SPEEDING UP THE PERFORMANCE ANALYSIS OF COMMUNICATION SYSTEMS

Gábor Lencse
Department of Telecommunications
Széchenyi István University
Egyetem tér 1.
H-9026 Győr, Hungary
E-mail: lencse@sze.hu

KEYWORDS

traffic-flow analysis, discrete event simulation, communication networks, performance analysis, parallel simulation

ABSTRACT

The methods for the performance analysis of communication networks are reviewed. Different ways are shown how the combination of DES and TFA can be made faster by using parallelism. An efficient algorithm for the parallel execution of the combined DES and TFA is presented.

INTRODUCTION

Because of the rapid development and growth of communication networks in the last decades, the analysis of their performance is an important issue. Event-driven ***discrete event simulation*** (DES) is a powerful tool for this task. Nevertheless, the systems are often so large and complex and the number of events is so high that the execution requires unacceptably long time even using a supercomputer. Let us review first, what solutions were proposed for this problem so far.

Parallelisation can be a natural solution. But it is not an easy task and the achievable speed-up is often limited. The reason is the algorithm of the event-driven DES. When doing ***parallel discrete event simulation*** (PDES), the model of the system is divided into segments, and the segments are assigned to processors that are executing them. To maintain causality, the virtual times of the segments must be synchronised. There are 3 different methods. The first two are described in (Fujimoto 1990). The ***conservative method*** ensures that causality is never violated. An event can be executed only if we know it for sure, that no events with smaller timestamp exist (and also will not be generated) anywhere in the model. Unless the simulated system has a special property that the so called *look-ahead* is large enough, the processors executing the segments need to wait for each other in the majority of time, so the achievable speed-up is poor. The ***optimistic method*** allows and detects causality errors and uses *roll-backs* to recover from them. The resource consumption of the roll-backs may hinder the good speed-up. The third one is the ***statistical synchronisation method*** (SSM) proposed by György Pongor. (Pongor 1992) This one does not exchange individual messages but rather the statistical characteristics of the message flow between the segments. In its original form, SSM was applicable for the analysis of steady state behaviour of systems. It was further developed (as SSM-T) by Gábor Lencse (Lencse 1998). The method can produce excellent speed-up, but has a limited area of application (Lencse 1999). For more information about the three methods, see (Lencse 2002) and its references.

The ***traffic-flow analysis*** (TFA) (Lencse 2001) is a different approach for the performance analysis of communication networks. Unlike discrete event simulation, TFA does not model the travelling of each packet through the network individually, but it rather uses statistics to model the networking load of applications. The method distributes the traffic (the statistics) in the network first, and calculates the specific traffic conditions for each line and switching node in the second step. The results are approximate but may characterize the traffic conditions of the network satisfactorily.

The ***combination of DES and TFA*** (Lencse 2004) is another promising idea. The combined method can be applied especially well for the performance analysis of the critical parts of communication networks. DES should be used for the precise analysis of the critical part only, and TFA is to be applied for the rest of the network. This solution has the following justification: the critical part is modelled accurately enough, but the computing power is not wasted for the execution of large number of events that individually are irrelevant for us, only their certain statistical consequences influence the behaviour of the critical part of the network.

Is it possible to speed up the combined TFA and DES by using parallelism? Is it worth doing so? How can it be done so that we gain satisfactory information of the traffic conditions of the network examined, in the shortest possible time? What can be done in parallel? What are the applicability criteria? How many processors should we use? The recent paper deals with similar questions.

WHAT TO DO IN PARALLEL?

There are a number of possibilities where we can put parallelism into our model. For the first sight there seem to be three main cases:

1. Execute the DES and the TFA segments parallel
2. Execute multiple DES segments parallel
3. Execute multiple TFA segments parallel

During our studies we shall refine these cases and consider their possible combinations, too.

Considering the parallel execution of the DES and the TFA segments

To examine the first case, let us recall the combination of DES and TFA (Lencse 2004). Between the DES segment and the TFA segment there is a bidirectional conversion between their traffic representations (messages and statistics) so that the traffic flow between the two segments can be modelled. In the aforementioned paper, we have shown three different possible ways for the implementation. We recommended the one where TFA is a set of functions within the DES program and it is called sometimes, and it uses the virtual time of the DES engine (and some of its services) for its internal purposes. In this way we can say that the whole thing is nothing else, but discrete event simulation. If we examine the time stamps of the events executed in this simulation, we see that they are growing while the DES part is executed, and are equal while the TFA part is executed and growing again while the DES part runs, etc. It is so because TFA is done at a given point of virtual time, and later at another given point of virtual time, and so on… In this way, if we want to execute the DES part and the TFA part by two processors parallel, we shall notice that always only one of the processors is working and the other is just waiting. This is a good negative example, how the parallel execution will result in no speed-up at all. It is not surprising, as the combination of DES and TFA was designed for sequential execution by a single processor. Now, we shall redesign it!

Designing the parallel execution of the DES segment and the TFA segment for two processors

Let us recall the original example application of combination of DES and TFA, to see what can be changed in the implementation. We have an X.25 network servicing ATM and POS terminals, and we would like to check what happens if an important link to the server fails. The basic idea of the suggested solution was the following: DES should be used for the precise analysis of the critical part only, and TFA is to be applied for the rest of the network. The common characteristic feature of these types of problems is that there is a critical part of the network (like the immediate neighbourhood of the server in the example above) that should be modelled accurately and there is all the rest of the network that cannot be omitted <u>because it gives the load for the critical part</u>. This observation is the key for the parallelisation. In the original model, bidirectional data flow was allowed between the DES segment and the TFA segment. It is general enough but not always necessary. If only the TFA part gives the load for the DES part and the traffic from the DES part to the TFA part is negligible, then the parallel implementation becomes very easy. (We shall see soon that the applicability criteria will not be so strict.)

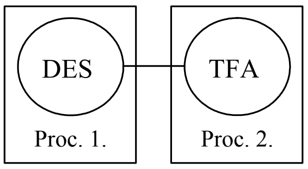


Figure 1. The execution of the combined DES and TFA by two processors

The parallel simulation works as follows:

- At t_0 the DES segment sends a message to the TFA segment that contains the timestamp t_1 and the parameters for the TFA segment that are valid at t_1 (and may also contain information about the traffic from DES segment to the TFA segment at t_1 – but it can only be a prediction based on the information available at t_0 virtual time)
- From t_0 to t_1 the two segments run independently.
- At t_1 the TFA segment sends a message to the DES segment containing the traffic information from the TFA segment to the DES segment, and then the DES segment sends a message to the TFA segment containing a new timestamp t_2, the parameters for the TFA segment valid at t_2 and possibly the approximation of the DES$\rightarrow$TFA traffic at t_2 as predictable at t_1.
- From t_1 to t_2 the two segments run independently.

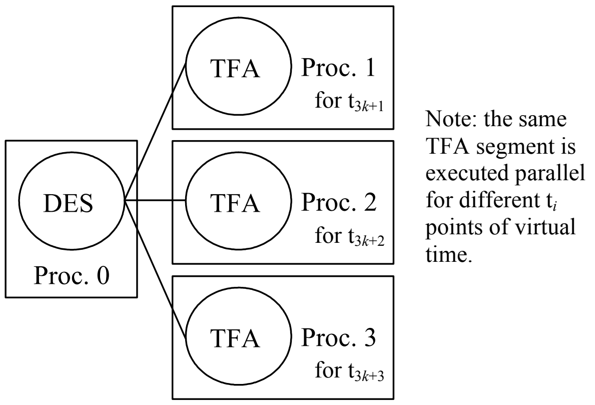
In the general step:

- At t_i the TFA segment sends a message to the DES segment containing the traffic information from the TFA segment to the DES segment, and then the DES segment sends a message to the TFA segment containing a new timestamp t_{i+1}, the parameters for the TFA segment valid at t_{i+1} and possibly the approximation of the DES$\rightarrow$TFA traffic at t_{i+1} as predictable at t_i.
- From t_i to t_{i+1} the two segments run independently.

This algorithm makes it possible, that the virtual times of the two segments must meet only at the t_0, t_1, t_2, t_i, … synchronisation points of virtual time and the processors may work independently in any other time. This is the same freedom that SSM-T offers. Recall, how good results we could achieve with that! (Lencse 1998) Of course the method just gives the potential for the speed-up. Good speed-up can be achieved only if the loads of the processors are balanced.

Designing the parallel execution of the DES segment and the TFA segment for more than two processors

If we consider that the DES segment is responsible for a small part of the network only (e.g. 1%) and the TFA part analyses all the remaining part (e.g. 99%) then even if TFA is much faster then DES (e.g. 10 times) the execution of the TFA part may last significantly longer then the execution of the DES part (e.g. 9.9 to 1). This observation gives us an idea for producing excellent speed-up: let us use n number of processors that are parallelly executing the evaluation of the TFA part for the consecutive $t_1, t_2, \ldots t_i, \ldots t_n$ points of virtual time. This means that altogether n+1 number of processors are used, one for the DES part and n for the TFA part.



DES
Proc. 0
TFA
Proc. 1
for t_{3k+1}
TFA
Proc. 2
for t_{3k+2}
TFA
Proc. 3
for t_{3k+3}
Note: the same TFA segment is executed parallel for different t_i points of virtual time.

Figure 2. The execution of the combined DES and TFA by 3+1 processors

Out of the n+1 processors p_0 executes the DES segment. At the t_0 point of virtual time it schedules for the $p_1, p_2, \ldots p_i, \ldots p_n$ processors the execution of the TFA segment for the $t_1, t_2, \ldots t_i, \ldots t_n$ points of virtual time, respectively. When p_0 reaches t_1, it has to wait for p_1 for the results of the TFA. (The processors p_2 to p_n are going to be ready with their task at the same wall clock time as p_1 and they have to wait for p_0 until it reaches the appropriate t_i to poll them for the result of TFA. See Figure 3. This way some processor time is wasted in the initial transient.) When p_0 receives the result of TFA from p_1, then p_0 schedules for p_1 the execution of TFA for the t_{n+1} point of virtual time, and in the same way when p_0 receives the result of TFA from p_i, then p_0 schedules for p_i the execution of TFA for the t_{n+i} point of virtual time (i=1..n).

In the next round there will be no significant waiting, let us see, why. When p_0 reaches the t_{n+1} point of virtual time p_1 must be ready with the TFA for t_{n+1}, as p_1 received this task at t_1 so it had enough time. Similarly, when p_0 reaches the t_{n+i} point of virtual time, p_i must be ready with the TFA for t_{n+i}, as p_i received this task at t_i so it had enough time (i=1..n). Of course p_0 always schedules the next task for the p_i processor, when it gets the result from p_i.

In the general step, let k denote the number of the round (k=0, 1, 2, 3, …). Now, p_0 gets the results TFA from p_i at t_{kn+i}, and asks p_i to run TFA for the $t_{kn+i+n} \equiv t_{(k+1)n+i}$ point of virtual time (i=1..n).

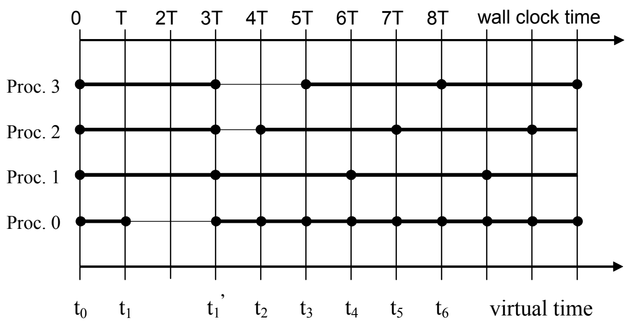


Figure 3. The operation of the combined DES and TFA, running parallelly, using 3+1 processors

Compared to the case where the combination of DES and TFA is executed in a sequential way, the ***applicability criteria*** for the parallel execution are stronger:

1. The results of TFA produced at t_i characterize the TFA→DES traffic well until t_{i+1}. (old criterion)

2. The DES→TFA traffic is either negligible, or predictable for the t_{i+n} point of virtual time at t_i point of virtual time. (new criterion)

For determining the optimal value of n, let us use the following simplifications. (They are used for the easy calculation of n and do not have to stand for the system always – they are not part of the applicability criteria.) For simplicity, let $t_i=t_0+i$*T. Let w^{DES}_i denote the computational work of the DES part from t_{i-1} to t_i. Again, for simplicity, let:

$w^{DES}_i = w^{DES}$ for all i (that is: constant)

And for the TFA part, let w^{TFA}_i denote the computational work necessary for the analysis at t_i virtual time. Again, let:

$w^{TFA}_i = w^{TFA}$ for all i (that is: constant)

Then n can be expressed as:

$$n_x = \frac{w^{TFA}}{w^{DES}}, \quad n = \lceil n_x \rceil$$

Note: we use $n = \lceil n_x \rceil$ to be able to utilise all possible parallelism. We consider n optimal in the sense that we can utilise all the available parallelism, and the speed-up is nearly linear (the processors do not have to wait for each other). If we have m < n+1 number processors only, we shall experience less speed-up than it is potentially available (and achievable using n+1 processors).

If w^{DES}_i is not constant (w^{TFA}_i is still constant) and it has w^{DES}_A average value, we can use that for the calculation of n_A. (This consideration covers also the case if $t_i=t_0+i$*T does not stand.)

$$n_A = \left\lceil \frac{w^{TFA}}{w^{DES}_A} \right\rceil$$

If $w^{\mathrm{DES}}{}_i < w^{\mathrm{DES}}{}_A$ for a given i, than p_0 has to wait for the results of TFA, and if $w^{\mathrm{DES}}{}_i > w^{\mathrm{DES}}{}_A$ occurs for another i, then the appropriate processor working on TFA has to wait for p_0.

If $w^{\mathrm{TFA}}{}_i$ is not constant (but $w^{\mathrm{DES}}{}_i$ is constant) we can use its maximum value $w^{\mathrm{TFA}}{}_M$ for the calculation of n_M to ensure that p_0 never has to wait.

$$n_M = \left\lceil \frac{w^{TFA}{}_M}{w^{DES}} \right\rceil$$

The problem deserves further discussion (e.g. dynamic scheduling of the n+1 processes that have different work-loads to be executed by m<n+1 number of processors), but it exceeds the space limitations of this paper.

Multiple DES segments are executed parallelly

Another possible way of parallelisation is that the DES part is partitioned and the partitions are executed parallel. Any of the before mentioned PDES synchronization methods can be used; the choice must depend on the system we model and of course on the simulation environment we use (which method is supported). In the basic case only one of the DES partitions has immediate connection to the TFA part, and it is executed in one process with the TFA part.

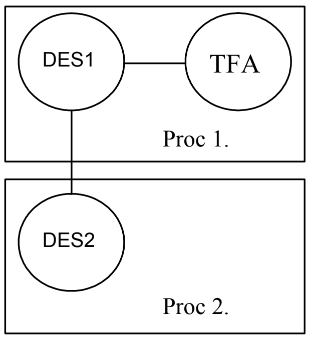


Figure 4. The DES part of the combined DES and TFA is further divided and executed by two processors

Of course the DES part can be cut into more than two partitions. And there are further possibilities. Using our previous results, the TFA part can be executed parallelly by a different processor, then the DES part that is connected to it.

Multiple TFA segments are executed parallelly

We may have a system, where the TFA part can be cut into partitions that do not have cross traffic between each other. For example let us imagine the network of a company. It has a firewall, some servers, intranet PC-s and an outside internet connection. If we would like to examine the performance of the web server, we can set up the following model: the DES part contains the firewall and the web server, the TFA1 part is the intranet and the TFA2 part is the public internet.

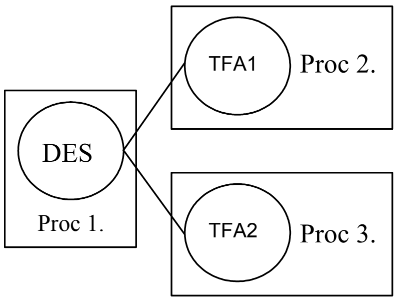


Figure 5. The execution of multiple TFA segments parallel

Using our previous results on how to execute the DES segment and one TFA segment parallel, we can do it with multiple TFA segments, too. Note, that this case is different from Figure 2, where the same TFA segment was executed parallelly for different points of virtual time. Of course the two solutions can even be combined!

And we have even more possibilities. We can combine the multiple TFA segments and the multiple DES segments, too.

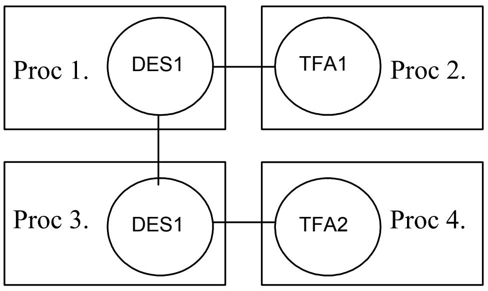


Figure 6. Parallel execution of multiple DES and multiple TFA segments

And this solution still can be made even faster with the trick of executing the same TFA segments in multiple instances by separate processors for the consecutive t_i points of virtual time.

THE METHODS WE DO NOT RECOMMEND

Until now, we put parallelism into TFA by two ways:

1. Executing the same TFA segment parallelly for different t_i points of virtual time (Figure 2)

2. Executing multiple TFA segments parallelly if there was no cross traffic between these segments (Figure 5)

Why do not we cut the TFA segment into smaller parts? (Note: we do the very thing with the DES model when we use PDES.) The reason is the operation of TFA. For its first step (the spatial distribution of the traffic) it has to know the whole topology that is relevant for its routing algorithm. There might be a point in parallelisation for special topologies (e.g. hierarchical or tree structured networks) nevertheless the commonly used large networks would probably not fit into these categories.

Another possible way would be to use multiple processors for the second step (correction for the finite capacities) only, as this could be done independently. So the first step should be done centralized and the second one distributed. This solution would require more experience with TFA than we now have (to decide if it worth doing so).

FUTURE WORK

The recommended methods should be tested by applying them to real life problems, to be able to see whether they produce reliable data about the examined network and if they show good enough speed-up.

TFA was developed for the Elassys Consulting Ltd. as a part of the Iminet Network Expert System. (Elassys 2005) The company plans to apply TFA, its combination with DES and some of the parallel versions described above to test both the reliability of the results and the speed-up produced by these methods.

As for the Iminet kernel, it does not need any modification, because it supports the parallel execution by multiple processors (it uses PVM) and it provides a very simple mechanism for the inter-segment synchronization. The user (the implementer of the simulation model) may send a `synch-point(time)` message from one segment to another. The target segment's local virtual time may not pass `time` (and its execution is suspended if it has no more events with less or equal timestamp than `time`) until a message from the source segment arrives. The first synchronization points are sent at the beginning of the simulation and the user must take care to send the next synchronization point always *before* he sends the expected message that deletes the actual synchronization point. This mechanism can be used for conservative or statistical synchronization, too.

CONCLUSIONS

We have reviewed the PDES methods and other similar performance analysis methods for communication networks.

We found that even more efficient methods can be developed by the combination of the known methods.

We have analysed the different ways for inserting parallelism into the combined DES + TFA method. We have shown several solutions. We have presented an algorithm how the same TFA segment can be run in multiple instances analysing the same system for different points of virtual time.

We have also mentioned some ways we do not think promising hence we do not recommend, though we do not state they would be surely useless.

We conclude that the recommended methods are worth further studying and testing, and it is planned to be done so.

ACKNOWLEDGEMENT

This research was financed by the Hungarian Academy of Sciences under the János Bolyai research grant.

This paper was supported by the Elassys Consulting Ltd.

REFERENCES

Elassys Consulting Ltd. 2005. "Iminet Network Expert System" http://www.elassys.hu

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

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, 3-8.

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, 159-164.

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

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

Lencse, G. 2004. "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.) EUROSIS-ETI, 89-93.

Lencse, G; L. Muka 2005. (expected) "Convergence of the Key Algorithm of Traffic-Flow Analysis" *Journal of Computing and Information Technology,* - accepted for publication

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

AUTHOR BIOGRAPHY

GÁBOR LENCSE received his M.Sc. in electrical engineering and computer systems from the Technical University of Budapest in 1994 and his Ph.D. in 2000. The area of his research is (parallel) discrete event simulation methodology. He is interested in the acceleration of the simulation of communication systems. Since 1997, he has worked for the Széchenyi István University in Győr. He teaches computer networks, networking protocols, network security and Linux. At the moment, he is an associate professor.

He has been doing research and development in the field of the simulation of communication systems for Elassys Consulting Ltd since 1998.