Simulation versus Analytic-Numeric Methods: a Petri Net Example

B. Tuffin
IRISA-INRIA
Campus universitaire de Beaulieu
35042 Rennes cedex
France

C. Hirel and K.S. Trivedi
CACC
Dept. of Electrical and Computer Eng.
Duke University
Durham, NC 27708-0291, U.S.A.

Abstract

Performance or dependability analysis is a tremendous challenge for the design or improvement of modern complex systems. Two different classes of solution methods are usually used: analytic-numeric methods and simulation methods. Surprisingly the choice between them in the literature depends more on the analyst's background than on the system itself. In this paper, we aim to illustrate on real problems the advantages and drawbacks of each method and to compare the results. Then we give some hints to choose the method. This is done by using SPNP, a Petri net analysis package including both kinds of technique.

I. Introduction

Performance and dependability evaluation of modern systems is a challenging problem due to the complexity of these systems. Many methods are available in the literature. The most interesting is the analytic one as it gives accurate results. Unfortunately, it becomes very quickly inapplicable, because of the size of the model or due to, for example, its non-Markovian nature, so we need to apply approximation methods. Even these approximation methods may become inefficient, in which case the last resort is to use simulation.

In the literature, the authors either use analytic-numeric methods, or simulation, but the choice is often arbitrary so a careful comparison of the two methods could

be useful. In fact, depending on model parameter settings, it may be useful to switch between the two methods. We wish to illustrate here the advantages and drawbacks of these methods in order to help a user to choose which one might be the most efficient.

We consider here as an illustration a client/server system, that is a system where a server station receives requests from its client stations, processes the requests and replies to the requesting stations [13]. We will compare the methods while varying the number of stations connected to the server.

The analysis of the example will be made using SPNP (Stochastic Petri Net Package) [7], [11]. The systems are modeled by Stochastic Reward Nets (SRNs) [14], an extension of generalized stochastic Petri nets [1]. Originally, SPNP only included analytic-numeric methods for studying these models [7], but the power of this package has been recently enhanced by the introduction of simulation methods [8], [9], [11], [22].

The lay-out of the paper is as follows. In Section II we introduce stochastic Petri nets, and the solution methods included in SPNP, the software used throughout this paper. Section III deals with the analysis of the client/server system and compares simulation and analytic-numeric methods. Finally, we give our conclusions in Section IV.

II. Stochastic Petri nets (SPNs) and solution methods in SPNP

Petri nets are formal graph models particularly well suited for representing the flow of information and control in systems with concurrency and synchronization characteristics [17]. We limit ourselves here to discrete SPNs, eventhough Fluid Stochastic Petri Nets (FSPNs) [8], [12], [20], hybrid extension of SPNs, can be

analyzed by SPNP as well.

The class of stochastic Petri Nets we consider is given by an 11-tuple:

$$(P, T, D, D^0, g, m^0, >, d, r, F, \omega)$$

where

- P is the set of places (represented by circles). Each place may contain *tokens*. The *marking* of the SPN is then defined by the number of tokens in each place.
- T is the set of transitions (represented by bars).
- D is the set of input arcs (from a place to a transition) and output arcs (from a transition to a place). Each arc has a multiplicity (default is multiplicity one). A transition is *enabled* if each of its input places contains at least as many tokens as the multiplicity of the corresponding input arc. The transition which *fires* is then the one with the smallest firing time. Then the transition removes a number of tokens from each of its input places equal to the multiplicity of the corresponding input arc and it deposits in each of its output places a number of tokens equal to the multiplicity of the corresponding output arc, leading then to possibly a new marking.
- D^0 is the set of inhibitor arcs, from a place to a transition (represented by an arc terminated by a small circle), with its associate multiplicity. With this addition, a transition will be enabled only if each of its inhibitor input places contains a number of tokens strictly less than the multiplicity of the corresponding arc.
- g is the (marking-dependent) guard function for each transition. It is a generalization of inhibitor arcs, saying whether or not a transition is enabled

in the current marking. Nevertheless we still also consider inhibitor arcs for their graphical usefulness and for sake of generality.

- m^0 is the initial marking.
- $>$ is the explicit priority that can be assigned to transitions.
- d defines the firing time distribution of each transition.
- r is the static resampling policy for each transition when it becomes enabled again after being disabled by the firing of a concurrent transition. Three policies are possible: PRI (preemptive repeat identical), PRD (preemptive repeat different) and PRS (preemptive resume).
- F defines the affecting resampling policy for each transition when another transition fires but the considered transition remains enabled. PRI, PRS and PRD are also possible.
- ω is the weight function to choose between several transitions when they are all enabled and have the same firing time.

This model (as well as its fluid extension) is used in SPNP, a versatile modeling tool for performance, dependability and performability analysis of complex systems [7] including advanced constructs such as marking-dependency or the ability to define its own reward function. The Petri nets and the solution methods are described in CSPL, an ANSI C library, but can also be specified by using a Graphical User Interface (GUI) [11]. Steady-state as well as transient analysis are possible.

The first class of solution methods is the analytic-numeric one. To apply these methods, several restrictions must be applied on the previous model. The first major limitation is that the distributions of transition firing times must generally be exponential or immediate. There exist less general restrictions (see for instance [4],

[5] where no more than one non-exponential distribution is enabled in any marking, leading to a Markov regenerative process), but they are still restrictive and are not currently implemented in SPNP yet. One could emphasize that general distributions can be approximated by phase type distributions, that is, by creating extra places and extra exponential transitions [16]. Nevertheless this can also increase the size of the state space. A second assumption we make is that functions F and r have fixed policies, PRD for r and PRS for F (this has been relaxed by some authors [2]). Given these assumptions, a Markov chain is constructed via the *reachability graph*. Several matrix analysis methods may then be used to solve the problem. Analytic-numeric solution methods in SPNP include steady-state SOR (Successive OverRelaxation), steady state Gauss-Seidel, steady-state power method [6], [18], [19] or transient solution using uniformization [15].

On the other hand, simulation methods [9], [10] can be used when the above restrictions are not satisfied, when the the storage requirements exceeds the memory capacity or when the computation time is very long. In fact, no generation of reachability graph is needed, so the simulation requires very little memory, and the computation time can be reduced by decreasing the number of replications while reducing accuracy. However the simulation time can be quite often long and all the simulation methods (except regenerative simulation) are actually transient simulations, introducing a bias. The simulation methods included in SPNP encompass the standard discrete event simulation with independent replications or with batches [9], regenerative simulation [21], and some variance reduction techniques well suited for rare event situation, importance splitting [22] and importance sampling [21]. As the following example does not involve rare events, we compare analytic-

numeric methods with the standard discrete event simulation.

III. CLIENT/SERVER EXAMPLE

The example we consider is that of a client/server system where a server station receives requests from its client stations, processes the requests and replies to the client stations [13]. This is a common feature in distributed computing but its analysis is made difficult by the various kinds of dependencies in the system [13]. A Markov chain will be able to capture the dependencies but a hand construction of the Markov chain is infeasible. For this reason, the use of SRNs as model of representation is very helpful. We consider here a distributed system consisting of N workstations and one file server interconnected by a local area network. For a complete description of the models, the reader is advised to read [13]. To avoid any confusion in the use of the word token, used in PNs and in ring networks, we will refer throughout this section to the network token or the PN token. We assume here that a client-station generates requests following an exponential distribution with rate λ and that the transmission time of this request is also exponentially distributed with rate μ. Other times (all assumed to be exponentially distributed) are the time for the network token to move from a station to another one (rate γ), the request processing time for the server (rate η) and the reply transmission time (rate β). Figure 1 shows the SRN model for a token ring network-based system with five client stations. Places P_{CkI} ($1 \leq k \leq N$) represent the condition that station k is idle and transitions tak that a request is generated at station k. Then the token moves to place P_{CkA} where the client is waiting for the network token to arrive (condition represented by place P_{CkS}). When the network token arrives at station k, the transmission of the request can be processed (transition tsk). Then a PN token is put in

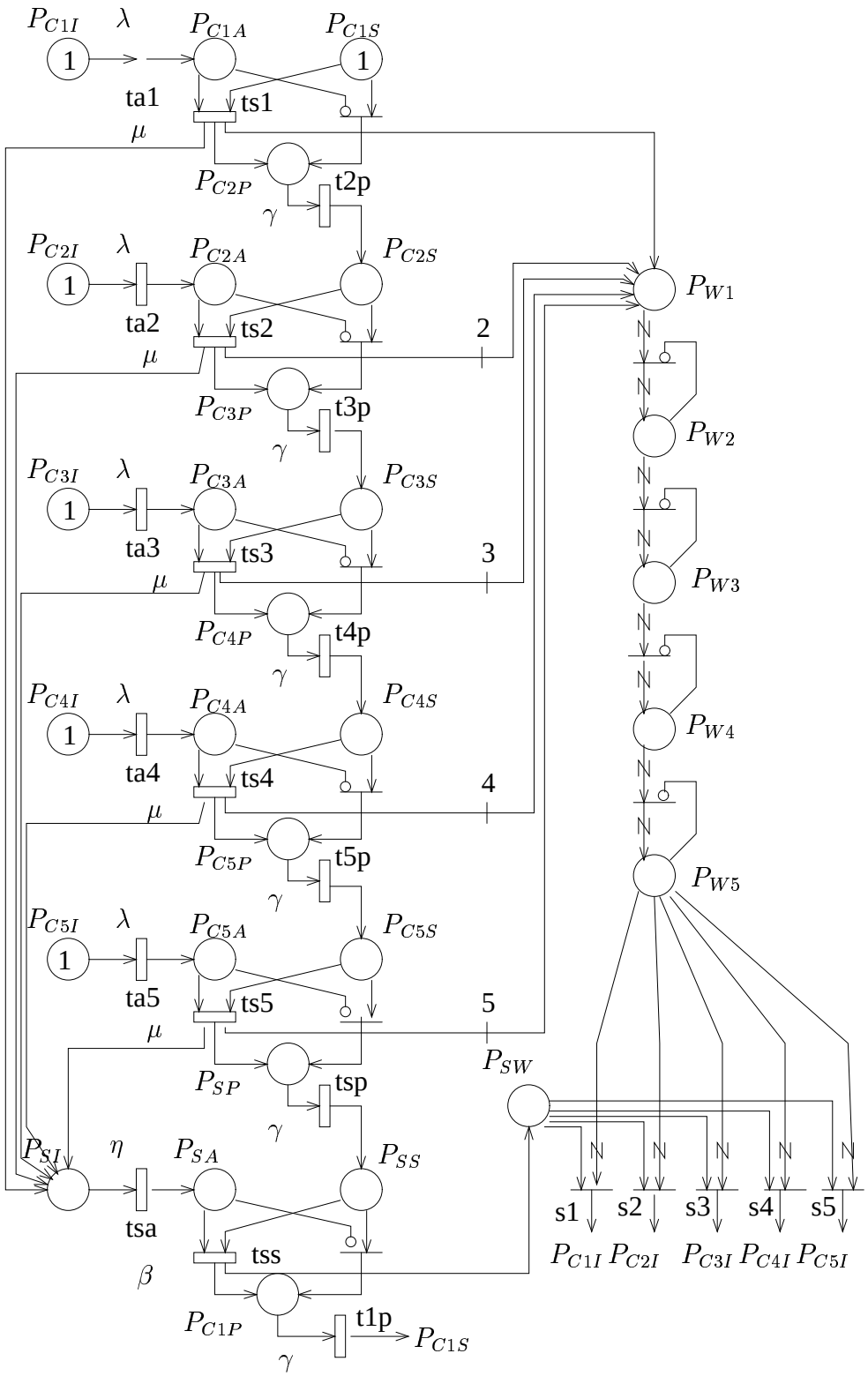

Fig. 1. SRN the accurate token ring network ($N > 1$)

place $P_{C(k+1)P}$ (or P_{SP} if $k = N$, or P_{C1P} if the current station is the server), meaning that the network token is waiting to move to the next station (or to the server for P_{SP}). This move is made by the firing of the corresponding transition tkp or tsp. Place P_{SI} represents the condition that the client's request has arrived at the server, where it is served by firing transition tsa. Therefore, place P_{SA} represents the condition that the request is completed. When the server has received the network token (condition represented by place P_{SS}) , it can commence transmitting an answer (by firing transition tss). Next we describe the modeling of the server's buffer. Places P_{Wk} represent the condition that a request is waiting for its reply at k^{th} slot of the queue from the tail. The multiplicity of input arcs from transitions tsk to P_{W1} is k, to identify the requesting stations. The firing of transition sk ($1 \leq k \leq N$) means that the server sends a reply to station k. A token in place P_{SW} means that the service is completed, so we can empty the first slot of the FIFO queue.

In [13], an approximation of this model is also described to reduce the state space size. This is done by considering a tagged client and lumping the remaining clients into one super-client. The SRN for this approximation for the system with $N > 1$ stations is given in Figure 2. For a detailed description of this model, the reader can consult [13].

Table I describes the state space and storage requirements for both the exact and approximate models, and when a memory overflow is obtained on a Sun SparcStation Ultra 60 with 640Mb of real memory and 982Mb of swapping memory. Nonzero entries are the number of nonzero elements in the infinitesimal generator of the underlying continuous time Markov chain. We observe that the state space size of the exact model increases quickly and becomes too large to construct the

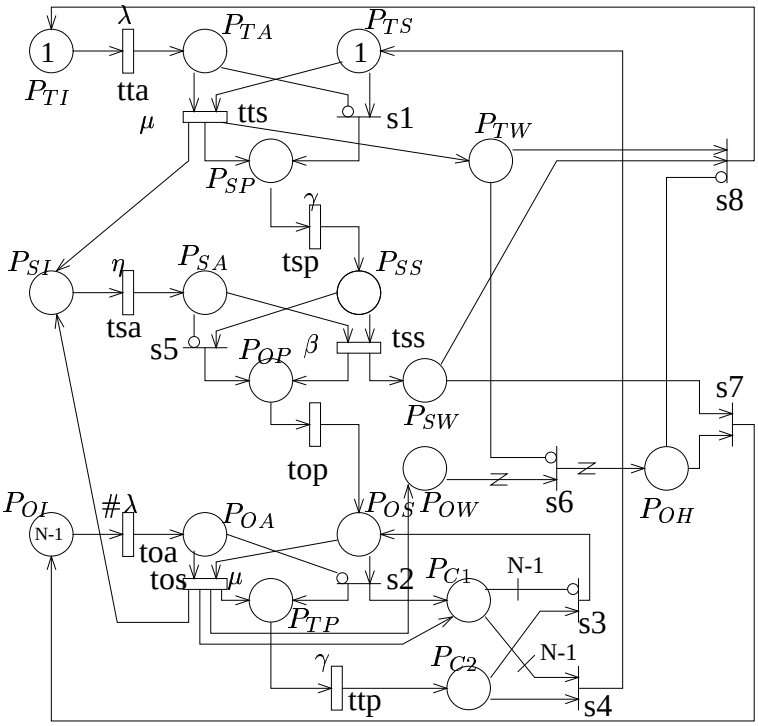

Fig. 2. SRN the approximate token ring network ($N > 1$)

reachability graph for small values of N ($N = 8$), even on our powerful computer. On the other hand, the approximate model reachability graph can be generated for larger values of N, but limited to $N = 31$ on our computer.

A. Transient behavior

A.1 Cumulative measure

Table II gives the results obtained when computing the transient cumulative probability that the server is idle (i.e., places P_{SA} and P_{SI} are empty) up to time $t = 500$ ms, using analytic-numeric methods and simulation (with 10% relative error and 95%-confidence interval) when the number N of workstations is varying. In these computations, we assume that $1/\eta = 2.0$ ms, $1/\mu = 0.1$ ms, $1/\beta = 3.2$ ms, $1/\gamma = 0.01/(N + 1) + 0.0024$ ms and that λ is varied with N so that the offered

N	No. of states (exact model)	Nonzero Entries (exact model)	No. of states (approx. model)	Nonzero Entries (approx. model)
3	476	1004	274	562
4	3416	7960	790	1754
5	26672	66192	1880	4400
6	228880	591568	3920	9524
7	2160160	5736992	7420	18536
8	Memory overflow	Memory overflow	13044	33292
10			34210	90050
15			209240	574700
20			785470	2204850
25			2230150	6343250
30			5281780	15156400
31			6172720	17738324
32			Memory overflow	Memory overflow

TABLE I

STORAGE REQUIREMENTS FOR THE CLIENT/SERVER EXAMPLE AS THE NUMBER N OF CLIENT STATION INCREASES

N	Exact model numer. res.	Approx. model numer. res.	Exact model simulation	Approx. model simulation
3	0.2501126340	0.2500131794	[2.4623e-01,3.0010e-01]	[2.2962e-01,2.7220e-01]
4	0.2177441140	0.2175922454	[1.8141e-01,2.2149e-01]	[1.8664e-01,2.2796e-01]
5	0.1957598128	0.1955790301	[1.7852e-01,2.1472e-01]	[1.7821e-01,2.1775e-01]
6	0.1796752514	0.1794767011	[1.3103e-01,1.5994e-01]	[1.7196e-01,2.0928e-01]
7	0.167301986	0.1670913055	[1.4746e-01,1.7913e-01]	[1.5417e-01,1.8842e-01]
8		0.1572124840	[1.3973e-01,1.7064e-01]	[1.3485e-01,1.6458e-01]
9		0.1491128548	[1.3349e-01,1.6218e-01]	[1.2827e-01,1.5670e-01]
10		0.1423266644964	[1.2955e-01,1.5799e-01]	[1.3960e-01,1.7019e-01]
15		0.1198182597846	[1.1394e-01,1.3894e-01]	[1.1342e-01,1.3843e-01]
20		0.106803594142	[9.5962e-02,1.1727e-01]	[9.8970e-02,1.2072e-01]
25		$\cdots$	[9.4095e-02,1.1495e-01]	[8.8553e-02,1.0816e-01]

TABLE II

RESULTS OBTAINED FOR ANALYTIC-NUMERIC AND SIMULATION METHODS FOR THE CUMULATIVE BEHAVIOR.

load $\rho = N\lambda[(1/\mu) + (1/\beta)]$ is fixed to 0.9. The analytic-numeric method uses uniformization [15] whereas the simulation method is the standard discrete event simulation using independent replications. The number of replications is unknown as the simulation is stopped only when a 10% half-width relative error 95% confidence interval is reached. We do not give all the results because the computation time can be very long for large values of N. We observe that the numerical values given by the approximate model are very close to the exact ones. Of course, we obtain exact results only for small values of N so that we can not be sure that the results are very good also for larger values, but simulation results show that the ap-

N	Exact model numer. res.	Approx. model numer. res.	Exact model simulation	Approx. model simulation
3	19.52	45.25	89.91	45.71
4	196.2	159.14	181.86	150.65
5	2106.34	432.56	154.22	92.92
6	26231.21	952.25	221.40	246.53
7	324109.00	1922.95	716.86	331.95
8		3569.55	986.33	347.60
9		6386.69	904.97	337.40
10		11914.97	2466.47	357.78
15		83379.73	6704.03	719.60
20		339514.88	10562.27	706.04
25		$\cdots$	17892.82	593.45

TABLE III

COMPUTATION TIMES (IN SECONDS) TO OBTAIN THE ANALYTIC-NUMERIC AND 10% HALF-WIDTH RELATIVE ERROR 95% CONFIDENCE INTERVAL SIMULATION RESULTS FOR THE CUMULATIVE ESTIMATION.

proximation is still accurate as N increases. The simulation results are in the range of the analytic-numeric ones. Only in the case of the exact model with $N = 6$ the exact value is not included in the confidence interval. This can have two explanations: first it can be due to risk of the confidence interval. In fact, the confidence level is 95%, which means that in 5% of cases the values may not be included in the interval. This is one of the main drawbacks of the simulation: we are never sure that the interval contains the exact value. More likely in our example, the error is due to the fact that very few replications (here 10) are necessary to obtain a 10% relative error accuracy so that the normal approximation (and variance estimation) may be bad. This number of replications is small because the variance of the estimator of cumulative probability is small. In conclusion about Table II, simulation allows us to solve larger models, but this is at the cost that we obtain only a confidence interval, i.e., we have only a given probability that the true value is contained in the interval.

Table III shows the computation times required for the methods to obtain the results. The computation time for the analytic-numeric solution of the exact model increases very fast with the state space size. The increase is slower for the approx-

imate model. Simulation time to obtain the 10% relative error confidence interval becomes competitive for $N = 5$ and is close to the analytic-numeric time for smaller N. In this example, simulation is then a better method to use (and even the only one as soon as there is a memory overflow while using analytic-numeric methods; see TableI). The reason of the very different simulation times between the exact and the accurate model is due to the number of events to simulate per run. Recall that the state space does not intervene in this difference as for analytic-numeric methods because we do not generate it. Here it is specific to the model in itself: in the exact model, more transitions occur, which results in a bigger number of events per run. Thus it is very interesting that simulation of approximate models can also be much more powerful than simulation of exact models because when applying simulation, people almost always use exact models.

A.2 Instantaneous behavior

We now compute the transient probability that the server is idle (i.e., places P_{SA} and P_{SI} are empty) at given time t, using analytic-numeric methods and simulation (with 10% relative error 95%-confidence interval) with the number N of workstations varying. Other parameters have the same values as in the previous subsection.

The results for $t = 500$ ms are displayed in Table IV and the computation times are given in Table V. Applying the numerical solution method to the exact model requires a very long time for $N = 5, 6, 7$ and afterthat it cannot be applied. The method performs well on the approximate model (restricted to the fact that $N < 31$), but the running time increases quickly with N. If we compare with simulation, we see that the results match (and do not forget anyway that we are statistically authorized 5% error). Similarly, if we compare the simulation results for the models,

N	Exact model numer. res.	Approx. model numer. res.	Exact model simulation	Approx. model simulation
3	0.245405983462	0.245309152837	[2.3759e-01,2.9029e-01]	[2.1142e-01,2.5833e-01]
4	0.212233219255	0.212087343327	[2.0257e-01,2.4758e-01]	[2.0477e-01,2.5019e-01]
5	0.189598853338	0.189427606917	[1.7465e-01,2.1338e-01]	[1.7795e-01,2.1745e-01]
6	0.172966955725	0.17278150511	[1.4788e-01,1.8069e-01]	[1.5488e-01,1.8925e-01]
7	0.160119663329	0.159925613372	[1.5103e-01,1.8458e-01]	[1.3570e-01,1.6582e-01]
8		0.149631257743	[1.3691e-01,1.6733e-01]	[1.2700e-01,1.5518e-01]
9		0.141158476542	[1.2036e-01,1.4711e-01]	[1.1887e-01,1.4527e-01]
10		0.134033184	[1.2125e-01,1.4817e-01]	[1.1019e-01,1.3465e-01]
15		0.110165627749	[1.0836e-01,1.3240e-01]	[1.0579e-01,1.2926e-01]
20		0.096121345693	[8.5813e-02,1.0487e-01]	[9.4162e-02,1.1506e-01]
25		...	...	[8.2386e-02,1.0067e-01]

TABLE IV

ANALYTIC-NUMERIC RESULTS AND 10% HALF-WIDTH RELATIVE ERROR 95% CONFIDENCE INTERVAL SIMULATION RESULTS FOR THE INSTANTANEOUS BEHAVIOR AT $t = 500$ MS.

N	Exact model numer. res.	Approx. model numer. res.	Exact model simulation	Approx. model simulation
3	7.50	20.55	8773.120	8656.66
4	77.12	73.84	15543.22	9715.90
5	821.17	196.48	27302.46	12194.67
6	10778.41	451.20	42109.10	14987.88
7	125182.97	941.24	54362.62	18113.44
8		1742.30	74371.53	19286.46
9		3092.11	107277.69	21250.31
10		5687.44	124570.68	23331.97
15		42696.84	286364.04	23882.59
20		158124.71	586347.98	27547.62
25		...	...	30651.96

TABLE V

COMPUTATION TIMES FOR THE ANALYTIC-NUMERIC AND 10% HALF-WIDTH RELATIVE ERROR 95% CONFIDENCE INTERVAL SIMULATION RESULTS FOR THE INSTANTANEOUS BEHAVIOR AT $t = 500$ MS.

we see that the approximate model gives about the same results as the exact one even for large values of N. The problem here is that simulation times are very long. This is due to the fact that to obtain just one replication at time $t = 500$ ms, we need to simulate the whole path to this time, which is long. We note as in the previous subsection that the simulation of the approximate model is much quicker because we have less number of events to deal with. We suggest then to use the simulation of the approximate model as soon as N gets close to 15 as then simulation time is almost half that of the numeric method one.

Next we consider the results for a smaller time horizon, say for instance $t = 20$ ms. The results are presented in Table VI and the computation times in Table VII.

N	Exact model numer. res.	Approx. model numer. res.	Exact model simulation	Approx. model simulation
3	0.248990414584	0.248891671673	[2.1377e-01,2.6119e-01]	[2.2215e-01,2.7147e-01]
4	0.220642010649	0.220487516342	[1.8385e-01,2.2470e-01]	[1.9193e-01,2.3453e-01]
5	0.20335484339	0.203163887109	[1.9346e-01,2.3643e-01]	[1.8598e-01,2.2730e-01]
6	0.19187084927	0.191651745495	[1.7333e-01,2.1181e-01]	[1.7560e-01,2.1458e-01]
7	0.183738062491	0.183494745277	[1.7639e-01,2.1552e-01]	[1.6655e-01,2.0351e-01]
8		0.177427525223	[1.5530e-01,1.8975e-01]	[1.6515e-01,2.0177e-01]
9		0.172741528561	[1.5780e-01,1.9281e-01]	[1.4994e-01,1.8322e-01]
10		0.169013059683	[1.4478e-01,1.7693e-01]	[1.4788e-01,1.8069e-01]
15		0.157887817108	[1.4537e-01,1.7763e-01]	[1.5103e-01,1.8458e-01]
20		0.152260872721	[1.3929e-01,1.7023e-01]	[1.6124e-01,1.9704e-01]
25		0.148769045652	[1.2261e-01,1.4986e-01]	[1.3494e-01,1.6492e-01]

TABLE VI

ANALYTIC-NUMERIC RESULTS AND 10% HALF-WIDTH RELATIVE ERROR 95% CONFIDENCE INTERVAL SIMULATION RESULTS FOR THE INSTANTANEOUS BEHAVIOR AT $t = 20$ MS.

N	Exact model numer. res.	Approx. model numer. res.	Exact model simulation	Approx. model simulation
3	1.96	1.08	506.12	400.25
4	17.01	3.60	925.64	617.95
5	169.82	9.38	1330.08	698.69
6	1779.51	21.22	1995.13	801.36
7	19878.53	42.27	2856.53	859.87
8		81.14	4094.41	892.77
9		143.11	5232.79	1153.12
10		232.45	6807.45	1109.47
15		1761.66	16296.33	1198.21
20		7114.52	27600.81	1174.72
25		21015.08	50124.95	1664.06

TABLE VII

COMPUTATION TIMES FOR THE ANALYTIC-NUMERIC AND 10% HALF-WIDTH RELATIVE ERROR 95% CONFIDENCE INTERVAL SIMULATION RESULTS FOR THE INSTANTANEOUS BEHAVIOR AT $t = 20$ MS.

The same type of remarks can be applied to the results as in the case $t = 500$ ms (but in one case, approximate model with $N = 20$, the exact value is not included in the confidence interval, probably due to the statistical risk), with the difference that the running times are smaller because the time horizon is smaller. Simulating the approximate model is the best method when N gets close to 20 and the simulation times are then less than 20 minutes.

As a conclusion of this subsection, simulation for instantaneous behavior is better when "small" time horizons are used. Anyway, it is the only solution when the state space is big (here when $N > 31$), but the results will take a very long time for large horizon times.

N	Exact model numer. res.	Approx. model numer. res.	Exact model simulation	Approx. model simulation
3	0.245405986097	0.245309143644	[2.0963e-01,2.5612e-01]	[2.2727e-01,2.7776e-01]
4	0.212233217673	0.212087148724	[1.8145e-01,2.2174e-01]	[1.7796e-01,2.1748e-01]
5	0.189598852479	0.189427509192	[1.6452e-01,2.0103e-01]	[1.7541e-01,2.1437e-01]
6	0.172966955399	0.172781457818	[1.4520e-01,1.7746e-01]	[1.6325e-01,1.9949e-01]
7	0.16011966318	0.159925589231	[1.5756e-01,1.9253e-01]	[1.4701e-01,1.7963e-01]
8		0.149631243957	[1.4030e-01,1.7145e-01]	[1.4145e-01,1.7283e-01]
9		0.141158469595	[1.2664e-01,1.5477e-01]	[1.1697e-01,1.4294e-01]
10		0.134033180235	[1.1820e-01,1.4446e-01]	[1.2562e-01,1.5352e-01]
15		0.110165627613	[9.1028e-02,1.1124e-01]	[1.1056e-01,1.3509e-01]
20		0.0961213455293	[9.4511e-02,1.1551e-01]	[6.4175e-02,7.8425e-02]
25		0.0865441102832	[8.1165e-02,9.9175e-02]	[8.4345e-02,1.0309e-01]

TABLE VIII

ANALYTIC-NUMERIC AND 10% HALF-WIDTH RELATIVE ERROR 95% CONFIDENCE INTERVAL SIMULATION RESULTS FOR THE STEADY-STATE BEHAVIOR.

B. Steady-state behavior

We now compute the steady-state probability that the server is idle (i.e., places P_{SA} and P_{SI} are empty) using analytic-numeric methods and simulation (with 10% relative error 95%-confidence interval) with the number N of workstations varying.

The analytic-numeric method used is steady-state SOR. As simulation method, the standard discrete event simulation with independent replications is not relevant anymore for it is meant for the transient behavior and it would require very long simulation paths to approach a steady state behavior. A better way is use batching techniques. Then only one simulation path is generated so that we can assume that steady-state will be reached (a warm-up can be used). This path is decomposed in several blocks assumed to be independent, so that a statistical analysis is performed using the method of batch means [19].

The results for analytic-numeric methods and simulation using batching methods (with a batch size of $t = 5$ ms) are displayed in Table VIII. The computation times are presented in Table IX. The analytic-numeric is powerful when applied on the exact model until $N = 6$. Beyond $N = 6$, the running time is long. Use of the approximate model is much better as the results are very close and the running

N	Exact model numer. res.	Approx. model numer. res.	Exact model simulation	Approx. model simulation
3	5.60	0.79	65.82	49.32
4	11.11	1.74	107.18	74.30
5	105.08	4.31	196.00	79.15
6	1078.54	9.43	255.12	92.39
7	13749.02	18.17	331.55	110.46
8		32.14	511.63	114.35
9		54.61	544.08	136.92
10		99.66	734.51	132.56
15		660.64	1905.18	163.36
20		2444.84	3266.75	222.21
25		6807.75	5918.24	202.02

TABLE IX

COMPUTATION TIMES FOR THE ANALYTIC-NUMERIC AND 10% HALF-WIDTH RELATIVE ERROR 95% CONFIDENCE INTERVAL SIMULATION RESULTS FOR THE STEADY-STATE BEHAVIOR.

times are much smaller (18 seconds as compared with 3 hours and 50 minutes for $N = 7$). But even the approximate model can not be used if $N > 31$. Simulation of the exact model is powerful for small values of N but the simulation time increases with N. On the other hand, simulation of the approximate model requires very small running times (about 3 minutes 30 seconds for $N = 20$ or $N = 25$). Thus the simulation of the approximate is a very powerful technique. The problem when using batch methods for steady-state estimation is the choice of the batch size. It has to be big enough so that successive blocks are nearly independent. In our case, the exact value gets closer to the edge of the confidence interval as N increases. This suggests that the batch size should be increased with N. Indeed, when N increases the number of tokens in the system is larger and hence more events have to occur to decorrelate blocks.

Regenerative simulation is another way to perform steady-state simulation [10]. The advantage is that the variance estimation is unbiased. The drawback is that we need a regeneration point (each state is suitable in the Markovian case) where a cycle will begin. Each cycle is then independent and a statistical analysis can be performed. Confidence intervals and computation times are given in Tables XI and

N	Exact model	Approx. model
3	[2.0995e-01,2.5657e-01]	[2.2702e-01,2.7746e-01]
4	[1.9919e-01,2.4343e-01]	[1.9277e-01,2.3559e-01]
5	[1.7791e-01,2.1743e-01]	[1.6220e-01,1.9825e-01]
6	[1.6976e-01,2.0747e-01]	[1.7197e-01,2.1017e-01]
7	[1.4151e-01,1.7295e-01]	[1.5128e-01,1.8484e-01]
8	[1.4514e-01,1.7739e-01]	[1.2864e-01,1.5720e-01]
9	[1.2253e-01,1.4976e-01]	[1.2850e-01,1.5702e-01]
10	[1.2906e-01,1.5774e-01]	[1.1046e-01,1.3499e-01]
15	[9.6841e-02,1.1835e-01]	[1.0197e-01,1.2461e-01]
20	[8.8288e-02,1.0791e-01]	[8.4866e-02,1.0372e-01]
25	[8.1428e-02,9.9516e-02]	[8.2673e-02,1.0103e-01]

TABLE X

10% HALF-WIDTH RELATIVE ERROR 95% CONFIDENCE INTERVAL REGENERATIVE SIMULATION RESULTS FOR THE STEADY-STATE BEHAVIOR.

N	Exact model	Approx. model
3	140.96	118.22
4	261.74	160.10
5	461.25	232.33
6	784.07	244.99
7	954.92	323.77
8	1382.12	399.25
9	1917.75	373.76
10	2180.04	467.40
15	6633.18	561.98
20	11764.46	670.85
25	24719.10	787.32

TABLE XI

COMPUTATION TIMES TO OBTAIN THE 10% HALF-WIDTH RELATIVE ERROR 95% CONFIDENCE INTERVAL REGENERATIVE SIMULATION RESULTS FOR THE STEADY-STATE BEHAVIOR.

X. The exact values look less at the edge of the confidence intervals. Note however, at least in this model, batch method is quicker than regenerative simulation. Moreover, as N increases, the mean length of a cycle is increasing, making it more difficult to estimate the variance of the estimator as we need a sufficient number of cycles.

Note that another way to perform regenerative simulation is using A-cycles [3]. Instead of using a state as a regeneration point, we use a subset of states, hence increasing the number of regenerations. This estimator is biased (because the initial state is not generated from the regenerative subset with the steady-state distribution, which would be too difficult) and the cycles are dependent. The analysis is then

more tricky and more approximate than the classical regenerative method, but it is more promising. Nevertheless, as it is not yet implemented in SPNP, we do not use it.

IV. CONCLUSION

To choose between simulation and analytic-numeric methods, we make the following recommendations:

- When the system is non-Markovian (especially without regenerative structure), very few analytic-numeric methods are available. Simulation is then the natural and often the only possibility.
- When the system is Markovian
  - When the state space is big (and no approximate model close to the exact model reducing the state space is available), the reachability graph can not be generated. Simulation is then again the only possibility (in our example if $N > 31$).
  - In the other cases, a choice is very specific to the application. Nevertheless, from our experience and the example of this paper, we can say that for "small" state spaces analytic-numeric methods perform well. When the state space increases, there is always a point where simulation time is more efficient (the worst situation is when the reachability graph is too big to be generated). But we can point out that the natural switch form analytic-numeric to simulation methods is faster for steady-state behavior, then cumulative transient behavior and finally instantaneous transient behavior (except for very small time horizons). Usually, simulation takes a very long time to

obtain good results for long horizon instantaneous behavior.

Note that even if the running time of analytic-numeric methods is a little larger than the one of simulation, it is still relevant to use them because they give an accurate result instead of a confidence interval. How different it should be is very subjective and depends on the user. Moreover, using approximate models can be very helpful even when using simulation, as pointed out in this paper. Indeed, even if the state space is not generated as in analytic-numeric methods, an approximate model can tremendously reduce the computation time per run by reducing the number of events.

REFERENCES

[1] M. Ajmone Marsan, G. Conte, and G. Balbo. A Class of Generalized Stochastic Petri Nets for the Performance Evaluation of Multiprocessor Systems. *ACM Transactions on Computer Systems*, 2(2):93–122, May 1984.

[2] A. Bobbio, V. Kulkarni, M. Puliafito, A. an d Telek, and K. Trivedi. Preemptive Repeat Identical Transitions in Markov Regenerative Stochastic Petri Nets. In *Proc. of Petri Nets Performance Models 1995*, 1995.

[3] C. Chang, P. Heidelberger, and P. Shahabuddin. Fast Simulation of Packet Loss Rates in a Shared Buffer Communications Switch. *ACM Transactions on Modeling and Computer Simulation*, 5(4):306–325, October 1995.

[4] H. Choi, V. Kulkarni, and K. Trivedi. Markov Regenerative Stochastic Petri Nets. *Performance Evaluation*, 20(1-3):337–357, 1993.

[5] H. Choi, V. Kulkarni, and K. Trivedi. Transient analysis of deterministic and stochastic petri nets. In M. A. Marsan, editor, *Application and Theory of Petri*

Nets 1993, volume 691 of *Lecture Notes in Computer Science*, pages 166–185. Springer Verlag, 1993.

[6] G. Ciardo, A. Blakemore, P. Chimento, J. Muppala, and K. Trivedi. Automated Generation and Analysis of Markov Reward Models using Stochatic Reward Nets. In C. Meyer and R. Plemmons, editors, *Linear Algebra, Markov Chains and Queuing Models*, volume 48 of *IMA Volumes in Mathematics and its Applications*, pages 145–191. Springer-Verlag, Heidelberg, 1993.

[7] G. Ciardo, J. Muppala, and K. Trivedi. SPNP: Stochastic Petri net Package. In *Proc. Third International Workshop on Petri Nets and Performance Models, PNPM'89*, pages 142–151, 1989.

[8] G. Ciardo, D. Nicol, and K. Trivedi. Discrete-Event Simulation of Fluid Stochastic Petri-Nets. *IEEE Transactions on Software Engineering*, 25(2):207–217, 1999.

[9] Computer Science Department. College of Willian and Mary. *On the Simulation of Stochastic Petri Nets*.

[10] G. Fishman. *Monte Carlo: Concepts, Algorithms and Applications*. Springer-Verlag, 1997.

[11] C. Hirel, B. Tuffin, and K. Trivedi. SPNP Version 6.0. In B. Haverkort, H. Bohnenkamp, and C. Smith, editors, *Computer performance evaluation: Modelling tools and techniques; 11th International Conference; TOOLS 2000, Schaumburg, Il., USA*, volume 1786 of *Lecture Notes in Computer Science*, pages 354–357. Springer Verlag, 2000.

[12] G. Horton, V. Kulkarni, D. Nicol, and K. Trivedi. Fluid Stochastic Petri nets: Theory, Application and Solution. *European Journal of Operational Research*,

105:184–201, 1998.

[13] O. Ibe, H. Choi, and K. Trivedi. Performance Evaluation of Client-Server Systems. *IEEE Transactions on Parallel and Distributed Systems*, 4(11):1217–1229, 1993.

[14] M. J.K. and T. K.S. Composite Performance and Availability Analysis using a Hierarchy of Stochastic Reward Nets. In G. Balbo and G. Serazzi, editors, *Computer Performance Evaluation, Modelling Techniques and Tools*, pages 335–350. Elsevier, Amsterdam, 1992.

[15] M. Malhotra, J. K. Muppala, and K. S. Trivedi. Stiffness-tolerant methods for transient analysis of stiff Markov chains. *Microelectron. Relib.*, 34(11):1825–1841, 1994.

[16] M. Neuts and K. Meier. On the use of phase type distributions in reliability modelling of systems with two components. *Oper. Res. Spektrum*, 2(4):227–234, 1981.

[17] J. Peterson. *Petri nets and the Modeling of Systems*. Prentice-Hall, EnglewoodCliffs, NJ, 1981.

[18] W. J. Stewart. *Introduction to the Numerical Solution of Markov Chains*. Princeton University Press, 1994.

[19] K. Trivedi. *Probability and Statistics with Reliability, Queuing, and Computer Science Applications*. John Wiley, 2001. Second Edition.

[20] K. S. Trivedi and V. G. Kulkarni. FSPNs: Fluid Stochastic Petri Nets. In *14th International Conference on Applications and Theory of Petri Nets*, pages 24–31, 1993.

[21] B. Tuffin. *Simulation accélérée par les méthodes de Monte Carlo et quasi-*

Monte Carlo : théorie et applications. PhD thesis, Université de Rennes 1, Octobre 1997.

[22] B. Tuffin and K. Trivedi. Implementation of importance splitting techniques in stochastic Petri net package. In B. Haverkort, H. Bohnenkamp, and C. Smith, editors, *Computer performance evaluation: Modelling tools and techniques; 11th International Conference; TOOLS 2000, Schaumburg, Il., USA*, volume 1786 of *Lecture Notes in Computer Science*, pages 216–229. Springer Verlag, 2000.