Performance Petri Net Analysis of Communications Protocol Software by Delay-Equivalent Aggregation

C. Murray Woodside and Yao Li
Telecommunications Research Institute of Ontario
Department of Systems and Computer Engineering
Carleton University
Ottawa, Ontario, Canada, K1S 5B6
email: cmw@sce.carleton.ca, or woodside@carleton.bitnet

Abstract

Performance analysis of Petri net models is limited by state explosion in the underlying Markovian model. To overcome this problem, an iterative approximate technique is outlined, using a number of auxiliary models, each of much lower state complexity. It is demonstrated on a substantial model which represents a parallel implementation of two layers of protocols for data communications. The model represents ten separate software tasks and their interactions via rendezvous, and is based on a testbed implementation in the laboratory. Submodels can be constructed in various ways, and this is illustrated with four different decompositions. Their state space complexity, solution time and solution accuracy are evaluated.

1. Introduction

Stochastic Petri nets [12] and Generalized SPNs (or GSPNs) [11] have proven to be extremely useful in modeling the performance of communicating concurrent systems such as protocols [15], distributed software [16] and flexible manufacturing systems [8]. Although SPNs are powerful, state explosion makes performance analysis by direct Markovian analysis impossible even for systems of modest size. The purpose of this paper is to approximate the solution of SPNs and GSPNs, in a way which overcomes state explosion.

Other authors have also described decomposition techniques. Giglmayr [7] decomposed the transition rate matrix of an SPN by using the concept of "nearly complete decomposability" from [5]. Ammar *et al* suggested a time scale decomposition method to hierarchically reduce the SPN [1], which has good accuracy when transition firing rates differ by orders of magnitude. Ciardo and Trivedi developed an approximate technique [4] that decomposes a GSPN into a set of subnets and solves each individual subnet separately. Because there exist dependences among the subnets, after solving each subnet, certain quantities need to be exported to other subnets, and this is conducted iteratively. Their decomposition is based on an assumption of near-independence among subnets and on special subnet structures which may not be present in a given system. The solution accuracy is usually acceptable but sometimes is poor.

Petri net reduction is another approach to reducing the state space of a model. Berthelot *et al*, in [2,3], defined reduction rules which preserve a set of logical properties. The rules do not deal with timing parameters, and some of the rules fuse unconnected places and transitions, which may be undesirable in a performance model. Song *et al* defined two types of reducible subnets: a 'well-behaved place module' and a 'well-behaved transition module' [13]. The former can be aggregated to a single place, and the latter to a single transition while preserving some logical properties. There is a restriction on a reducible subnet that it must contain one and only one exit point, either a transition or a place. This type of reduction will be adopted with slight modification one of the three reduction rules used in this paper.

Approximations for queueing networks are also relevant to the present research, particularly the method of surrogates for simultaneous resource possession [9]. It partitions queueing delay according to which of the resources is responsible, then iterates between two models, each of which explicitly represents one resource and has a "surrogate delay" to represent the congestion at the other resource. This iteration strategy is somewhat similar to that of Ciardo and Trivedi, and has been adopted also in the present work.

To approximately solve a large GSPN, the proposed technique works in two phases: one static and one dynamic. In the static phase, identical replicates of the original model are created and then reduced in different ways to produce smaller complementary auxiliary models. During this process, different parts of the original model are retained unaggregated in each auxiliary model, such that the union of unaggregated parts covers the original SPN. The major concern in the static phase

TH0386-3/91/0000/0064/$01.00 © 1991 IEEE

is to ensure the logical equivalence between the reduced auxiliary models and the original SPN by preserving liveness and boundedness. In the dynamic phase, some parameters of the auxiliary SPN models are iteratively tuned, using a mean delay equivalence relationship, until the models are statistically equivalent.

This paper is organized in the following way. Section 2 gives some preliminary definitions. Section 3 describes a Petri net model of some parallel communications software. Section 4 describes the proposed iterative reduction technique. Section 5 describes four decompositions of the software, with results shown in Section 6. Section 7 concludes the paper.

2. Preliminaries

A formal definition of the PN used in this paper is

Definition 2.1. A *PN* is a quadruple of K places $P=\{p_1, p_2, ..., p_K\}$, M transitions $T=\{t_1, t_2, ..., t_M\}$, arcs $A \subset \{P \times T\} \cup \{T \times P\}$ and an initial marking $\mathbf{M}_0=(n_1(0), n_2(0), ..., n_K(0))$:

$$PN = (P, T, A, \mathbf{M}_0) \tag{2.1}$$

The set of reachable markings from initial marking $\mathbf{M}_0$ is denoted by $R(\mathbf{M}_0)$. The sets of input and output places for transition t are denoted by ${}^\bullet t$ and $t^\bullet$ respectively, and the input and output transitions for place p are similarly ${}^\bullet p$ and $p^\bullet$.

Definition 2.2. A PN is

- *live* if every transition in the net is ultimately firable by progressing through some firing sequence for every $\mathbf{M}$ in $R(\mathbf{M}_0)$;
- *k-bounded* or simply *bounded* if the number of tokens in each place does not exceed a finite number k for any marking in $R(\mathbf{M}_0)$;
- a *marked graph* (*MG*) if for all $p \in P$, $|{}^\bullet p| = |p^\bullet| = 1$. That is, each place has exactly one input and exactly one output transition.

Definition 2.3. A *stochastic Petri net* has firing rates $\mu = \{\mu_1, \mu_2, ..., \mu_M\}$ for the transitions. The time to fire is exponentially distributed with mean μ_i^{-1} for t_i. A *generalized stochastic Petri net* has timed transitions (with firing rates) and immediate transitions (with probabilities) [11].

3. A Petri Net Model of a Parallel Communications Software System

Fig.3.1 shows the software tasks and the data flows between them. A message originates at the Sender Side User task, and is conveyed, following the arrows, to the Receiver Side User task. The message is processed by two tasks implementing the ISO Session and Transport layers (e.g. [14]) at each end, and is conveyed between users and layer tasks by "Courier" tasks which are active buffers for a single data unit.

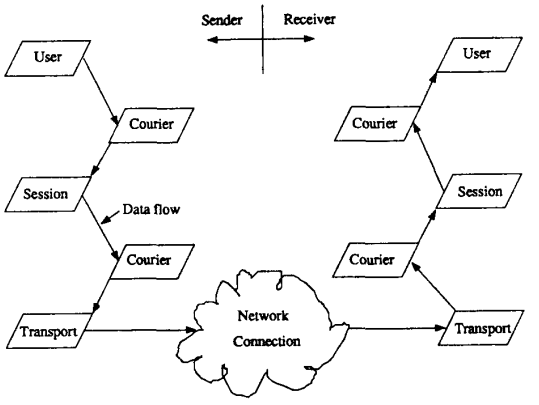


Figure 3.1 Tasking Model of Courier Protocol Software

The detailed stochastic Petri net model for one-way data transfer is given in Fig.3.2. The transport level fragments data, which is modelled by having two paths from place p_{13} to p_{33}. The path via t_8 with probability $q_1/(q_1+q_2)$ creates and carries fragments before the last one, at rate λ_{frg}, to p_{33} where acknowledgements are generated, but no data is delivered to higher layers. The path via t_9 with probability $q_2/(q_1+q_2)$ carries the last fragment of each TSDU at rate λ_{lsp} ('last packet') to p_{34} where it generates an acknowledgement and causes a data token to be delivered upwards via t_{27}. There is $(1+q_1/q_2)$ fragments (data TPDUs) per TSDU. The transport window size N is represented by the number of initial tokens in p_{14}. The transport buffer space M is represented by the number of initial tokens in p_{17}.

The software was implemented on a Sun workstation and on a VME bus-based multiprocessor [6]. The timed transitions in the model are labelled by rate parameters $\{r_i\}$ representing execution times measured on the workstation, with 1000-byte messages and no fragmentation. The measured processing time in seconds for 5000 messages was $\tau_1, \tau_2, \ldots, \tau_{10}$, with values 0.57, 4.97, 1.09, 10.37, 4.29, 0.39, 0.68, 2.88, 3.45 and 1.25. Then the rate parameters are $r_i = 5000/\tau_i$

The throughput rate λ from the users' point of view equals the rate of 'last packets' at the transport level, λ_{lsp}, which go through the path from t_9 tp p_{34}. The probability that task x is idle will be denoted by P_x, and is the probability of a token in the place labelled by the name x, e.g., for x=*transp*1, p_{12} is labelled transp_task1 and $P_{transp1} = Prob\{p_{12}$ *is marked with 1 token*$\} = Prob\{$*transport task 1 is idle*$\}$. Similarly define $P_{transp2}$ for p_{32}; P_{sess1} and P_{sess2} using p_6 and p_{41}; and P_{send}, P_{recv} using p_1 and p_{46}.

These measures were computed for various values of window size N, transport space M, and the ratio q_1:q_2. For a small window size and transport space, exact solutions are given in Table 3.1. The values for M=2

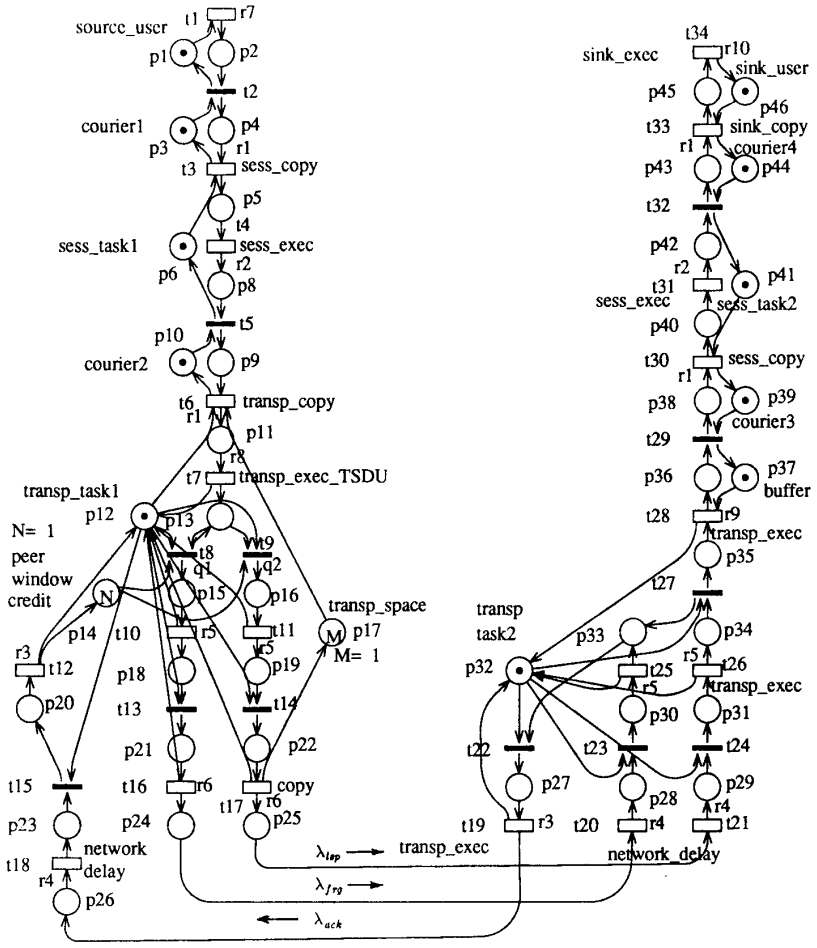


Figure 3.2 Complete GSPN Model of Courier Protocol Software

Table 3.1 Exact Performance Measures

N	Performance indices	Fragmentation ratio (q_1:q_2) 1:1	2:1
1	$\lambda_{lsp}=\lambda$	74.346664	50.432453
	P_{send}	0.010111	0.006859
	P_{recv}	0.981413	0.987392
	P_{sess1}	0.008476	0.005749
	P_{sess2}	0.926096	0.949869
	$P_{transp1}$	0.785584	0.796354
	$P_{transp2}$	0.788707	0.802406
2	$\lambda_{lsp}=\lambda$	120.372086	92.875839
	P_{send}	0.016371	0.012631
	P_{recv}	0.969907	0.976781
	P_{sess1}	0.013722	0.010588
	P_{sess2}	0.880294	0.907652
	$P_{transp1}$	0.652847	0.624967
	$P_{transp2}$	0.657896	0.636110

were almost identical to those for M=1, so they are not shown.

The size of the Markov chain state spaces of the complete model for various values $q(N, M)$ are given in Table 3.2, where the first (heading) row are values of (N,M), the second row contains the number of tangible markings, and the third row contains the number of vanishing markings. As the window size N increases, the state space of the complete GSPN model increases rapidly. When N=3, the total number of states gets near to one million. Then an approximate decomposition technique is required.

Table 3.2 State Spaces of the Complete Model

(1, 1)	(1, 2)	(1, 3)	(2, 1)	(2, 2)
11,700	20,700	29,700	84,600	143,550
17,310	30,990	44,670	143,820	245,250

4. Principles of Decomposition, Aggregation, and Parameter Derivation and Tuning

The proposed technique uses the concept of "divide and conquer". It first divides a large original GSPN into several subnets, then constructs a set of auxiliary GSPN models, each containing some unique subnet(s) from the original GSPN and their aggregated complementary subnets. Next, these auxiliary models are solved iteratively based on statistical relationships between the subnets and their aggregates.

4.1 Decomposition

The first step is to construct a set of L auxiliary Petri nets $APN_1,...,APN_L$, each much smaller than the original model PN. To do this, PN is divided into L subnets $SN_1,...,SN_L$ as shown in Fig.4.1(a), and for each SN_i a corresponding aggregated subnet sn_i is found (by means which will be described shortly). Then APN_i consists of some combination of original subnets SN_i and aggregated subnets sn_j. One approach is to have just one SN_i in each APN_i, as in Fig.4.1(b). The aggregation of SN_i to sn_i retains the subnet liveness and input-output properties, and the arcs connecting the subnets, so the interconnection of subnets in APN_i is exactly the same as in PN.

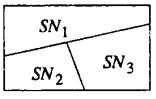


Figure 4.1 (a) An Original PN Model

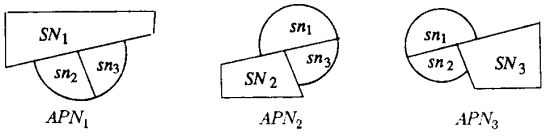


Figure 4.1 (b) Structure of Auxiliary Models APN_i's

The informal goal of the approximation is that, when the parameters of the sn_i are tuned and all the APN_i are solved, the delays in the subnet sn_i in each APN_j will approximately match the corresponding delays in SN_i in APN_i and also in PN, the throughputs of the interconnections will be roughly equal in all APN_j and in PN, and the probability distributions of tokens in places within SN_i in APN_i will closely match the distributions for the same places in PN. The APN_i should be constructed so as to approach this goal while solving the smallest possible auxiliary models, for a given level of decomposition.

In Fig.4.1 there is exactly one auxiliary model per subnet. It is also possible and sometimes useful (an example will be given in Section 5.2) to place several subnets SN_i into one APN_j. The only rule is that each SN_i should appear in just one APN_j, and its solution is used to tune the corresponding sn_i in all the other auxiliary models. Some parts of PN may be unaggregated; they appear in every auxiliary model, unchanged.

The parameter tuning process is iterative and heuristic. The $\{APN_i\}$ are derived and initialized and then solved in a cycle; in every APN_i the subnet sn_j is tuned using results from the APN which contains SN_j.

There are two stages to the creation of the decomposition model—the definition of the structure of the sn submodels and the tuning of their parameters.

4.2 Subnet Aggregation

The structure of the sn submodels is determined by recursively applying the following reduction rules.

Rule 1. A flow-conserving single-input-single-output (SISO) subnet has one input arc (to a place), an output arc (from a transition) and is such that, if one token is placed in the input place, all subsequent firing sequences include exactly one firing of the output transition. Thus for each token going in, one will come out. (Note that there may be tokens representing resources inside the subnet. However, they do not leave.) Such a subnet is aggregated into a place-transition pair as shown in Fig.4.2.

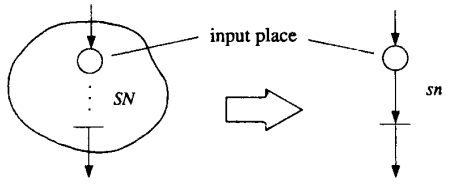


Figure 4.2 Rule 1: Aggregation of a SISO Subnet

Rule 2. A marked graph (MG) subnet will be aggregated into a single equivalent transition and a set of unchanged input places by the method described in [10].

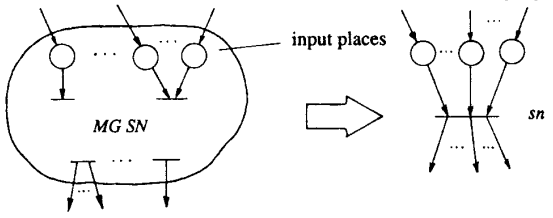


Figure 4.3 Rule 2: Aggregation of a MG Subnet

Rule 3. *Redundant structure* constitutes net components which can be eliminated without changing the reachability graph of the whole net. They are eliminated. For a complete definition, refer to [2].

4.3 Mean Subnet Delay

An artificial "weighted delay" closely related to delay has been defined so that Little's result can be applied to Petri nets, even if tokens are not conserved when transitions fire. For an original subnet SN weights w_i are attached to places as follows: each input

place has weight 1; for any transition the sum of output-place weights equals the sum on input places, and weights on output places are made equal to each other. The weighted token sum of SN is $N'_{SN}=\sum_{i\in SN} w_i \cdot n_i$ with mean value $\overline{N}'_{SN}$, where n_i is the number of tokens in place p_i. A weighted rate λ' of tokens traversing an arc is defined as the rate of flow of tokens times the weight of the place attached to the arc; λ'_{SN} is the total weighted throughput of subnet SN. Then the mean delay D_{SN} is defined by $D_{SN}=\overline{N}'_{SN}/\lambda'_{SN}$. Since input weights are 1, λ'_{SN} equals the input rate λ_{SN} of tokens to SN. If SN has a single input and output then D_{SN} is exactly the delay in seconds, and is not artificial. The same weighting rules are applied to aggregated sn's.

4.4 Parameter Derivation and Tuning Process

Once the auxiliary models are determined, the rate parameters of the aggregated transitions are found by an iterative tuning process.

The parameters of sn_i are derived from SN_i by relationships which can be stated for each reduction rule. For a given sn_i created by a combination of reductions, the parameter derivation can either be composed from the corresponding separate relationships, or can be derived for each of a series of intermediate reductions until sn_i is reached.

The Use of Constant Rates for Aggregated Transitions

In this work each aggregated transition has a constant rate parameter. In principle greater accuracy is attainable by making the rate a function of some markings, but we have found the constant rate calculation to be more robust and to give acceptable errors (under 5%).

Derivation of Iterative Formulas for the Rates of Aggregated Transitions

To derive the iterating formulas for the rates, the mean weighted delays of each SN and the corresponding sn are matched. The following derivations are related to the first two reduction rules in Section 4.2 (the third rule gives no derivation of parameters).

Consider first the aggregation of a single-input-single-output subnet. In this case, for sn the mean delay is $\overline{n}_{sn}/\lambda_{sn}$, the ratio of mean tokens in p_a to mean subnet throughput. For SN the weighted number of mean tokens must be used, giving the mean delay $\overline{N}'_{SN}/\lambda_{SN}$. Then set $\overline{n}_{sn}/\lambda_{sn}=\overline{N}'_{SN}/\lambda_{SN}$, substitute $\lambda_{sn}=\mu\cdot[1-P_a(0)]$, and express it in iterative form, to obtain

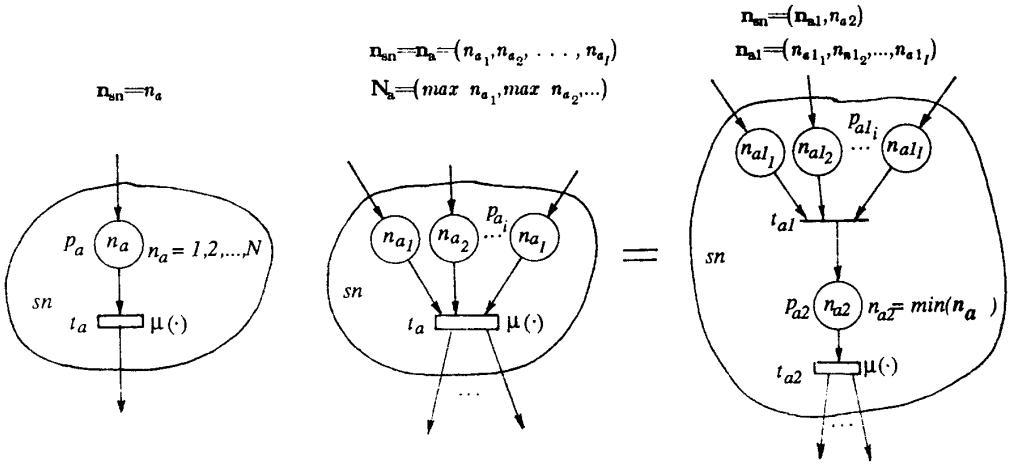
$$\mu^{(k+1)}=\frac{\overline{n}_{sn}^{(k)}}{1-P_a^{(k)}(0)}\cdot\frac{\lambda_{SN}^{(k)}}{\overline{N}'^{(k)}_{SN}} \qquad (4.1)$$

where k is the k^{th} iteration.

Now consider the aggregation of a marked graph subnet. For ease of calculating the throughput of the aggregated subnet sn in Fig.4.4(b), its equivalent form is constructed as in Fig.4.4(c). (Note that if there exist some timed join-type transitions outside the sn sharing some common input places with t_a in (b), i.e., in conflict with some outside transitions, they also need to be split into two transitions as t_a splits in (c).) The notations, referring to Fig.4.4(c), are as follows:

$P_{a2}(n_{a2})$: the marginal probability of p_{a2} containing n_{a2} tokens;

w_{a1_i}: the weight of input place p_{a1_i}, defined by V: $w_{a1_i}=1$;



(a) The Simplest SISO sn (b) Aggregated MG sn (c) Its Equivalent Form

Figure 4.4 Aggregated Single-Input-Single-Output Subnet and Marked Graph Subnet

w_{a2}: the weight of p_{a2}, and $w_{a2}=\sum_i w_{a1_i}=I$;

$\overline{n}_{a1_i}$: the mean number of tokens in place p_{a1_i};

$\overline{n}_{a2}$: the mean number of tokens in p_{a2};

$\overline{n}'_{sn}=w_{a2}\cdot\overline{n}_{a2}+\sum_{i=1}^{I} w_{a1_i}\cdot\overline{n}_{a1_i}$: the mean number of weighted tokens in aggregated subnet sn;

$\lambda'_{sn}=\mu\cdot I\cdot[1-P_{a2}(0)]$: the weighted throughput of aggregated subnet sn;

$d_{sn}=\overline{n}'_{sn}/\lambda'_{sn}$: the mean delay of aggregated subnet sn.

Setting $d_{sn}=D_{SN}$ and proceeding as before gives, similarly to (4.1), the rate in iterative form as:

$$\mu^{(k+1)} = \frac{\overline{n}'^{(k)}_{sn}}{I\cdot[1-P^{(k)}_{a2}(0)]}\cdot\frac{\lambda'^{(k)}_{SN}}{\overline{N}'^{(k)}_{SN}} \qquad (4.2)$$

Parameter Tuning Process

The tuning procedure can be written in algorithmic form. In initializing the auxiliary models, arbitrary positive real numbers can be chosen for all aggregated transition rates, but a better choice is to use average rates as specified in the algorithm below. If there is a sequence of cases, such as those indexed by the transport window size N in our model, the initial aggregated transition rates can be chosen as $\mu_i^{(0)}\{N\}=\mu_i\{N-1\}$ where $\mu_i\{N-1\}$ is the result for the previous case $N-1$, and the bracket notation $\{N\}$ says that N is a parameter rather than a variable/argument.

For ease of writing the method, the iteration uses simultaneous substitutions. That is, solve all auxiliary models in one iteration, calculate all aggregated transitions' rates for next iteration, substitute new rates into all auxiliary models, and repeat. This is analogous to Jacobi's method in solving a system of linear equations. For quick convergence, one may use successive substitutions for the iteration, an analogy to Gauss-Siedel's method for solving a system of linear equations. A variety of stopping criteria may be used, including the relative percentage change of any tuned transition rate between two adjacent iterations falling into some prescribed threshold (used here), and the percentage difference in throughput of the two models.

Algorithm.

Order all auxiliary models $\{APN_i\}$ *from* $i=1$ *to* L;
{*Choose proper initial rates for the aggregated transitions in each* APN_i}
for $i=1$ *to* L *do begin*
  {*Define* AT_i *to be the set of aggregated transitions in* APN_i.}
  $\forall t_j\in AT_i$: *set* $\mu_j^{(0)}=\tilde{\mu}_j$, *where* $\tilde{\mu}_j$ *is the average rate of some timed transitions in the SN of another* APN_i *that the aggregated transition* t_j (*with rate* μ_j) *is aggregated from;*
end;
$k:=0$;
do {*iteration* k}
  for $i=1$ *to* L *do*
  begin
    $\forall t_j\in AT_i$: *set* $\mu_j=\mu_j^{(k)}$;
    Solve APN_i *for delays, probabilities in* (*4.1*), (*4.2*) *used in deriving corresponding rates related to other auxiliary models;*
  end;
  for $i=1$ *to* L *do*
    Calculate all $\mu_j^{(k+1)}$*'s for* $t_j\in AT_i$ *by* (*4.1*) *or* (*4.2*);
  $k:=k+1$;
Until (*The accuracies of all* $\mu^{(k)}$*'s meet the stopping criterion.*)

5. Four Levels of Decompositions

In this section four levels of decompositions are applied to the complete GSPN model. Two of them, Decompositions B and C, are described in some details. The purpose is to show various practical aspects of the technique and to demonstrate its effectiveness. The four decompositions are described in the order of decreasing state space size and increasing number of auxiliary models. They differ in their treatment of the two large, densely interconnected blocks at the transport level. In all these cases the transport space value is $M=1$.

Decomposition A—Two Auxiliary Models

Two auxiliary models A-1 and A-2 are constructed. The simplest subnets to approximate are the portions of each protocol stack above the transport level, since each of these is a marked graph subnet. They become SN_1 and SN_2 and the lower part of the model is left untouched. The corresponding aggregated subnets sn_1 and sn_2 consist of aggregated transitions t_5 and t_{29} in Fig.3.2, with rates which are tuned using (4.2).

Decomposition B—Two Simpler Auxiliary Models

More subnets are to be aggregated in this decomposition, with SN_1 being retained and new subnets SN_2—SN_8 defined as shown in Fig.5.2. However there are still only two auxiliary models, the left-hand model B-1 having SN_1—SN_5 and the other (B-2) having SN_6—SN_8.

The reduction process which gives the auxiliary models requires multiple steps. For the left-hand auxiliary model B-1 in Fig.5.1,

1) aggregate the three marked graph subnets SN_6, SN_7, SN_8 into sn_6, sn_7, sn_8 as shown in Fig.5.2;
2) remove redundant place p_{32} and its input/output arcs, because it is always marked.

For the right-hand auxiliary model B-2 in Fig.5.1,

1) aggregate the five marked graph subnets SN_1—SN_5 in auxiliary model B-1 of Fig.5.1 separately into subnets

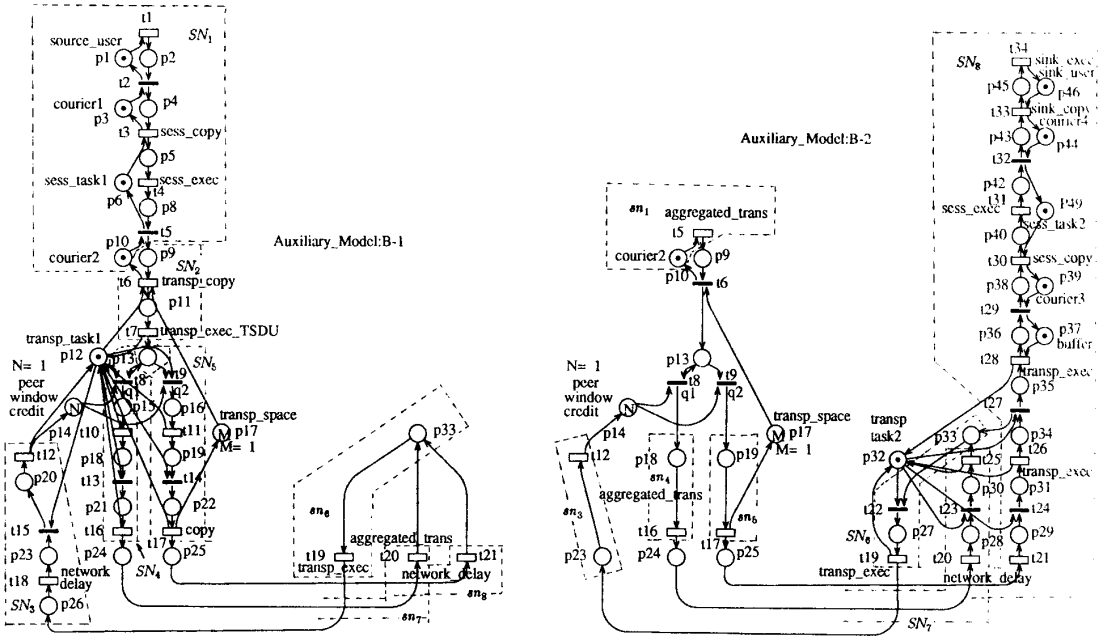


Figure 5.1 Decomposition B—Two Simpler Auxiliary Models B-1 and B-2

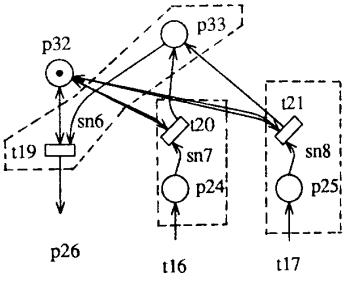
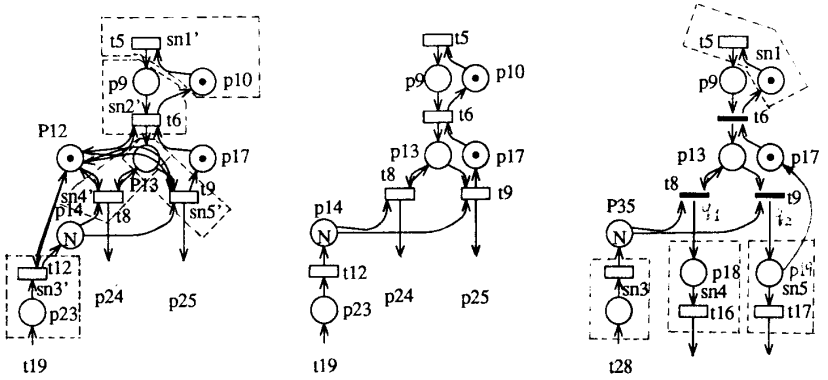


Figure 5.2 Intermediate Result After Aggregating Three MG Subnets in Auxiliary Model B-2

sn_1'—sn_5' in Fig.5.3(a);

2) remove redundant place p_{12} together with its input/output arcs in Fig.5.3(a), because it is always marked. This leads to Fig.5.3(b);

3) some further changes were made to make it easier to obtain quantities for the calculation. These are not necessary and do not involve the reduction rules quoted earlier. Separate timed transition t_8 in Fig.5.3(b) into an immediate transition t_8 and a timed transition t_{16} as in Fig.5.3(c) for easy measurement of marginal probability of place p_{18}. Similarly, separate timed transition t_9 in Fig.5.3(b) into two transitions t_9



(a) Aggregate 5 MG SNs (b) Remove Redundant Place (c) Separate Timed Transitions

Figure 5.3 Step Reductions of MG Subnets in Auxiliary Model B-1 of Decomposition B

and t_{17} in Fig.5.3(c). Next, convert timed transition t_6 in Fig.5.3(b) into an immediate transition as in Fig.5.3(c) to reduce tangible markings. (The conversion is not a necessity.) The lost delay time is distributed to other four timed aggregated transitions.

For window size N the tuned rates will be designated as $\mu_{19}\{N\}$, $\mu_{20}\{N\}$, $\mu_{21}\{N\}$ in auxiliary model B-1, $\mu_5\{N\}$, $\mu_{12}\{N\}$, $\mu_{16}\{N\}$, $\mu_{17}\{N\}$ in auxiliary model B-2. These rates are obtained iteratively by matching mean delays (d_{sn}=D_{SN}) for N=1,2,3,4. One of the derived formulas is:

$$\mu_{20}\{N\} = \frac{\overline{n}_{24}}{1-P24(0)}\Big|_{B\text{-}1} \cdot \frac{1}{D_{SN7}}$$

$$\text{where } \frac{1}{D_{SN7}} = \frac{\mu_{25} \cdot P30(n>0)}{\sum_n n \cdot [P24(n)+P28(n)+P30(n)]}\Big|_{B\text{-}2}$$

and $\overline{n}_{24}$ is the mean number of tokens in place p_{24}, $P30(n>0)$ is the marginal probability of place p_{30} being nonempty, and "$|_{B\text{-}1}$" denotes values from auxiliary model B-1. The others are similar.

Decomposition C—Three Auxiliary Models

This decomposition has three auxiliary models C-1, C-2 and C-3. The bottom model C-2 includes both transport blocks. Aggregated transitions are $\{t_6\}$, $\{t_5, t_{29}\}$, $\{t_{28}\}$ for C-1, C-2 and C-3 respectively. Four subnets are aggregated here. In constructing the upper-left model C-1, the single-input-single-output part of Fig.3.2, starting from p_9 and extending till the receiver task, is aggregated. Similarly, in constructing the upper-right model C-3, the part starting from p_{37} and extending back to the sender task is aggregated. In constructing the middle model C-2, two subnets (p_{10} upwards and p_{36} upwards) are aggregated separately.

Decomposition D—Four Auxiliary Models

This decomposition has four auxiliary models D-1, D-2, D-3 and D-4. It can be viewed as the combination of Decompositions B and C. Decomposition D provides a submodel for each transport block separately. The iterative calculation is similar to B.

6. Experimental Results

The average user throughput λ and average probability $P_{transp1}$ yielded from the exact results and four Decompositions A—D are plotted in Fig.6.1 and Fig.6.2, where window size N runs to 2 for the exact results; to 3 for Decomposition A; to 4 for Decomposition B; to 5 for Decomposition C; and to 6 for Decom-

Table 6.1 State Spaces of Various Auxiliary Models from Four Decompositions

Auxiliary Models	Types of Markings	Transport Window Size N						
		1	2	3	4	5	6	7
A-1	tangible	780	5640	27960				
	vanishing	836	7133	37847				
A-2	tangible	1560	11280	55920				
	vanishing	1476	13160	71508				
B-1	tangible	285	1165	4710	13005			
	vanishing	227	1193	4387	12471			
B-2	tangible	720	4410	19050	65250			
	vanishing	848	6156	28046	98360			
C-1	tangible	15	15	15	15	15		
	vanishing	8	8	8	8	8		
C-2	tangible	104	752	3728	14512	47632		
	vanishing	56	550	3058	12552	42188		
C-3	tangible	30	30	30	30	30		
	vanishing	16	16	16	16	16		
D-1	tangible	15	15	15	15	15	15	15
	vanishing	8	8	8	8	8	8	8
D-2	tangible	38	182	628	1734	4126	8800	17246
	vanishing	10	62	250	738	1804	3878	7588
D-3	tangible	48	294	1270	4350	12594	32092	73980
	vanishing	37	279	1268	4414	12845	32753	75396
D-4	tangible	30	30	30	30	30	30	30
	vanishing	16	16	16	16	16	16	16

position D, and M=1. It's seen that they overlap or nearly overlap for all common N—an indication of good approximation.

The size of the state space of all the auxiliary models is given in Table 6.1. Using more auxiliary models greatly decreases their sizes. However the sizes of the two (largest) auxiliary models for Decompositions B and D are not well balanced. Due to exponential state space growth of the net, it is important to minimize the size of the largest auxiliary model. This can be achieved by balancing the sizes of all auxiliary models used in decomposition. Because of the smaller state spaces, C and D can be solved for larger N than A and B.

The accuracy of the converged values can be assessed against the exact results for N=1,2. For A and C the error in throughput and $P_{transp1}$ was under 0.001%, while for B and D it was between 0.01% and 0.5%. B and D used reductions on the bottom half of the model in Fig.3.1 to obtain smaller state spaces, but paid a severe price in accuracy. In particular C has about the same tangible state space size as B but is much more accurate. In D also the various auxiliary models produced different throughput values when N was large; the values were averaged in the iteration. This may be because the *SN*s contain in infinite servers which are being approximated by constant-rate servers in the *sn*s.

A variety of iteration convergence behaviors were observed. A, B and C settled down quickly (in 2-3 iterations) for all N examined. D tended to oscillate as N was increased, but still settled within 4 iterations. A

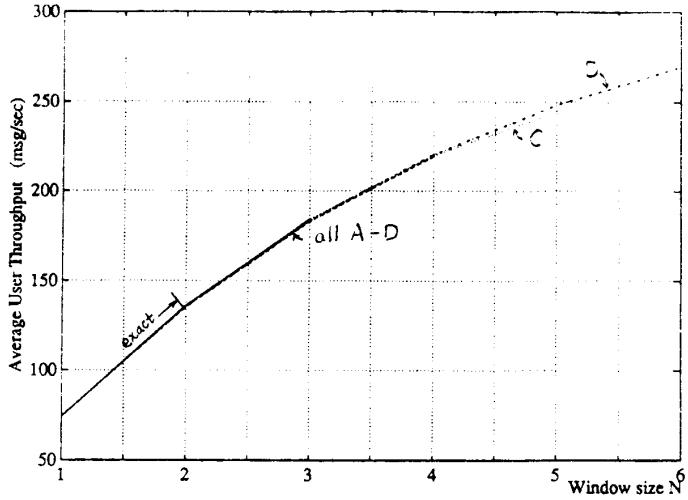


Figure 6.1 Average User Throughput From Four Decompositions A—D

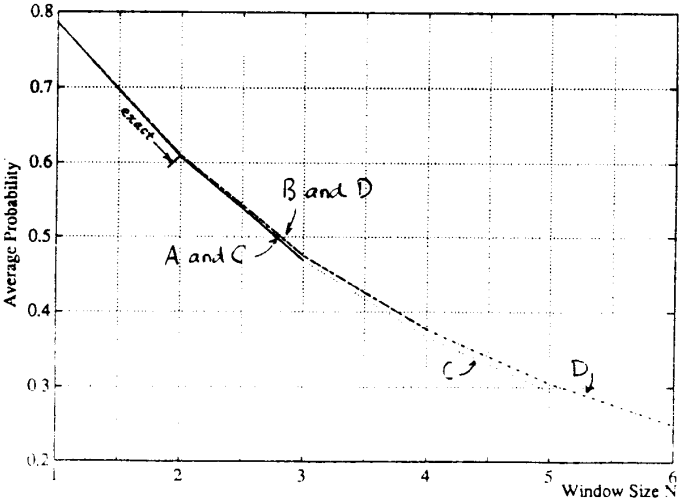


Figure 6.2 Average Probability $P_{transp1}$ From Four Decompositions A—D

simple relaxation technique was applied to iteration processes for large N that reduces oscillation by 2 to 3 orders of magnitudes at later stage of iteration processes.

7. Conclusions

In this paper, an iterative GSPN solution technique for large models has been briefly described. It consists of two parts: PN reduction and iterative aggregation. It can be viewed as a generalization of the concept of "surrogate delay server" from queueing networks to GSPNs, which also allows multiple arc connections between an aggregated subnet and its complementary part, several aggregated subnets in one auxiliary model, and several auxiliary models. With these extensions, the new technique is useful on rather complex practical systems.

This approximate technique has been applied to the analysis of a data communication protocol software system. The example shows a variety of ways of obtaining reduced subnets and demonstrates the state-space reduction achievable. For example for window size N=2 the original Markov chain has 84,600 states while decomposition D has four models, the largest having 294 states. The solvable size of problem was extended from N=2 to N=7. Accuracies better than 0.5% were achieved for N=2; these appear to be acceptable.

An important question for larger models is, can the size of the auxiliary models always be kept small by using enough of them? We have shown that this can be achieved to some extent. Another question regards the value of using marking-dependent rates.

Acknowledgements

This research was supported by the University Research Incentive Fund of the Province of Ontario, and by the Telecommunications Research Institute of Ontario through the Telecom Software Methods project. The use of the GreatSPN Petri-Net solving software of the University of Turin, Italy is gratefully acknowledged.

References

[1] Ammar, H.H. and S.M.R. Islam, "Time Scale Decomposition of A Class of Generalized Stochastic Petri Net Models," *IEEE Transactions on Software Eng.*, Vol. 15, No. 6, June 1989, pp. 809-820.

[2] Berthelot, G. and Lri-Ile, "Checking Properties of Nets Using Transformations," *Advances in Petri Nets, Lecture Notes in Computer Science*, Vol. 222, 1985, pp. 19-40.

[3] Berthelot, G., Cham-Ile and Lri, "Transformations and Decompositions of Nets," *Advances in Petri Nets, Lecture Notes in Computer Science*, Vol. 254, Part 1, 1986, pp. 359-376.

[4] Ciardo, G. and K.S. Trivedi, "Solution of Large GSPN Models," *The First Int'l Conf. on the Numerical Solution of Markov Chains*, Raleigh, North Carolina, Jan. 8-10, 1990, pp. 603-626.

[5] Courtois, P.J., "Decomposability: Queueing and Computer Applications," *Academic Press, New York*, 1977.

[6] Franks, R.G., "Experience in Concurrency in OSI Communications Software: A Comparative Performance Study," *Master's Thesis*, Dept. of Systems and Computer Eng., Carleton University, July 1989.

[7] Giglmayr, J., "Analysis of Stochastic Petri Nets by the Decomposition of the Transition Rate Matrix," *NTZ Archiv* (Germany), Part 1: Vol. 9, No. 5, May 1987, pp. 115-120; Part 2: Vol. 9, No. 6, June 1987, pp. 147-152.

[8] Hatono, I., K. Yamagata and H. Tamura, "Modeling and On-Line Scheduling of Flexible Manufacturing Systems Using Stochastic Petri Nets," *IEEE Trans. on Software Eng.*, Vol. 17, No. 2, Feb. 1991, pp. 126-132.

[9] Jacobson, P.A. and E.D. Lazouska, "Analyzing Queueing Networks with Simultaneous Resource Possession," *Communications of the ACM*, Vol. 25, No. 2, Feb., 1982, pp. 142-151.

[10] Li, Y. and C.M. Woodside, "Iterative Decomposition and Aggregation of Stochastic Marked Graph Petri Nets," *The 12th Int'l Conference on Application and Theory of Petri Nets*, Aarhus, Denmark, June 26-28, 1991, pp. 257-275.

[11] Marsan, M.A. and G. Balbo, "A Class of Generalized Stochastic Petri Nets for the Performance Evaluation of Multiprocessor Systems," *ACM Trans. on Computer Systems*, Vol. 2, No. 2, May 1984, pp. 93-122.

[12] Molloy, M.K., "Performance Analysis Using Stochastic Petri Nets," *IEEE Trans. on Computers*, Vol. C-31, No. 9, Sept. 1982, pp. 913-917.

[13] Song, J-S., S. Satoh and C.V. Ramamoorthy, "The Abstraction of Petri Net," *Telcon 87*, Vol. 2 of 3, August 25-28, 1987, Seoul, Korea, pp. 467-471.

[14] Stallings, W., "Handbook of Computer-Communications Standards. 2nd ed." *Macmillan Publishing Com.*, New York, 1990.

[15] Suzuki, T., S.M. Shatz and T. Murata, "A Protocol Modeling and Verification Approach Based on a Specification Language and Petri Nets," *IEEE Trans. on Software Eng.*, Vol. 16, No. 5, May 1990, pp. 523-536.

[16] Woodside, C.M., "Throughput Calculation for Basic Stochastic Rendezvous Networks," *Performance Evaluation*, No. 9, 1988/89, pp. 143-160.