Parallel Network Simulation With OMNeT++

András Varga [1], Ahmet Y. Şekercioğlu [2]
[1] OpenSim Ltd, Budapest, Hungary
Email: andras@omnetpp.org
[2] Centre for Telecommunication and Information Engineering
Monash University, Melbourne, Australia
Email: asekerci@ieee.org

Abstract: **This paper reports a new parallel and distributed simulation architecture for OMNeT++, an open-source discrete event simulation environment. The primary application area of OMNeT++ is the simulation of communication networks. Support for a conservative PDES protocol (the Null Message Algorithm) and the relatively novel Ideal Simulation Protocol has been implemented. Placeholder modules, a novel way of distributing the model over several logical processes (LPs) is presented. The OMNeT++ PDES implementation has a modular and extensible architecture, allowing new synchronization protocols and new communication mechanisms to be added easily, which makes it an attractive platform for PDES research, too. We intend to use this framework to harness the computational capacity of high-performance cluster computers for modeling very large scale telecommunication networks to investigate protocol performance and rare event failure scenarios.**

Keywords: Parallel simulation, discrete-event simulation, PDES

I. INTRODUCTION

Telecommunication networks are increasingly becoming more complex as the trend toward the integration of telephony and data networks into integrated services networks gains momentum. It is expected that these integrated services networks will include wireless and mobile environments as well as wired ones. As a consequence of the rapid development, reduced time to market, fusion of communication technologies and rapid growth of the Internet, predicting network performance, and eliminating protocol faults have become an extremely difficult task. Attempts to predict and extrapolate the network performance in small-scale experimental testbeds may yield incomplete or contradictory outcomes. Application of analytical methods is also not feasible due to the complexity of the protocol interactions, analytical intractability and size [1]. For large scale analysis in both the spatial and temporal domain, accurate and detailed models using parallel simulation techniques offer a practical answer. It should be noted that simulation is now considered as a tool of equal importance and complementary to the analytical and experimental studies for investigating and understanding the behavior of various complex systems such as climate research, evolution of solar system and modeling nuclear explosions.

This paper reports about the results of implementing parallel simulation support in the OMNeT++ discrete event simulation tool [17]. OMNeT++ is a useful framework for creating various simulation models to evaluate the performance of various algorithms, mechanisms and solutions in telecommunication networks [21-25]. The user community meets at annual workshops [26, 27]. The parallel simulation project has been motivated by and forms part of our ongoing research programs at CTIE, Monash University on the analysis of protocol performance of large-scale mobile IPv6 networks. We have developed a set of OMNeT++ models for accurate simulation of IPv6 protocols [7]. We are now focusing our efforts to simulate mobile IPv6 networks in very large scale.

For this purpose, we intend to use the computational capacity of APAC (http://www.vpac.org) and VPAC (http://www.apac.edu.au) supercomputing clusters. In a series of future articles, we will be reporting our related research on synchronization methods, efficient topology partitioning for parallel simulation, and topology generation for mobile/wireless/cellular Internet.

II. PARALLEL SIMULATION OF COMMUNICATION NETWORKS TODAY

Discrete event simulation of telecommunications systems is generally a computation intensive task. A single run of a wireless network model with thousands of mobile nodes may easily take several days and even weeks to obtain statistically trustworthy results even on today's computers, and many simulation studies require several simulation runs [1].

Independent replicated simulation runs have been proposed to reduce the time needed for a simulation study, but this approach is often not possible (for example, one simulation run may depend on the results of earlier runs as input) or not practical. Parallel discrete event simulation (PDES) offers an attractive alternative. By distributing the simulation over several processors, it is possible to achieve a speedup compared to sequential (one-processor) simulation.

Another motivation for PDES is distributing resource demand among several computers. A simulation model often exceeds the memory limits of a single workstation. Even though distributing the model over several computers and controlling the execution with PDES algorithms may result in slower execution than on a single workstation (due to communication and synchronization overhead in the PDES mechanism), but at least it is possible to run the model. It is a recent trend that clusters (as opposed to shared memory multiprocessors) are becoming an attractive PDES platform [12], mainly because of their excellent price/performance ratio. Also, very large-scale network simulations demand computing capacity that can only be provided with cluster computing at affordable costs.

Despite about 15-20 years on research on parallel discrete event simulation (see e.g.[3]), PDES is today still more of a promise than part of everyday practice. Fujimoto, a PDES veteran [4], recently expressed this as: "Parallel simulation provides a benefit, but it has to be transparent, automatic, and virtually free in order to gain widespread acceptance. Today it ain't. It may never be." [5]

What parallel simulation tools are available today for the communication networks research community ? A parallel simulation extension for the traditionally widely used ns2 simulator has been created at the Georgia Institute of Technology [11], but it is not in wide use. SSFNet [15] claims to be a standard for parallel discrete event network simulation. SSFNet's Java implementation is becoming popular in the research community, but SSFNet for C++ (DaSSF) does not seem to receive nearly as much attention, probably due to the lack of network protocol models. J-Sim [6], another popular network simulation environment does not have PDES support. Parsec [1] with its GloMoSim library have morphed into the commercial Qualnet network simulation product [13]. The optimistic parallel simulation tool SPEEDES [14] [16] has similarly become commercial, and it is apparently not being used for simulation of communication networks.

The best-known commercial network simulation tool, OPNET [10], does support parallel simulation, but little has been disclosed about it. It appears that OPNET simulations can make use of multiprocessor architectures, but cannot run on clusters.

Apparently, the choice is limited for communication networks research groups that intend to make use of parallel simulation techniques on clusters. SSFNet for

Java appears to be a feasible choice, but in the C/C++ world there is probably no really attractive choice today. The project effort published in this paper attempts to improve this situation, and there is a good chance that OMNeT++ can fill this niche.

III. PARALLEL SIMULATION SUPPORT IN OMNET++

A. About OMNeT++

OMNeT++ [17] is a discrete event simulation environment. The primary application area of OMNeT++ is the simulation of communication networks, but because of its generic and flexible architecture, it has been successfully used in other areas like the simulation of complex IT systems, queueing networks or hardware architectures as well. OMNeT++ is rapidly becoming a popular simulation platform in the scientific community as well as in industrial settings. The distinguishing factors of OMNeT++ are its strongly component-oriented approach which promotes structured and reusable models, and its extensive graphical user interface (GUI) support. Due to its modular architecture, the OMNeT++ simulation kernel (and models) can be easily embedded into your applications. OMNeT++ is open-source and free for academic and non-profit use.

An OMNeT++ model consists of modules that communicate with message passing. The active modules are termed simple modules; they are written in C++, using the simulation class library. Simple modules can be grouped into compound modules. Both simple and compound modules are instances of module types. While describing the model, the user defines module types; instances of these module types serve as components for more complex module types. Finally, the user creates the system module as an instance of a previously defined module type.

Modules communicate with messages which – in addition to usual attributes such as timestamp – may contain arbitrary data. Simple modules typically send messages via gates, but it is also possible to send them directly to their destination modules.

Gates are the input and output interfaces of modules: messages are sent out through output gates and arrive through input gates. An input and an output gate can be linked with a connection. Connections are created within a single level of module hierarchy: within a compound module, corresponding gates of two submodules, or a gate of one submodule and a gate of the compound module can be connected.

Due to the hierarchical structure of the model, messages typically travel through a chain of connections, to start and arrive in simple modules. Compound modules act as "cardboard boxes" in the model, transparently relaying messages between their inside and the outside world. Connections can be assigned properties such as propagation delay, data rate and bit error rate.

B. PDES Features

This section introduces the new PDES architecture in OMNeT++ [19] (OMNeT++ has had experimental, statistical synchronization-based PDES support [20] before our work). In its current form, the new architecture supports conservative synchronization via the classic Chandy-Misra-Bryant (or Null Message) Algorithm [3] over MPI, and accommodates extension points to implement other synchronization mechanisms and other transport layers as well.

The OMNeT++ design places a big emphasis on *separation of models from experiments*. The main rationale is that usually a large number of simulation experiments need to be done on a single model before a conclusion can be drawn about the real system. Experiments tend to be ad-hoc and change much faster than simulation models, thus it is a natural requirement to be able to carry out experiments without changing the simulation model itself.

Following the above principle, OMNeT++ allows simulation models to be executed in parallel without modification. No special instrumentation of the source code or the topology description is needed, as partitioning and other PDES configuration is entirely described in the configuration files (in contrast, ns2

requires modification of the Tcl code, and SSFNet requires modification of the DML file(s)).

OMNeT++ supports the Null Message Algorithm (NMA) with static topologies, using link delays as lookahead. The laziness of null message sending can be tuned. Also supported is the Ideal Simulation Protocol (ISP) introduced by Bagrodia in 2000 [2].

ISP is a powerful research vehicle to measure the efficiency of PDES algorithms, optimistic or conservative; more precisely, it helps determine the maximum speedup achievable by any PDES algorithm for a particular model and simulation environment. In OMNeT++, ISP can be used for benchmarking the performance of the NMA. Additionally, models can be executed without any synchronization, which can be useful for educational purposes (to demonstrate the need for synchronization) or for simple testing.

For the communication between logical processes (LPs), OMNeT++ primarily uses MPI, the Message Passing Interface standard [9]. An alternative communication mechanism is based on named pipes, for use on shared memory multiprocessors without the need to install MPI. Additionally, a file system based communication mechanism is also available. It communicates via text files created in a shared directory, and can be useful for educational purposes (to analyze or demonstrate messaging in PDES algorithms) or to debug PDES algorithms.

Implementation of a shared memory-based communication mechanism is also planned for the future, to fully exploit the power of multiprocessors without the overhead of and the need to install MPI.

Nearly every model can be run in parallel. The constraints are the following:

- modules may communicate via sending messages only (no direct method call or member access) unless mapped to the same processor
- no global variables.
- there are some limitations on direct sending (no sending to a *sub*module of another module, unless mapped to the same processor)
- lookahead must be present in the form of link delays
- currently static topologies are supported

PDES support in OMNeT++ follows a modular and extensible architecture.

New communication mechanisms can be added by implementing a compact API (expressed as a C++ class) and registering the implementation – after that, the new communications mechanism can be selected in the configuration file.

New PDES synchronization algorithms can be added in a similar way. PDES algorithms are also represented by C++ classes that have to implement a compact API to integrate with the simulation kernel. Setting up the model on various LPs as well as relaying model messages across LPs is already taken care of and not something the implementation of the synchronization algorithm needs to worry about it (although it can intervene if needed, because the necessary hooks are present).

The implementation of the NMA is also modular in itself in that a lookahead discovery mechanism can be plugged in via a defined API. Currently implemented lookahead discovery uses link delays, but it is possible to implement more sophisticated ones and select them through the configuration file.

C. Parallel Simulation Example

For demonstrating PDES capabilities of OMNeT++, we use the closed queuing network (CQN) model described in [2]. The model consists of N tandem queues where each tandem consists of a switch and k single-server queues with exponential service times (Figure 1). The last queues are looped back to their switches. Each switch randomly chooses the first queue of one of the tandems as destination, using uniform distribution. The queues and switches are connected with links that have nonzero propagation

delays. Our OMNeT++ model for CQN wraps tandems into compound modules.

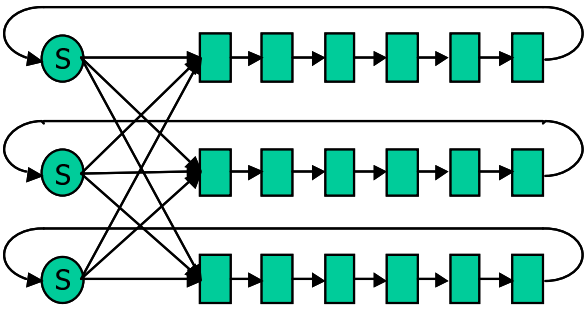


Figure1: The Closed Queueing Network (CQN) model

To run the model in parallel, we assign tandems to different LPs (Figure 2). Lookahead is provided by delays on the marked links.

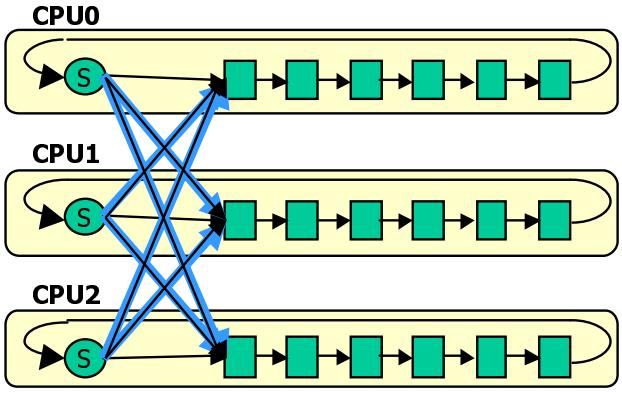


Figure2: Partitioning the CQN model

To run the CQN model in parallel, we have to configure it for parallel execution. In OMNeT++, the configuration is in a text file called omnetpp.ini. For configuration, first we have to specify partitioning, that is, assign modules to processors. This is done with the following lines:

```
[Partitioning]
*.tandemQueue[0].partition-id = 0
*.tandemQueue[1].partition-id = 1
*.tandemQueue[2].partition-id = 2
```

The numbers after the equal sign identify the LP. Also, we have to select the communication library and the parallel simulation algorithm, and enable parallel simulation:

```
[General]
parallel-simulation=true
parsim-communications-class="cMPICommunications"
parsim-synchronization-class="cNullMessageProtocol"
```

When the parallel simulation is run, LPs are represented by multiple running instances of the same program. When using LAM-MPI [8], the mpirun program (part of LAM-MPI) is used to launch the program on the desired processors. When named pipes or file communications is selected, the opp_prun OMNeT++ utility can be used to start the processes. Alternatively, one can launch the processes manually:

```
./cqn -p0,3 &
./cqn -p1,3 &
./cqn -p2,3 &
```

Here, the -p flag tells OMNeT++ the index of the given LP and the total number of LPs. For PDES, one will usually want to select the command-line user interface of OMNeT++, and redirect the output to files (OMNeT++ provides the necessary configuration options.)

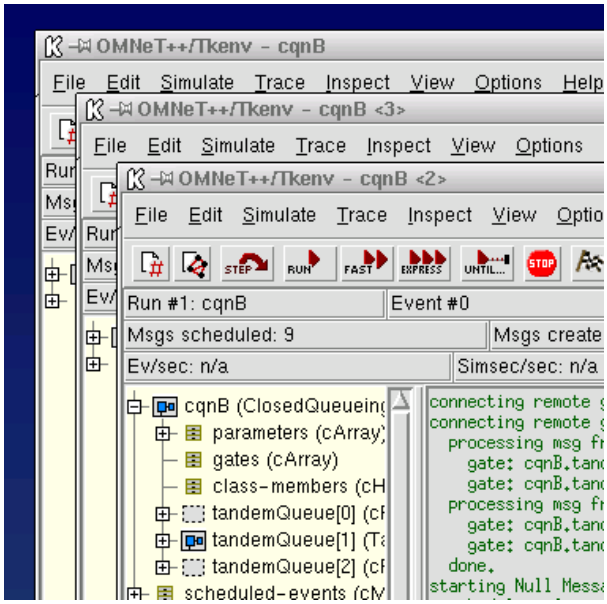


Figure 3: Screenshot of CQN running in three LPs

The GUI of OMNeT++ can also be used (as evidenced by Figure 3), independent of the selected communication mechanism. The GUI interface can be useful for educational or demonstration purposes as OMNeT++ shows the operation of NMA in a log

window, and one also can examine EIT and EOT values.

D. Instantiation of Modules

When setting up a model partitioned to several LPs, OMNeT++ uses placeholder modules and proxy gates. In the local LP, placeholders represent sibling submodules that are instantiated on other LPs. With placeholder modules, every module has all of its siblings present in the local LP – either as placeholder or as the "real thing". Proxy gates take care of forwarding messages to the LP where the module is instantiated (see Figure 4).

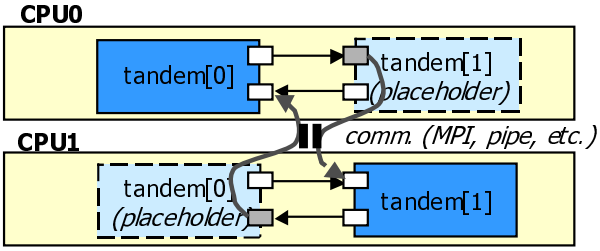


Figure 4: Placeholder modules and proxy gates

The main advantage of using placeholders is that algorithms such as topology discovery embedded in the model can be used with PDES unmodified. Also, modules can use direct message sending to any sibling module, including placeholders. This is so because the destination of direct message sending is an input gate of the destination module, thus if the destination module is a placeholder, the input gate will be a proxy gate which transparently forwards the messages to the LP where the "real" module was instantiated.

A limitation is that the destination of direct message sending cannot be a *submodule* of a sibling (which is probably a bad practice anyway, as it violates encapsulation), simply because placeholders are empty and so its submodules are not present in the local LP.

Instantiation of compound modules is slightly more complicated. Since its submodules can be mapped to different LPs, the compound module may not be "fully present" on any given LP, and it may forced to be present on several LPs (on all LPs where if one or more submodules instantiated). Thus, compound modules are instantiated wherever they have at least one submodule instantiated, and are represented by placeholders everywhere else (Figure 5).

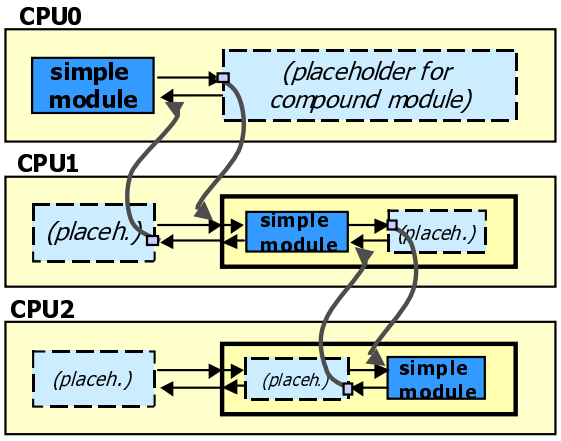


Figure 5: Instantiating compound modules

E. Performance Measurements

We have made several runs with the CQN model on 2 and 4 processors, with the following parameters: $N=16$ tandem queues, $k=10$ and 50 queues per tandem, with lookahead $L=1$, *5* and *10*. The hardware environment was a Linux cluster (kernel 2.4.9) of dual 1 Ghz Pentium III PCs, interconnected using a 100Mb Ethernet switch. The communication library was LAM-MPI [8]. The MPI latency was measured to be 22μs. Sequential simulation of the CQN model achieved P_{seq}=120,000 events/sec performance.

We executed simulations under NMA and (for comparison) under ISP.

The results are summarized in Table1. P_{ISP}, P_{NMA} are the performance (events/second) under the ISP and the NMA protocol, and S_{ISP}, S_{NMA} are the speedups under ISP and NMA, respectively. It can be observed that the L lookahead strongly affects performance under NMA. An analysis of NMA performance versus lookahead and other performance factors can be found in [18]. However, it is probably too early to draw conclusions from the figures below about the performance of the OMNeT++ parallel simulation

implementation, because we are still optimizing the code.

Table1: Comparison of NMA and ISP simulations

Configuration			Performance			
#LPs	*k*	*L(s)*	P_{ISP} *(ev/s)*	P_{NMA} *(ev/s)*	S_{ISP}	S_{NMA}
2	10	1	147,618	76,042	1.23	0.63
2	10	5	151,250	143,289	1.26	1.19
2	10	20	157,200	153,600	1.31	1.28
2	50	1	168,830	131,398	1.41	1.09
2	50	5	170,289	164,563	1.42	1.37
2	50	20	172,811	173,249	1.44	1.44
4	10	1	300,479	45,190	2.50	0.38
4	10	5	311,392	148,007	2.59	1.23
4	10	20	314,892	271,648	2.62	2.26
4	50	1	359,517	144,979	3.00	1.21
4	50	5	364,663	284,978	3.04	2.37
4	50	20	372,844	352,557	3.11	2.94

IV. DESIGN OF PDES SUPPORT IN OMNET++

Design of PDES support in OMNeT++ follows a layered approach, with a modular and extensible architecture. The overall architecture is depicted in Figure 6.

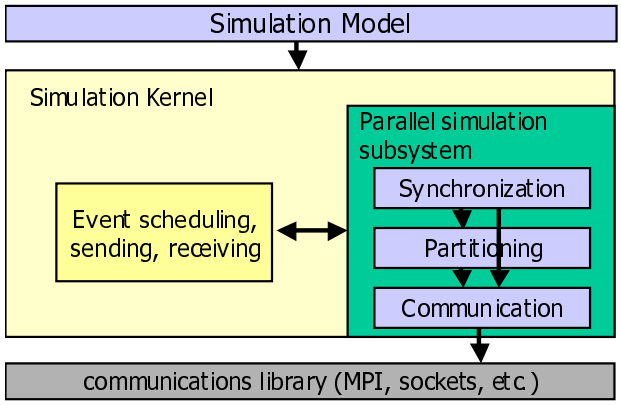


Figure 6: Architecture of OMNeT++ PDES implementation

The parallel simulation subsystem is an optional component itself, which can be removed from the simulation kernel if not needed. It consists of three layers, from the bottom up: communication layer, partitioning layer and synchronization layer.

The purpose of the *Communications Layer* is to provide elementary messaging services between partitions for upper layer. The services include send, blocking receive, non-blocking receive and broadcast. The send/receive operations work with *buffers*, which encapsulate packing and unpacking operations for primitive C++ types. The message class and other classes in the simulation library can pack and unpack themselves into such buffers. The Communications Layer API is defined in the cFileCommunications interface (abstract class); concrete implementations like the MPI one (cMPICommunications) subclass from this, and encapsulate MPI send/receive calls. The matching buffer class cMPICommBuffer encapsulates MPI pack/unpack operations.

The *Partitioning Layer* is responsible for instantiating modules on different LPs according to the partitioning specified in the configuration, for configuring proxy gates. During the simulation, this layer also ensures that cross-partition simulation messages reach their destinations. It intercepts messages that arrive at proxy gates, and transmits them to the destination LP using the services of the communication layer. The receiving LP unpacks the message and injects it at the gate pointed to be the proxy gate.

The implementation basically encapsulates the cParsimPartition, cPlaceholderModule and cProxyGate classes.

The *Synchronization Layer* encapsulates the parallel simulation algorithm. Parallel simulation algorithms are also represented by classes, subclassed from the cParsimSynchronizer abstract class. The parallel simulation algorithm is invoked on the following hooks: event scheduling, processing model messages outgoing from the LP, and messages (model messages or internal messages) arriving from other LPs. The first hook, event scheduling is a function invoked by the simulation kernel to determine the next simulation event; it also has full access to the future event list (FEL) and can add/remove events for its own use.

Conservative parallel simulation algorithms will use this hook to block the simulation if the next event is unsafe, e.g. the null message algorithm implementation (cNullMessageProtocol) blocks the

simulation if an EIT has been reached until a null message arrives (see [2] for terminology); also it uses this hook to periodically send null messages. The second hook is invoked when a model message is sent to another LP; the NMA uses this hook to piggyback null messages on outgoing model messages. The third hook is invoked when any message arrives from other LPs, and it allows the parallel simulation algorithm to process its own internal messages from other LPs; the NMA processes incoming null messages here.

The null message protocol implementation itself is modular as it employs a separate, configurable lookahead discovery object. Currently only link delay based lookahead discovery has been implemented, but it is possible to implement more sophisticated ones.

The ISP implementation, in fact, consists of two parallel simulation protocol implementations: the first one is based on the NMA and additionally records the external events (events received from other LPs) to a trace file; the second one runs the simulation using the trace file to find out which events are safe and which are not.

Note that although we implemented a conservative protocol, the provided API itself would allow implementing optimistic protocols, too. The parallel simulation algorithm has access to the executing simulation model, so it could perform saving / restoring model state if the code of the simulation model supports this (unfortunately, support for state saving/restoration needs to be individually and manually added to each class in the simulation, including user-programmed simple modules).

We also expect that because of the modularity, extensibility and clean internal interfaces of the parallel simulation subsystem, the OMNeT++ framework has the potential to become a preferred platform for PDES research.

V. CONCLUSION

The paper presented a new parallel simulation architecture for OMNeT++. A merit of the implementation is that it features the "separation of experiments from models" principle, and thus allows simulation models to be executed in parallel without modification. It relies on a novel approach of placeholders to instantiate the model on different LPs.

The placeholder approach allows simulation techniques such as topology discovery and direct message sending to work unmodified with PDES. The architecture is modular and extensible so it may serve as a potential framework for research on parallel simulation.

REFERENCES

[1] R. L. Bagrodia, R. Meyer, M. Takai, Y. Chen, X. Zeng, J. Martin, and H. Y. Song. Parsec: *A parallel simulation environment for complex systems*. IEEE Computer, pages 77–85, October 1998.

[2] R. L. Bagrodia and M. Takai. *Performance evaluation of conservative algorithms in parallel simulation languages*. IEEE Trans. on Parallel and Distributed Systems, 11(4):395–414, 2000.

[3] M. Chandy and J. Misra. *Distributed simulation: A case study in design and verication of distributed programs*. IEEE Transactions on Software Engineering SE-5, (5). 440–452, 1979.

[4] R. M. Fujimoto. *Parallel discrete event simulation*. Communications of the ACM, 33(10):30–53, October 1990.

[5] R. M. Fujimoto. *Parallel and distributed simulation in the 21th century*. In Grand Challenges for Modeling and Simulation (Seminar 02351), 26-30 August 2002, Dagstuhl Castle, Germany, 2002.

[6] J-Sim home page. http://www.j-sim.org.

[7] J. Lai, E. Wu, A. Varga, Y. A. Şekercioğlu, and G. K. Egan. *A simulation suite for accurate modeling of IPv6 protocols*. In Proceedings of the 2nd OMNeT++ Workshop, pages 2–22, Berlin, Germany, January 2002.

[8] LAM-MPI home page. http://www.lam-mpi.org/.

[9] MPI: *A message-passing interface standard. International Journal of Supercomputer Applications*, 8(3/4):165–414, 1994. Message Passing Interface Forum.

[10] OPNET. *OPNET Technologies*, Inc. home page. http://www.opnet.com/.

[11] Atlanta PADS Research Group, *Georgia Institute of Technology*. PDNS-Parallel/Distributed NS home page. http://www.cc.gatech.edu/computing/compass/pdns.

[12] C. D. Pham. High performance clusters: *A promising environment for parallel discrete event simulation*. In Proceedings of the PDPTA'99, June 28-July 1, 1999, Las Vegas, USA, 1999.

[13] QualNet home page. http://www.qualnet. com

[14] SPEEDES home page. http://www.speedes.com

[15] SSFNet home page. http://www.ssfnet.org

[16] J. Steinman. *Scalable parallel and distributed military simulations using the SPEEDES framework*. ELECSIM'95, 2nd Electronic Simulation Conference, Internet, May-June, 1995.

[17] A. Varga. *The OMNeT++ discrete event simulation system*. In Proceedings of the European Simulation Multiconference (ESM'2001). June 6-9, 2001. Prague, Czech Republic, 2001.

[18] A. Varga, Y. A. Şekercioğlu, and G. K. Egan. *A practical efficiency criterion for the null message algorithm*. In Proceedings of the European Simulation Symposium (ESS2003), Oct. 2003, Delft, The Netherlands. Society for Computer Simulation, 2003.

[19] Y. A. Şekercioğlu, A. Varga, and G. Egan. *Parallel simulation made easy with OMNeT++*. In Proceedings of the European Simulation Symposium (ESS2003), Oct. 2003, Delft, The Netherlands. Society for Computer Simulation, 2003.

[20] G. Lencse. *Parallel simulation with OMNeT++ using the Statistical Synchronization Method*. In Proceedings of the 2nd OMNeT++ Workshop, pp. 24—32, Berlin, 2002.

[21] I. Dietrich and F. Dressler. *On the lifetime of wireless sensor networks*. ACM Trans. on Sensor Networks, 5(1):1–39, 2009.

[22] R. Cuevas, A. Cabellos-Aparicio, A. Cuevas, J. Domingo-Pascual and A. Azcorra. fP2P-HN: *A P2P-based route optimization architecture for mobile IP-based community networks*. Computer Networks, 53(4):528–540, 2009.

[23] H. Wang, N. Agoulmine, M. Ma, Y. Li and X. Wang. *Network Lifetime Optimization by KKT Optimality Conditions in Wireless Sensor Networks*. Wirel. Pers. Commun., 49(2):179–196, 2009.

[24] J. Glaser, D. Weber, S. Madani and S. Mahlknecht. *Power aware simulation framework for wireless sensor networks and nodes*. EURASIP J. Embedded Syst. 2008(3):1–16, 2008.

[25] K. S. Kim, D. Gutierrez, F. An and L. Kazovsky. *Design and Performance Analysis of Scheduling Algorithms for WDM-PON under SUCCESS-HPON Architecture*. IEEE/OSA J. Lightwave Techn. 23(11):3716–3731, 2005.

[26] Proceedings of the 1st International Workshop on OMNeT++. Marseille, France: ICST, 2008.

[27] Proceedings of the 2nd International Workshop on OMNeT++. Rome, Italy: ICST, 2009.

AUTHOR BIOGRAPHIES

András Varga received his M.Sc. in computer science with honors from the Technical University of Budapest, Hungary in 1994. He is the author of the OMNeT++ open-source network simulation tool currently widely used in academic and industrial settings. He worked for several years as software architect for Encorus (formerly Brokat Technologies) before founding OpenSim Ltd that develops OMNeT++, and Simulcraft Inc. that provides commercial licenses and services for OMNeT++ worldwide. He is currently working towards PhD, his research topic being large-scale simulation of communication networks. Between February and September 2003, and between February and June 2005 he visited CTIE at Monash University (Melbourne, Australia) to participate in parallel simulation research.

Ahmet Y. Şekercioğlu is a researcher at the Centre for Telecommunications and Information Engineering (CTIE) and a Senior Lecturer at Electrical and Computer Systems Engineering Department of Monash University, Melbourne, Australia. Between 1998 and 2006 he also held the position of Program Leader for the Applications Program of Australian Telecommunications Cooperative Research Centre (ATcrc, http://www.atcrc.com). He completed his PhD degree at Swinburne University of Technology, Melbourne, Australia (2000), MSc (1985) and BSc (1982) degrees at Middle East Technical University, Ankara, Turkey (all in Electrical Engineering). He has lectured at Swinburne University of Technology for 8 years, and has had numerous positions as a research engineer in private industry. His recent work focuses on development of tools for simulation of large-scale telecommunication networks. He is also interested in application of intelligent control techniques for multiservice networks as complex, distributed systems.

An Efficient Statistical Synchronization Method For Parallel Simulation

Gábor Lencse
Department of Telecommunications
Széchenyi István University, Győr, Hungary
Email: lencse@sze.hu

Abstract: **In this paper, we propose the improved Statistical Synchronization Method (SSM-T) for parallel discrete event simulation. Criteria are given for the time-driven approach (SSM-T). It is proven that the level of the output error can be guaranteed. SSM-T is implemented in the OMNeT++ discrete event simulation tool, which is a useful and widespread framework for creating various simulation models to evaluate the performance of telecommunication networks. Case studies have been performed, which shows that SSM-T is a very efficient synchronization method for the parallel simulation of communication networks.**

Keywords: Paralleldiscrete event simulation, statistical synchronization, applicability criteria, efficiency, accuracy.

I. 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 simulation of large and complex networks is often a practical need when they are designed or analyzed. In many cases, the only option for the execution of the simulation is the use of a cluster of workstations. Due to the nature of the algorithm of the event driven discrete event simulation the parallelization is not an easy task.

The conventional synchronization methods for parallel discrete event simulation (e.g, conservative, optimistic) [2] 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" [3] often produces excessive rollbacks and inter-processor communication.

The Statistical Synchronization Method (SSM) [16] is a promising alternative to the conventional synchronization methods for parallel discrete event simulation. Unlike the conventional synchronization 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 explanations 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 the next generation networks. An addition advantage of SSM is that it is relatively easy to extend existing non-parallel simulation software for use with SSM, which is not necessarily true for other synchronization methods.

In this paper, we propose the improved Statistical Synchronization Method (SSM-T) for parallel discrete event simulation. SSM-T is implemented in the OMNeT++ discrete event simulation tool [8,10,18,19], which is a useful and widespread framework for creating various simulation models to evaluate the performance of telecommunication networks.

The remainder of this paper is organized as follows: after a brief introduction to SSM and SSM-T in Section II, the applicability criteria for SSM-T are given in an informal way in Section III. They are formalized in Section IV together with a proof that the required level of the output error can be guaranteed by satisfying the criteria. Next in Section V, we show positive and negative examples to give a better insight of the criteria. Afterwards in Section VI, we present the conditions for a good speed-up. Finally in section VII, we give the conclusions of the paper.

II. THE STATISTICAL SYNCHRONIZATION METHOD

A. The Original SSM

For those not familiar with SSM, a short summary of the Statistical Synchronization Method [16] 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 (Fig. 1).

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.

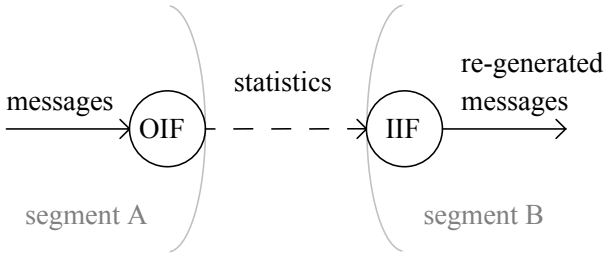


Figure 1: An OIF - IIF pair

B. SSM-T: Refinement of SSM

The original SSM does not explicitly state when the OIF's should send their statistics to the appropriate IIF's. The results of [6,7] would suggest that the statistics collection must be continued until the sample contains the required number of observations, then the statistics should be sent and the statistics collection should be restarted. This was called *SSM-C* (the counter driven approach) in [5]. In that paper, *SSM-T* (the time driven approach) was proposed for parallel simulation, which works as follows.

Using the notations of Fig. 1, at the beginning of the simulation the OIF of segment A must tell the IIF of segment B at what virtual time it will send the first statistics. This is t_1 in Fig. 2. In this figure the thin horizontal lines show the wall-clock (real) time of the processors executing the segments and the thick lines are the virtual times of the segments. Segment B takes into consideration the first statistics from segment A at t_1 virtual time. It is done in the following way: In the figure, segment B receives the first statistics from segment A at t_x (according to its own virtual time) and as $t_x<t_1$, segment B schedules the arrival of the statistics for t_1. The other possibility is shown on the

example of t_2. Segment B has not received the statistics until t_2, and it has no more events scheduled with less than or equal time stamp, so the execution of the simulation of segment B is suspended until the statistics arrive from segment A. Then segment B receives the statistics and the execution resumes. Segment B always knows at what virtual time to expect the next statistics, because Segment A always includes the virtual arrival (=sending) time of the i-th statistics in the (i-1)th statistics package. We called this solution *loose synchronisation* [5]. This method makes it possible for the simulation of the segments to run independently on separate processors in the vast majority of time and therefore it may result in excellent speed-up.

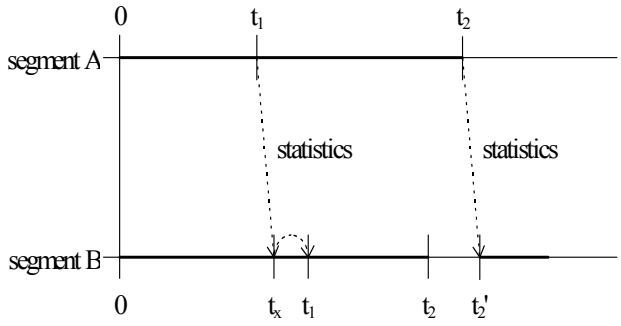


Figure 2: The operation of SSM-T. See the text for explanation.

III. THE APPLICATBILITY CRITERIA OF SSM-T

SSM-T can be applied and produces meaningful results if the following conditions are met:

(a) The simulated system can be divided into segments so that not the individual messages but only their statistical characteristics are important between the segments.

(b) A *small error* in the approximation of the statistical characteristics of the message flow causes *small error* in the output of the simulation that *depends only on the measure of the approximation error*.

(c) The parameters of the model may change during the simulation but the changes in the statistical characteristics of the message flows between the segments are *rare enough*.

Note that a change in the statistical characteristics of the message flow is only propagated to other segments when the statistics package is actually sent over by the OIF. The fourth condition is that this delay causes error in the output of the simulation only during the delay and/or at most during an additional time interval with a length proportional to the delay.

IV. THE OUTPUT ERROR OF SIMULATION WITH SSM-T

Let us denote the concerned statistical characteristics of the message flow by the random variable $\boldsymbol{X}$ and its approximation by $\boldsymbol{X}^*$. The error of the approximation is also a random variable: $h_{\boldsymbol{X}}=\boldsymbol{X}^*-\boldsymbol{X}$. The observed output of the simulation is denoted by $\boldsymbol{O}$. The $h_{\boldsymbol{X}}$ error of the approximation causes $h_{\boldsymbol{O}} = \boldsymbol{O}^*-\boldsymbol{O}$ error in the output. Condition (b) is defined formally as follows:

(b') $h_{\boldsymbol{O}} = f(h_{\boldsymbol{X}})$ for some f, and $\forall\varepsilon>0\ \exists\delta: |h_{\boldsymbol{X}}|<\delta \Rightarrow |h_{\boldsymbol{O}}|<\varepsilon$.

Note, that sometimes the following conditions may be enough:

(b") $h_{\boldsymbol{O}}=f(h_{\boldsymbol{X}})$ for some f, and $\forall\varepsilon>0\ \exists\delta: E\{h_{\boldsymbol{X}}\}<\delta \Rightarrow E\{h_{\boldsymbol{O}}\}<\varepsilon$, where $E\{\}$ is the expected value of the random variable.

Let us denote the number of observations in a sample by n. $\forall\delta>0\ \exists N: n>N \Rightarrow |h_{\boldsymbol{X}}|<\delta$. The value of N depends on both the required value of δ and the convergence speed of the statistics collection method used. See [6] for more details about the convergence speed of some well-know statistics collection methods.

In the stationary case we can guarantee $|h_{\boldsymbol{O}}|<\varepsilon$ with the appropriate choice of N. Let us consider the transients in the system. If the distribution of the random variable $\boldsymbol{X}$ changes in segment A at t_c in the i-th sample collection interval, the exact new statistics arrive at segment B at the t_{i+1} synchronization point of

virtual time (Fig. 3). If T_N is an upper bound for the length of any $(t_{i+1}-t_i)$ interval, then the length of the transient $T_{TR}<2T_N$.

Condition (d) says that the transient may cause output error during the T_{TR} time of the transient, plus maximum c_eT_{TR} time after it. (c_e is an appropriate constant). Thus the output of the simulation may contain an error due to the transient caused by SSM-T less than $c_e'T_N$ time, where $c_e'=2(1+c_e)$.

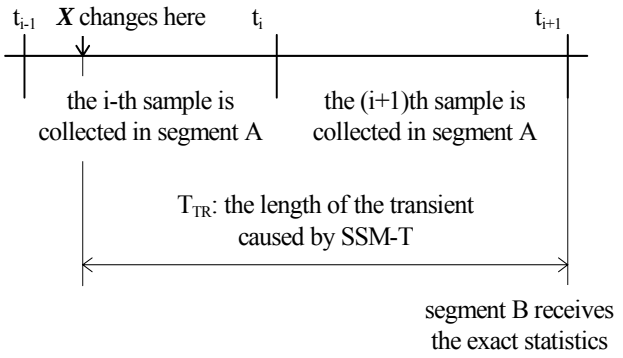


Figure3: The transient caused by SSM-T

Let us denote the time elapses from the end of the $c_e'T_N$ time interval to the next change of ***X*** by T_{QST} , the time of the quasi stationary state. The empirical meaning of "*rare enough"* in condition (b) is: $c_e'T_N << T_{QST}$. Let us make it a bit more formal: H_O is an upper bound for the absolute value of the output error during the transient caused by SSM-T. Now, we show that $|\bar{h}_o|<\varepsilon$ can be ensured. Let us choose δ': $|h_X|<\delta' \Rightarrow |h_O|<½\varepsilon$ during the T_{QST} time period. Let N_{TR} and N_{QST} denote the number of the collected output statistics values during the $c_e'T_N$ and the T_{QST} time periods, respectively. The before mentioned "*rare enough"* condition is:

$$(c') \quad N_{QST} \geq \frac{2N_{TR}H_O}{\varepsilon} - N_{TR}$$

The average output error is:

$$|\bar{h}_O| \leq \left|\frac{N_{TR}H_O}{N_{QST}+N_{TR}}\right| + \left|\frac{N_{QST}h_O}{N_{QST}+N_{TR}}\right| \leq ½\varepsilon+½\varepsilon \leq \varepsilon$$

However, the condition for N_{QST} is quite strict, and it is not always necessary. We have eliminated the error of the output of the simulation by averaging very many values. If we know the time of the changes of ***X*** in advance, or if we can detect the change, we can just omit the N_{TR} number of output statistics values with error. By doing so, a lot of virtual time (and therefore simulation execution time too) may be saved, because in this way only the $c_e'T_N$ virtual time is wasted, and the requirement for N_{QST} is just the same as it is in the case of the traditional event-by event synchronization:

(c") N_{QST} must be large enough to collect the statistics of the output of the simulation with the required accuracy.

The proportion of the wasted $c_e'T_N$ and useful T_{QST} virtual time is very important. The $c_e'T_N$ virtual time is used up just to eliminate the transient caused by SSM-T. In addition to its execution time comes the execution time wasted due to communication overhead between the processors executing the segments. However, if both of them are low compared to the execution time of T_{QST} and the model of the simulated system was partitioned in the way that the load of the executing processors is nearly balanced, our simulation may produce a good speed-up.

V. EXAMPLES FOR THE APPLICATION OF SSM-T

A. Satellite Power Consumption - APositive Example

Let us consider the following example. A hurricane forecasting satellite collects data of the atmosphere and after some preprocessing, it sends them to the Earth for evaluation. The whole system is built up by three major functional units:

1. The Intelligent Measurement System controls the sensors and evaluates their signals, collects and preprocesses measurement data. Its output is a variable rate packet data flow. The packet rate depends on environmental conditions such as the state of the atmosphere, hurricane suspicious observations, etc.
2. The Data Transmission System carries the data packets from the Intelligent Measurement System

to the Data Evaluation System on the Earth. The Data Transmission System contains a radio link downwards and another one upwards. The data sent through the downlink is acknowledged on the uplink. The transmission power is controlled in the function of the packet loss ratio, so the power consumption/bit depends on the environmental conditions (state of the atmosphere, orbit deviations, etc) too. To save power, the carrier of the downlink is turned off when there is no transmission.

3. The Earth Data Evaluation System is responsible for the final evaluation of the collected measurement data.

Figure 4 shows the block diagram of the physical system.

Our task is to determine the behavior of the power consumption of the Satellite Radio System to be able to tell the solar cell and battery requirements. The power consumption depends on both the packet rate from the Intelligent Measurement System and the radio channel conditions.

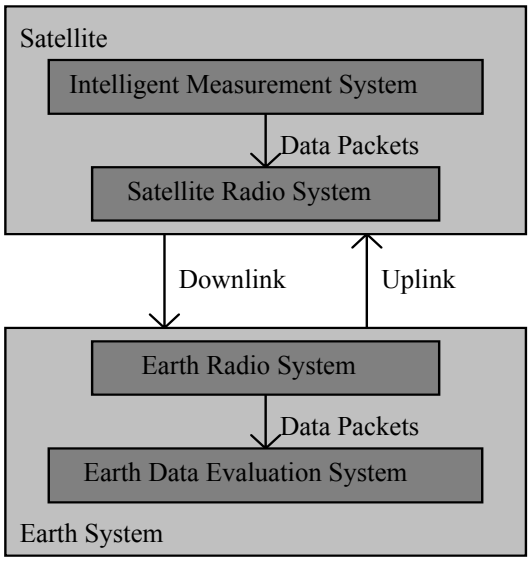


Figure 4: The satellite data collection system

Both quantities depend on environmental conditions that are too complex for an analytical treatment. Some of the environmental conditions (e.g. atmosphere) influence both the packet rate and the required transmission power, so they cannot be simulated independently. However, it is known from earlier experiments that the environmental conditions change very slowly compared to the data packet rate, that is typically millions of data packets are transmitted between two consecutive significant changes in the environmental conditions. It is also known that the channel capacity is more than twice as much than it is necessary for the maximum packet rate, so there is practically not buffering except that the packets are stored (to be able to retransmit them) until an acknowledgement is received. We propose the following parallel simulation:

- The Earth Data Evaluation System is omitted as it has no influence for the investigated power consumption.
- The remainder of the system is divided into two segments: the Intelligent Measurement System and the Data Transmission System, they are simulated parallel on two processors. The segments model those environmental conditions that are relevant to their operation.
- The two segments of the model are executed by two processors. SSM-T is applied between the two segments. The packet inter-arrival time statistics are collected by the OIF of the Intelligent Measurement System and the result is sent to the IIF of the Data Transmission System. The packet data flow is regenerated by the IIF of the Data Transmission System.

Figure 5 shows the block diagram of the simulation model.

The applicability criteria of the SSM-T are satisfied:

(a) At the boundaries of the two segments, not the individual packets but only the average packet rate is important in the point of view of the power consumption.

(b) A small error of the estimation of the packet rate (or the distribution of the packet inter-arrival time) causes a small error in the calculation of the power consumption and depends only on the

measure of the error, not the actual value of the packet rate.

(c) Significant changes are rare enough to make the necessary number of observations in quasi stationary state.

(d) As there is practically no buffering, the delay of the changes of the packet rate causes error only until the arrival of the new perfect statistics.

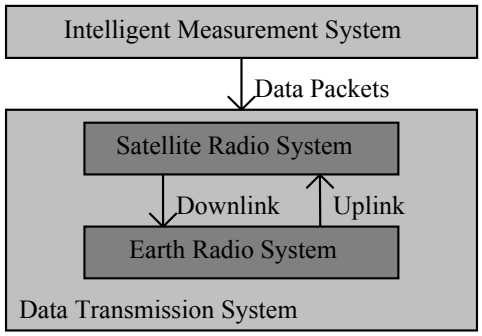


Figure 5: The simulation model of the satellite data collection system

B. Negative Examples

The first example is the simulation of an FDDI [1] ring. If the ring is divided into two (or more) segments and the segments are simulated by separate processors, the explicit passing of the token cannot be replaced by the arrival time statistics of the token collected at the segment boundaries. This would be a violation of the Media Access Control protocol resulting in ring recovery (token loss, duplicate tokens etc). Here SSM-T cannot be applied, because condition (a) is not satisfied. Our second negative example is derived from the before mentioned positive one. Let us modify the system described there in the following way:

The Data Transmission System contains two downlinks with no transmission power control. The unacknowledged packets are retransmitted until an acknowledgment arrives for the packet.

The primary downlink is always operational and the secondary works only if it is necessary due to high packet rate and poor channel conditions (that is, a high number of packet retransmissions). The earth station does not have burst demodulators, so when the second carrier is put into operation, there is a significant synchronization overhead. For this reason, the carrier is not switched off immediately after transmitting a packet, but only after a certain delay. This also means that even a few packets on the secondary link may result in significantly increased power consumption.

For any given channel conditions one can calculate what packet rate can completely exhaust the capacity of the primary downlink. An arbitrarily small error in the estimation of the packet rate from the Intelligent Measurement System can cause a serious error in the output of the simulation if the packet rate is close enough to the calculated critical rate. Now, condition (b) is not satisfied.

C. Two Interconnected FDDI Rings - Another Positive Example

In [5] SSM-T was used in the simulation of the FDDI backbone of the Technical University of Budapest. This network 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 two rings are interconnected by a router. The topology of the network and the cable lengths were taken from the real system. The load used in the simulation model came from measurements taken on the real FDDI rings.

First, a very accurate simulation of the network was performed. Then, SSM-T interfaces were inserted between the two FDDI rings and the simulation was performed with the same parameters as before.

The utilization of the rings were measured in both cases. It was found that the utilization values were close to each other in the two simulations. More detailed discussion of the experiments can be found in [5].

VI. CONDITIONS FOR A GOOD SPEEP-UP

In the practical application of SSM-T, the length of the virtual time interval while the OIF collects a statistics package is very important. There are three points:

1. The time interval should be large enough to collect a sample that is based on enough observations to produce an estimation with the required accuracy.
2. The time interval should be short enough to bound the length of the transient caused by SSM-T.
3. The frequency of the statistics exchange should not be too high to produce good speed up.

The first two conditions should be evident for the reader, but the third one requires some explanations. Until now, the simulation with SSM-T was independent from the execution environment. However, the main aim of SSM-T is to produce both good results and good speed-up. Thus, the overhead caused by the statistics exchange is very important. Let us consider this overhead.

The statistics transmission directed graph is defined as follows:

- The nodes are the segments of the simulation model,
- The (directed) edges are the segment to segment routes of statistics transmission.

If there are no directed loops in the graph then the simulation may work as a pipeline with infinite buffers between the stages. However, if there is a directed loop in the graph the virtual times of the segments of this loop are synchronized in some way. Let us consider the simplest example: there are two segments and they send statistics to each other. Let they exchange their statistics every T virtual time interval. The segments are executed by two processors A and B. The processors are identical and they do not have any other load. The execution time of the T virtual time is τ_A and τ_B. The time of communication is denoted by τ_C. This time includes the time of data packing and conversion to the network data format (e.g. XDR, if necessary) data transmission time and propagation delay plus data conversion from network format (if necessary) and unpacking. The execution time of a T virtual time interval with SSM-T is:

$$\tau_2=\max(\tau_A,\tau_B)+\tau_C \quad (1)$$

The overhead of the statistics collection and regeneration done by the IIF's and OIF's is denoted by τ_{IIF} τ_{OIF}. These are included in τ_A and τ_B, so they must be subtracted in the calculation of the execution time of the traditional simulation. The execution time of a T long virtual time interval using traditional uni-processor simulation is:

$$\tau_1=\tau_A-\tau_{OIF\text{-}A}-\tau_{IIF\text{-}A}+\tau_B-\tau_{OIF\text{-}A}-\tau_{IIF\text{-}A}+\tau_C \quad (2)$$

Let us group the I/O interfaces overhead into τ_{IF}.

$$\tau_{IF}=\tau_{OIF\text{-}A}+\tau_{IIF\text{-}A}+\tau_{OIF\text{-}A}+\tau_{IIF\text{-}A} \quad (3)$$

The speed-up is:

$$s=\frac{\tau_A+\tau_B-\tau_{IF}}{\max(\tau_A,\tau_B)+\tau_C} \quad (4)$$

This value can be close to 2 if $\tau_A \approx \tau_B$, $\tau_C << \tau_A$ and $\tau_{IF} << \tau_A$, that is the load of the processors is well balanced, the communication overhead, and the overhead caused by the statistics collection and message regeneration are small.

In a large (communication) system there are usually multiple points where the SSM-T applicability criteria are satisfied. It means that the insertion of the SSM-T interfaces to these points will not cause significant degradation of the accuracy of the results. Out of these points, the simulationist must carefully select those, that separate the model to segments of similar complexity, so that the computing powers required by the simulation of the segments are in the same order.

When selecting the statistics collection method it is worth considering its algorithmic complexity [6]. An interesting new density estimation method, k-split [17] may also be used in SSM OIF's. And last but not least the frequency of the statistics exchange should not be higher than it is required. In earlier experiments on simulating two interconnected FDDI rings by two

processors using SSM-T [5], we achieved 1.75, 1.86 and 1.91 speed-up depending on the frequency of statistics exchange.

The existence of weaker criteria for the applicability of SSM-T is also a question of interest.

VII. CONCLUSIONS

The applicability of the modified Statistical Synchronization Method (SSM-T) was studied. Criteria were given for the applicability of SSM-T in parallel discrete event simulation. We have proven that the small level of output error of the parallel simulation using SSM-T compared to the uni-processor simulation without SSM-T can be guaranteed if the criteria are satisfied.

We showed a real life example where the applicability criteria are satisfied and SSM-T can be applied. We gave negative examples too. We presented the conditions when the application of SSM-T results in a good speed-up.

The results confirm that SSM-T is a very efficient synchronization method in the parallel simulation of the communication networks. One interesting step of the current work is to combine the Traffic Flow Analysis (TFA) [9] and the detailed event-by-event simulation [11] for the parallel execution of the combined system [12, 14, 15].

REFERENCES

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

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

[3] D. Jefferson et al., *"Distributed Simulation and the Time Warp Operating System."* Proceedings of the 12th SIGOPS - Symposium on Operating System Principles, 1987. pp. 73-93.

[4] G. Lencse, *"Efficient Simulation of Large Systems - Transient Behaviour and Accuracy"* Proceedings of the 1997 European Simulation Symposium (ESS'97) (Passau, Germany, Oct. 19-23, 1997.). SCS Europe, 660-665.

[5] G. Lencse, *"Efficient Parallel Simulation with the Statistical Synchronization Method"* Proceedings of the Communication Networks and Distributed Systems Modeling and Simulation (CNDS'98) (San Diego, CA. Jan. 11-14, 1998.). SCS International, 3-8.

[6] G. Lencse, *"Statistics Collection for the Statistical Synchronisation Method"* Proceedings of the 1998 European Simulation Symposium (ESS'98) (Nottingham, UK. Oct. 26-28, 1998.). SCS Europe, 46-51.

[7] G. Lencse, *"Applicability Criteria of the Statistical Synchronization Method"* Proceedings of Communication Networks and Distributed Systems Conference (CNDS'99), (San Francisco, CA, USA, January 17-20, 1999.) SCS, 159-164

[8] G. Lencse, *"Design Criterion for the Statistics Exchange Control Algorithm used in the Statistical Synchronization Method"* Proceedings of the 32nd Annual Simulation Symposium (San Diego, CA, USA, April 11-15, 1999.) IEEE Computer Society, 138-144

[9] G. Lencse, *"Traffic-Flow Analysis for Fast Performance Estimation of Communication Systems"* Journal of Computing and Information Technology Vol. 9, No. 1, (2001.) 15-27.

[10] G. Lencse, *"Parallel Simulation with OMNeT++ using the Statistical Synchronization Method"* Proceedings of the 2nd International OMNeT++ Workshop (Technical University Berlin, Berlin, Germany, Jan. 8-9, 2002.) 24-32.

[11] G. Lencse, *"Combination and Interworking of Traffic-Flow Analysis and Event-Driven Discrete Event Simulation"* Proceedings of the 2004 European Simulation and Modelling Conference (ESM'2004) (Paris, France, Oct. 25-27, 2004.) EUROSIS-ETI, 89-93.

[12] G. Lencse, *"Speeding up the Performance Analysis of Communication Systems"* Proceedings of the 2005 European Simulation and Modelling Conference (ESM'2005) (Porto, Portugal, Oct. 24-26, 2005.) EUROSIS-ETI, 329-333.

[13] G. Lencse and L. Muka, *"Convergence of the Key Algorithm of Traffic-Flow Analysis"* Journal of Computing and Information Technology, Vol. 14, No. 2, (June 2006.) pp. 133-140.

[14] G. Lencse and L. Muka, *"Combination and Interworking of Four Modelling Methods for Infocommunications and Business Process Systems"* Proceedings of the 2007 Industrial Simulation Conference (ISC'2007) (Delft, The Netherlands, June 11-13, 2007.) EUROSIS-ETI, 350-354.

[15] L. Muka and G. Lencse, *"Decision Support Method for Efficient Sequential and Parallel Simulation: Time Decomposition in Modified Conceptual Models"* Proceedings of the 2007 European Simulation and Modelling Conference (ESM'2007) (Malta, Oct. 22-24, 2007.) EUROSIS-ETI, 574-581.

[16] Gy. Pongor, *"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, 1992.) SCS Europe, 125-129.

[17] A. Varga, *"K-split – On-Line Density Estimation for Simulation Result Collection"*. Proceedings of the European Simulation Symposium (Nottingham, UK. Oct. 26-28, 1998.) SCS Europe, 41-45.

[18] Y. A. Sekercioglu, A. Varga, and G.K. Egan, (2003), *"Parallel simulation made easy with OMNeT++"* Proceedings of the European Simulation Symposium (ESS'2003), (Delft, The Netherlands, Oct. 2003.) SCS, 2003.

[19] M. Kozlovszky, Á. Balaskó and A. Varga, *"Enabling OMNeT++-based simulations on Grid Systems"*. Proceedings of the 2nd International Workshop on OMNeT++ (OMNeT++ 2009), (Rome, Italy, Mar. 2009.) ICST, 2009.

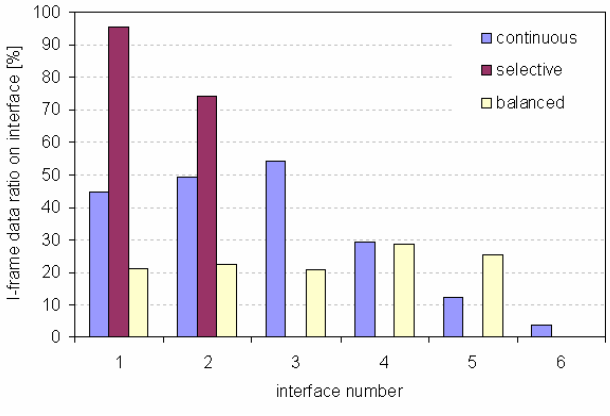
[20] http://www.omnetpp.org

[21] http://www.omnest.com.

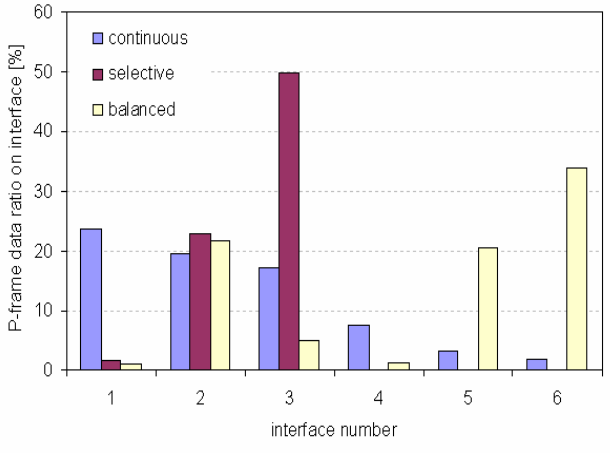
AUTHOR BIOGRAPHIES

Gábor Lencse received his M.Sc. in electrical engineering and computer systems at the Technical University of Budapest in 1994 and his Ph.D. in 2001. The area of his research is (parallel) discrete event simulation methodology. He is interested in the acceleration of the simulation of info-communication systems. Since 1997, he has been working for the Széchenyi István University in Győr. He teaches computer networks and networking protocols. Now, he is an Associate Professor. He is a founding member of the Multidisciplinary Doctoral School of Engineering, Modelling and Development of Infrastructural Systems at the Széchenyi István University. He does R&D in the field of the simulation of communication systems for the Elassys Consulting Ltd. since 1998. Dr Lencse has been working part time at the Budapest University of Technology and Economics (the former Technical University of Budapest) since 2005. There he teaches computer architectures.

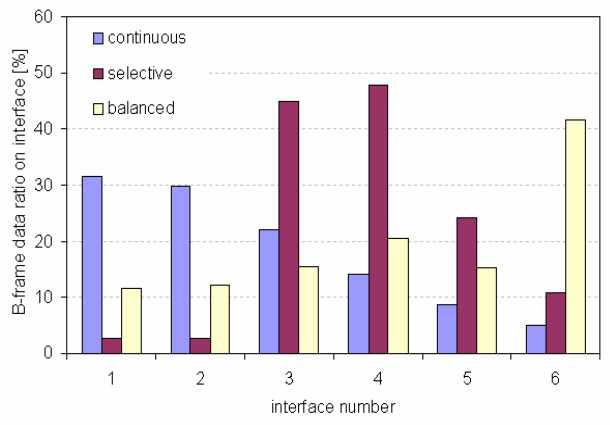
I–frame data ratio of the MPEG–4 video stream is P_I=51%, hence the I–frame data bandwidth requirement is about 150kbps.



a) I–frame distribution on the interfaces



b) P–frame distribution on the interfaces



c) B–frame distribution on the interfaces

Figure 5: Different frame type distributions in the interface buffers

As the results show, I–frame data can be delivered on the first two interfaces (100kbps+80kbps) when the proposed content–based selective packet distributor algorithm is used (Figure 5. a). Most of the P–frames were delivered on the 3th interface, while the B–frames were transferred on the less reliable channels. Due to this ordered packet delivery, the number of lost packets belonging to I–frames is estimated to be significantly lower then the number of other frame packet losses.

The observed video quality highly depends on the number of lost packets and the packet content. In order to analyze the behavior of the proposed multipath delivery method we have examined the packet loss of different frame types . The overall packet loss ratio was also measured, which is the average of the I–, P– and B–frame packet loss ratios. The results are presented in 0Similarly to the previous measurements the heap up buffer size was set to 80 frames.

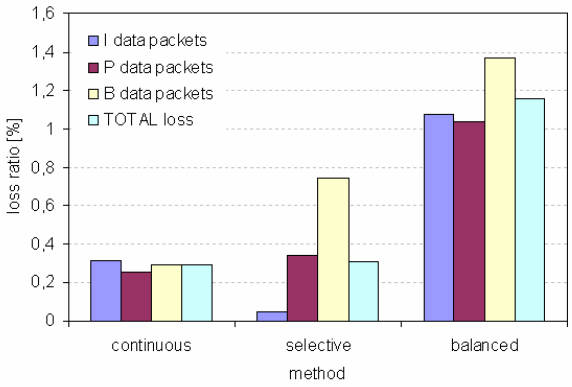


Figure 6: Total and I–, P–, B–frame type data packet loss ratios

As we expected, the total number of packet losses was equal using the continuous packet distributor method and the selective one. In spite of the similarity, the number of I–frame data packet losses was significantly lower when the proposed interface selection algorithm was used. In case of the continuous algorithm, the loss probabilities of different types of frame data were nearly identical. However, by protecting the I–frame data, the error