Design Criterion for the Statistics Exchange Control Algorithms Used in the Statistical Synchronization Method

Gábor Lencse

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

Department of Telecommunications
Széchenyi István College
Hédervári u. 3.
H-9026 Győr, Hungary
E-mail: lencse@szif.hu
http://rs1.szif.hu/~lencse

Abstract

The conditions of the statistics exchange of the Statistical Synchronization Method (SSM) are studied.

A statistics exchange control algorithm is proposed that is based on prediction and synchronization point deletion. The so-called penalty functions are introduced. They are used to give a mathematical criterion that can be a measure of the goodness of the different statistics exchange control algorithms.

Both analytical treatment and simulation show that there is a trade-off between the accuracy of the results and the achievable speed-up.

According to the simulation results, we can get near-optimal results for a relatively wide range values of the parameters of the prediction, thus SSM-T is robust enough to tolerate some inaccuracy of the parameters of the prediction.

1: Introduction

The Statistical Synchronization Method (SSM) [8] is a promising alternative to the conventional synchronization methods for parallel discrete event simulation (e.g., conservative, optimistic) [2].

The conventional synchronization methods use event-by-event synchronization between the segments of the simulated system and they are unfortunately not applicable to all cases, or do not provide the desirable speed-up. 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.

SSM does not exchange individual messages between the segments but rather the statistical characteristics of the message flow. Actual messages are regenerated from the statistics at the receiving side. (Further explanation will be given later.) SSM claims to be less sensitive to communication delay and it requires less network bandwidth than event-by-event methods. Nevertheless, it is not accurate in the sense that an event that occurred in one segment of the system does not have an immediate influence on another segment. For this reason, the method cannot be applied in some simulations, for example in the case of digital circuits but remains feasible in other classes of simulation such as the performance estimation of communication systems.

The transient behavior and accuracy of SSM applied to an FDDI simulation were already demonstrated in [4], and the questions of parallelisation using SSM-T were discussed in [5]. The latter paper shows that a very good speed-up can be achieved in a PDES (parallel discrete-event simulation) applying SSM-T and loose synchronization between the segments of the simulated system, because the processors executing the segments run independent in the vast majority of time.

In [6], we have examined what statistics collection methods should be used for SSM to be able to faithfully regenerate the statistical characteristics of the message flow. We have determined the L_1 error of some frequently used statistics collection methods in the function of the collected observations from some well-known distributions. Formal criteria are given for the applicability of SSM in [7].

The aim of this paper is to determine under what conditions the statistics should be exchanged and the statistics collection be restarted. We are faced with the following dilemma: To achieve a desired accuracy of the statistics, one needs to have a certain number of observations. E.g. the L_1 error of the approximation of the distribution with the collected statistics is small enough when the number of observations is 1000. In this way, the statistics exchange should be triggered by the number of observations collected. However, to achieve a good speed-up, the source segment needs to know and tell the destination segment in advance that at what virtual time the statistics will be sent so that the simulation of the segments may run independently until that time. That is, the statistics exchange should be triggered by the virtual time. To resolve this dilemma a method with prediction and synchronization point deletion is proposed. We introduce the penalty functions to give a mathematical criterion that can be a measure of the goodness of the different statistics exchange control algorithms. We conclude that using this prediction there is a trade-off between the accuracy of the results and the speed-up of the parallel simulation.

The remainder of this paper is organized as follows: first, a brief introduction to SSM and SSM-T is given, then the conditions of the statistics exchange are considered introducing the statistics exchange control method with prediction and synchronization point deletion and the penalty functions, next an analytical treatment is carried out, finally the recommended solution is tested by simulation in special cases.

This topic was identified as being of importance in the parallel simulation of large (communication) systems.

2: The statistical synchronization method

2.1: The original SSM

For those not familiar with SSM, a short summary of the Statistical Synchronization Method is given here. See [8] for more information.

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

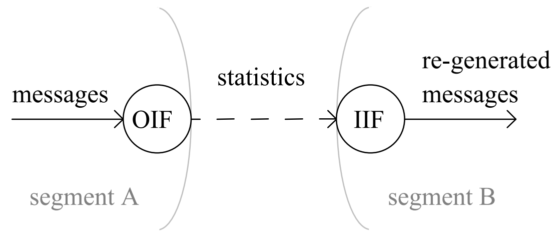

Figure 1. An OIF - IIF pair

2.2: 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] 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 Figure 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 Figure 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 of segment B. 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 $t \le t_2$ 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*

synchronization. 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. See [5] and [7] for more details.

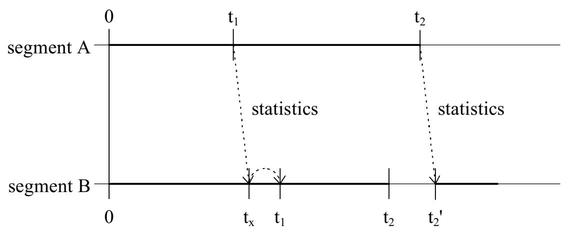

Figure 2. The operation of SSM-T. See the text for explanation

3: When to exchange the statistics?

In the first paper about SSM-T [5], the segments exchanged their statistics in the way that at the same virtual time when the OIF of segment A sent the collected statistics to the IIF of segment B, the OIF of segment B also sent the collected statistics to the IIF of segment A. However, the data flow between two segments is not necessarily bi-directional, or even if it is bi-directional, the optimal virtual time of the statistics sending may be different in the case of the two direction. For this reason, now we consider only one direction: segment A sends the collected statistics to segment B.

3.1: Condition associated with loose synchronization

Segment A must tell segment B at what t_i virtual time the i-th statistics will be sent. [5] Until t_i, segment B can be simulated independently from segment A. At t_i, segment B either has to wait for the arrival of the statistics from segment A, or the statistics already arrived. This depends on the speed of the simulation of the segments and the speed of the communication channel between the two simulation processes. Anyway, the statistics take effect at t_i virtual time.

3.2: Condition to ensure accuracy

To achieve the desired accuracy, that is the L_1 error of the statistics is less then a desired limit, the collected statistics must be based on at least N number of observations, where N depends on the statistics collection method, the distribution of the observations and the required level of the L_1 error. [6]

3.3: The solution proposed

The relationship between the N number of messages and the $\Delta t_i=t_i-t_{i-1}$ time they are observed is the *message rate in the i-th interval*: $R_i=N/\Delta t_i$. R_i is not known in t_{i-1} yet, it can only be predicted on the basis of the past R_j, $j<i$ values. The simplest *prediction* is $R_i=R_{i-1}$, which is good if the message rate changes slowly. Of course, the prediction lacks data at the beginning of the simulation unless they are provided on the basis of a-priori knowledge (e.g. previous simulation, analytical results, etc.).

If the number of the collected observations is less then N, then the accuracy of the collected statistics is not satisfactory, so we cannot guarantee the desired accuracy of the results of the simulation. If the number of the collected observations is significantly greater then N, then the transmission of the statistics was unnecessarily delayed until the given virtual time, and transient caused by the OIF-IIF pair is longer then it would have been in the optimal case. This is also against the accuracy of the results.

To ensure the accuracy of the results let us introduce the following method: Segment A tells segment B an earlier point of virtual time to expect statistics than what would come from the prediction. If the statistics are "ready" at this earlier point of time, then they are sent, otherwise segment B is notified that the simulation of segment B can proceed until another predicted point of virtual time, that is *the current synchronization point of virtual time is deleted*. This is continued until the statistics have the desired accuracy and can be sent to segment B. In this way the accuracy is ensured. The number of the deleted synchronization points of virtual time must be kept low, otherwise the speed-up of the parallel simulation will not be satisfactory.

In this way, there are three kinds of deviations from the optimal case:

1. too many synchronization points of time were deleted
2. the statistics are based on too few observations
3. the transient is too long (too many observations)

They all have negative consequences for the results or the execution time of the simulation.

3.4: The penalty functions

To be able to develop or judge a *statistics exchange control algorithm* that tells us when to send the statistics, create or delete a synchronization point of virtual time, let us introduce the *penalty functions*. They are to express how bad we consider the before mentioned three kinds of deviations from the optimal. They show the penalty of the deviation in the function of the number of observations (n) in the sample.

The overhead caused by the synchronization points is proportional to the number of them, that is inversely proportional to the number of observations (n) in the sample. The penalty function is:

(1) $P_{SP}(n) = c_1 / n, \quad n > 0$

The L_1 error in the function of n is proportional to $n^{-1/3}$ if the statistics collection is done with histogram. To determine the penalty function, one should know the function that tells the error of the output of the simulation in the function of the L_1 error of the statistics exchanged. Of course, this function can not be given in general, only in the knowledge of the particular simulation. If one expects the L_1 error to be always less then a predefined value and does not care about its exact value, a very simple penalty function is:

(2) $P_{L_1}(n) = \begin{cases} c_2, & 0 < n < N \\ 0, & n \geq N \end{cases}$

Similarly, the error caused by the transient and so the penalty depends on the given system we simulate. A logical choice is the following: If n is just a bit greater than N then there is no much penalty, but significantly larger values deserve increasing penalty. An example penalty function is:

(3) $P_{TR}(n) = \begin{cases} 0, & 0 < n < cN \\ c_3 n + c_4, & n \geq cN \end{cases}$

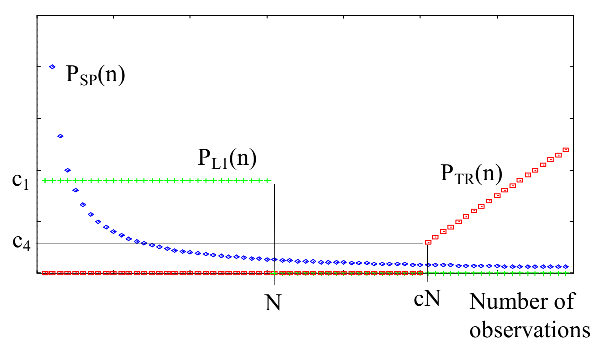

Figure 3. Example penalty functions

The *overall penalty function* is the sum of the three penalty functions. Of course this 'sum' is equivalent to a 'linear combination', as the penalty functions already have factors of freedom to be given different weights. The sum was found better then the product of the three penalty functions, because in the case of the sum we can tune each penalty function independently from the others; extremely small or zero values are also allowed.

(4) $P_{\Sigma}(n) = P_{SP}(n) + P_{L_1}(n) + P_{TR}(n)$

4: Design criterion for the statistics exchange control algorithm

An optimal statistics exchange control algorithm should minimize the expected penalty. It can be calculated in the following way: Let us suppose that the inter-arrival time of the observations (that is packets) is exponential with parameter λ. The number of packets arrive to the OIF in a [0,T] time interval can be described by a Poisson process with intensity λ.

(5) $p_T(n) = \frac{(\lambda T)^n}{n!} e^{-\lambda T}$

In the simplest case, when the synchronization points are not deleted but the collected statistics are sent at all synchronization points of virtual time, the task of the statistics exchange control algorithm is to choose T so that the expected value of the P(T) penalty be minimal.

(6) $E\{P(T)\} = \sum_{n=1}^{\infty} p_T(n) P_{\Sigma}(n)$

If we allow the deletion of the synchronization points, then the *overall penalty function* must be modified:

(7) $P_{k\Sigma}(n, n') = P_{SP}(n) + I_k P_{L_1}(n') + I_k P_{TR}(n')$,

where I_k is 1 if the statistics are really sent at the k-th synchronization point and it is 0 otherwise; n is the number of observations from the last (maybe deleted) synchronization point and n' is the cumulated number of observations from the last statistics sending.

It would be nice to find K and $T_1, T_2, \ldots, T_k, \ldots, T_K$, that minimize the expected value of the $P(K, \ldots, T_k, \ldots)$ penalty.

(8) $E\{P(K, \ldots T_k \ldots)\} = \sum_{k=1}^{K} \sum_{n=1}^{\infty} p_{T_k}(n) P_{SP}(n) +$

$+ \sum_{n'=1}^{\infty} p_T(n')(P_{L_1}(n') + P_{TR}(n'))$, where $T = \sum_{k=1}^{K} T_k$

At the beginning of this chapter we supposed exponential inter-arrival time. If it is not true the Poissonian process is still a good approximation for large N values. The typical values of N in [6] are 1000 and above - they are large enough. The λ parameter should be set according to the following:

(9) $\lambda = \frac{1}{\text{average inter-arrival time}}$

Let us use the penalty functions defined by (1) and (2). For analytical simplicity, we use

$$(3')\quad P_{TR}(n) = c_3 n^2$$

The expected value of the speed-up penalty in a single T_k interval with n observations is calculated according to the following:

$$(10)\quad \sum_{n=1}^{\infty} p_{T_k}(n) P_{SP}(n) = \sum_{n=1}^{\infty} \frac{(\lambda T_k)^n}{n!} e^{-\lambda T} \frac{c_1}{n} = = c_1 \mathrm{E}\left\{ \frac{1}{U} I_{\{U \geq 1\}} \right\} \leq c_1 \left(\frac{1}{\lambda T_k} + \frac{3}{(\lambda T_k)^2} \right) = c_1 \frac{\lambda T_k + 3}{(\lambda T_k)^2}$$

The inequality is proven in [1].

The L_1 error penalty depends on the whole T interval and the n' cumulated number of observations:

$$(11)\quad \sum_{n'=1}^{\infty} p_T(n') P_{L1}(n') = \sum_{n'=1}^{\infty} \frac{(\lambda T)^{n'}}{n'!} e^{-\lambda T} c_2 I_{n'<N} = = c_2 \sum_{n'=1}^{N-1} \frac{(\lambda T)^{n'}}{n'!} e^{-\lambda T}$$

And the transient penalty is:

$$(12)\quad \sum_{n'=1}^{\infty} p_T(n') P_{TR}(n') = \sum_{n'=1}^{\infty} \frac{(\lambda T)^{n'}}{n'!} e^{-\lambda T} c_3 n'^2 = c_3 \mathrm{E}\left\{ U^2 \right\} = = c_3 \left((\lambda T)^2 + \lambda T \right)$$

The expected value of the overall penalty is:

$$(8')\quad E\left\{ P(K, \ldots T_k \ldots) \right\} = c_1 \sum_{k=1}^{K} \frac{\lambda T_k + 3}{(\lambda T_k)^2} + c_2 \sum_{n'=1}^{N-1} \frac{(\lambda T)^{n'}}{n'!} e^{-\lambda T} + + c_3 \left((\lambda T)^2 + \lambda T \right), \text{ where } T = \sum_{k=1}^{K} T_k$$

This equation deserves some discussion. The choice of the c_1, c_2 and c_3 constants is a compromise between the guaranteed level of the accuracy of the results and the speed-up of the parallel simulation. By choosing $c_2 \to \infty$ one can assure that the criteria for the L_1 error is always satisfied. However, in this case we must give up one factor of freedom: K. That is the statistics can be sent only if they are based on enough observations, otherwise the synchronization point must be deleted and the statistics collection must be continued. By doing so we can ensure the level of the L_1 error but it has a negative side effect: this change in the algorithm influences the expected value of the remaining two components (speed-up penalty and transient penalty) of the overall penalty function. Thus, in this case we can not use the analytical results. We present simulation results for this case in the next section.

5: Simulation results for special cases

As we wrote in the previous section, now we do not allow the L_1 error to be greater than a required value, that is we always use statistics of not less than the required N number of observations.

Let us suppose, that our prediction tells us that the n' number of observations will reach N in T time. How shall we choose the T_i lengths of the time intervals to minimize the penalty? Let us choose some numerical values: N=1000, λ=1, T=1000, c_1=10, $c_3=10^{-6}$. We test two methods for calculating the virtual time of the synchronization points:

a) $T_i = i\alpha T, \quad i \geq 1$

b) $T_i = \begin{cases} \beta \frac{T}{2}, & i = 1 \\ T_{i-1} + \beta \frac{T}{2^i}, & i \geq 2 \end{cases}$

In both cases the statistics collection must be continued until $n' \geq N$. Simulation experiments were executed for different values of α and β. The inter-arrival time of the consecutive observations (messages) was drawn according to exponential distribution with λ parameter. At the i-th synchronization point (T_i) the statistics were considered to be 'ready' if the n'>N was satisfied. Otherwise the synchronization point was considered to be deleted and the statistics collection (that is, the counting the messages) was continued. Table 1 and Table 2 show the results. All the values are based on 10000 experiments.

The 0.35 value seems to be the best choice for α, however the standard deviation of the transient penalty is so high that we can say only that a value between 0.15 and 0.60 seems to be a good choice for α. We can observe, that for α>0.25 the transient penalty values in the function of α show fluctuation. The explanation is the following: The T_i values are comparable with T. Let n' reach N at the K-th synchronization point. K is about

$$K = \left\lceil \frac{T}{T_i} \right\rceil = \left\lceil \frac{T}{\alpha T} \right\rceil = \left\lceil \frac{1}{\alpha} \right\rceil$$

The transient penalty is proportional with $(K\alpha)^2$. E. g. for α=0.40 we get K=3 and the transient penalty is 1.2^2=1.44. Of course, 1 is a lower bound for the transient penalty.

For β, the values between 1.25 and 1.5 seem to be good, however in this case the deviation of the speed-up penalty causes uncertainty. We must be careful with the selection of the value of β, because T_i values converge to βT. If β is very close to 1 then it may result in a very large number of (deleted) synchronization points and so a huge speed-up penalty.

We are not concerned with the actual optimal values of α and β, but the point is the following: depending of the simulated system, the c_i constants and the method used for determining the T_i values one can find the optimal case, but this depends on the c_i values, that is we must balance between the guaranteed level of the accuracy of the results and the achievable speed-up.

Another conclusion is, that we can get near-optimal results for a relatively wide range values of parameters α and β, thus SSM-T is robust enough to tolerate some inaccuracy in the selection of them.

α	speed-up penalty		transient penalty		total
	average	σ	average	σ	penalty
0.05	4.1838	0.2720	1.0507	0.0315	5.2345
0.10	1.0599	0.0796	1.1026	0.0675	2.1624
0.15	0.4733	0.0262	1.1228	0.0728	1.5961
0.20	0.2760	0.0320	1.2138	0.1716	1.4898
0.25	0.1807	0.0244	1.2755	0.2309	1.4562
0.30	0.1338	0.0041	1.4400	0.0841	1.5738
0.35	0.0875	0.0083	1.1510	0.1686	1.2385
0.40	0.0752	0.0022	1.4411	0.0848	1.5163
0.50	0.0499	0.0111	1.6003	0.5662	1.6502
0.60	0.0334	0.0010	1.4426	0.0839	1.4759
0.70	0.0286	0.0008	1.9618	0.1057	1.9905
0.80	0.0250	0.0006	2.5615	0.1288	2.5866
0.90	0.0222	0.0006	3.2410	0.1610	3.2632
1.00	0.0150	0.0053	2.4818	1.4280	2.4968

Table 1. The penalty values in the case when the synchronization points are generated according to method "a".

β	speed-up penalty		transient penalty		total
	average	σ	average	σ	penalty
1.10	0.3453	0.7768	1.0591	0.0390	1.4044
1.15	0.1825	0.0790	1.0747	0.0467	1.2572
1.20	0.1251	0.0343	1.1136	0.0598	1.2387
1.50	0.0401	0.0016	1.2672	0.0754	1.3073
2.00	0.0201	0.0103	1.6186	0.5668	1.6387

Table 2. The penalty values in the case when the synchronization points are generated according to method "b".

6: Conclusion

The conditions of the statistics exchange of the Statistical Synchronization Method (SSM) were studied. A statistics exchange control algorithm with prediction and synchronization point deletion was proposed.

We introduced the penalty functions to give a mathematical criterion that can be a measure of the goodness of the different statistics exchange control algorithms.

We carried out both an analytical survey for the general case and simulation experiments for special cases with two kinds of methods for synchronization point calculation.

We conclude that using the proposed statistics exchange control algorithm there is a trade-off between the guaranteed level of the accuracy of the results and the speed-up of the parallel simulation.

Simulation results showed, that we can get near-optimal results for a relatively wide range values of the parameters of the prediction, thus SSM-T is robust enough to tolerate some inaccuracy of the parameters of the prediction.

7: Acknowledgement

The OMNeT++ discrete-event simulator was used for the experiments. See its home page for more information [9].

8: References

[1] Beirlant, J.; L. Győrfi. (1998) "On the Asymptotic Normality of the L_2-Error in Partitioning Regression Estimation" *Journal of Statistical Planning and Inference*, 71, pp. 93-107.

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

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

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

[5] Lencse, G. (1998) "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). SCS International, pp. 3-8.

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

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

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

[9] Varga, A. (1998) "OMNeT++ Discrete Event Simulation System" http://www.hit.bme.hu/phd/vargaa/omnetpp.htm

9: Biography

Gábor Lencse was born in Gyôr, Hungary, in 1970. He received his M.S. in electrical engineering and computer systems from the Technical University of Budapest in 1994. He is currently pursuing his Ph. D. at the same university. The area of his research is computer architectures and parallel processing. He is interested in (parallel) discrete event simulation. Since 1997, he works for the Széchenyi István College in Gyôr. He teaches computer networks and networking protocols. He is a member of the Society for Computer Simulation International, IEEE Hungary Section, IEEE Computer Society and IEEE Communications Society.