He works as an assistant professor at IITIS PAN Gliwice and lecturer at Katowice Institute of Information Technologies. Authored many research papers in international and national journals and conferences in the field of parallel processing, computer network modeling and communication protocols. TPC member of a few networking conferences.