From W. H. Sanders and J. F. Meyer, "Reduced Base Model Construction Methods for Stochastic Activity Networks," in *IEEE Journal on Selected Areas in Communications*, special issue on Computer-Aided Modeling, Analysis, and Design of Communication Networks, vol. 9, no. 1, Jan. 1991, pg. 25-36.

REDUCED BASE MODEL CONSTRUCTION METHODS FOR STOCHASTIC ACTIVITY NETWORKS*

W. H. Sanders† and J. F. Meyer‡

†The University of Arizona
Department of Electrical and Computer Engineering
Tucson, AZ 85721
whs@ece.arizona.edu
and
‡The University of Michigan
Department of Electrical Engineering and Computer Science
Ann Arbor, MI 48109
jfm@eecs.umich.edu

ABSTRACT

Computer-aided modeling has become an accepted technique for predicting the performance and dependability of computer-communication networks. Many model formalisms have been proposed for this purpose, and have achieved varying degrees of success. One particular model type, stochastic Petri nets, has received considerable attention during the last ten years. Several classes of models of this type now exist and, as evidenced by their use in specific evaluation studies, they are capable of representing the kind of complex behavior exhibited by computer-communication networks. However, a number of problems remain open, particularly those associated with the evaluation of large-scale systems. Specifically, the issue here is the size and complexity of the stochastic process, derived from the underlying net model, which serves as a "base model" for subsequent solution of the measures in question. If this base model constructed by standard means, e.g., it is identified with the marking behavior of the net, traditional methods of solution quickly become intractable for large systems, limiting their application to systems of only moderate complexity. This problem is addressed in the context of a particular class of stochastic Petri nets, stochastic activity networks (SANs), by developing base model construction methods that account for symmetries in SAN structure and are tailored to the variable (e.g., response time, time to failure, etc.) in question. We find that such a technique can yield dramatic reductions in state-space size while preserving stochastic properties required for practical means of solution. Moreover, unlike state aggregation methods that rely on explicit knowledge the more detailed state space, this technique permits direct construction of a reduced base model, thus avoiding size limitations associated with more traditional approaches to model simplification. The presentation includes both the theory required to establish the technique's validity and an illustration of its effectiveness in the evaluation of a CSMA/CD computer network.

This work was supported in part by the Digital Equipment Corporation Faculty Program: Incentives for Excellence and by the Office of Naval Research under Contract No. N00014-85-K-0531.

I INTRODUCTION

Large-scale computer-communication networks pose challenging evaluation problems due to their user requirements, complexity, and scale. In particular, demands for both high performance and high dependability are common. This, coupled with the fact that resources that are distributed, results in systems which exploit parallel processing. Furthermore, due to a balance of demands for high performance and dependability, such requirements typically call for systems which exhibit degradable performance in the presence of faults, i.e., between extremes of nondegraded performance (as would be exhibited if the system were fault-free throughout utilization) and fully degraded performance (system failure), the system is able to perform at intermediate levels which provide some benefit to the user.

These characteristics, namely parallelism, fault tolerance, and degradable performance, must be accounted for in the evaluation process, and hence place significant constraints on the types of models that may be used. Specifically, while queueing networks have been used extensively for the evaluation of computing systems [1, 2], their application to systems that exhibit complex parallelism (e.g., contention for multiple resources, blocking phenomena), fault tolerance, and degradable performance is not straightforward. Furthermore, queueing networks cannot easily represent fault-related behavior, since the probabilistic nature of their structures (e.g., branching probabilities) is fixed. On the other hand, stochastic Petri nets (SPNs) [3, 4] and subsequent extensions thereof are much better suited to the modeling of systems that exhibit such properties. In particular, the use of timed "transitions" or "activities" with probabilistic timing permits, via different interpretations of tokens, simultaneous representation of characteristics related to both performance and dependability.

Because of this capability, over the past ten years, considerable effort has been devoted to the development of SPN-type models for the evaluation of complex distributed systems (see, for example, the work documented in [5, 6]). More precisely, if the property being evaluated (e.g., throughput rate, response time, time to failure, etc.) is denoted by a random variable, say Y, an SPN-type model determines a stochastic process X which, in turn, is the basis for determining the probabilistic nature of Y. Accordingly, X, in this context, is referred to as a *base model* (that supports the solution of Y; see [7], for example). This research has been particularly fruitful and has resulted in the definition of formal model classes that are well-suited to generating base models for systems of the type described above. As a consequence, process models that, in the past, were either hand-

constructed Markov models or were determined by complex hand-written simulation code, could now be generated by simple, understandable stochastic network models. However, even with computer implementation, a number of problems remain open, particularly those associated with the evaluation of large-scale systems. When SPN-derived base models are constructed by standard means, traditional methods of solution quickly become intractable for large systems, limiting their application to systems of only moderate complexity.

In particular, when the solution methods to be employed are analytic, construction of an associated process model (base model) typically assumes that the state space is the set of all possible reachable, stable (or tangible) markings of the net. Likewise, when simulation methods are used to solve the model, each of the transitions or activities is taken to be a different event type in the associated simulation program. In the case of large-scale systems, the resulting base models can become intractably large in terms of state-space size, precluding analytic solutions, or the number of event types, precluding accurate simulations. Efforts aimed at solving this problem have been limited. In particular, prior to the work described below, neither the structure of the net nor the nature of the performance variable has been accounted for in attempts to construct simpler base models.

If stochastic extensions to Petri nets are to become a truly useful basis for system evaluation, there is a need for techniques that permit systems (represented in this manner) to be evaluated without intermediate generation of a detailed stochastic process or simulation program. In effect, what is needed are developments, analogous to product-form solutions for queueing networks, that permit evaluation of a large system to be based on special knowledge of the network model and performance variable. This knowledge can then be used to reduce the amount of information required to obtain the desired evaluation results or, in short, to "solve" the probabilistic nature of the performance variable.

To date, work toward this goal has proceeded in several directions. In particular, one approach has been to look for subclasses of nets that have the "product-form" property. While some results have been obtained [8, 9], they appear to be limited to very restricted subclasses of nets. Another approach has been through the use of colored tokens (e.g., [10]), high-level Petri nets [11], or regular nets [12, 13, 14]. A third approach has been to combine product form queueing networks and stochastic Petri nets in single model, taking advantage of the product-form property of the queueing networks whenever possible [15, 16]. A fourth approach, and the one we have taken, makes use of both the structure of the stochastic Petri net and the desired performance variables to generate a "smaller" stochastic

process that is both solvable and yields the desired information regarding the performance variable. The approach is based on the use of stochastic process lumping theorems, but makes use of the specified performance variable and structure of the network to select an appropriate notion of state and does not require generation of the complete marking space of the network. Furthermore, the generalized notion of state employed permits the determination of "activity-related" as well as "marking-related" behaviors. This permits one to ask questions regarding the number of activities that complete during an interval. Note that the approach taken in the use of colored and regular nets also makes use of the lumping idea, but does not take into account the nature of the performance variable.

Preliminary work with this goal was documented in [17], under the name "variable driven construction methods." In [17], we considered a single class of variables and a subclass of stochastic activity networks which consisted of one or more subsystems, each of which could contain, in turn, one or more replications of an even smaller subsystem. This work generalizes that of [17] in several significant ways. First, we consider many different variable types, capturing transient as well as steady-state system behaviors. Second, we consider a more general multi-level stochastic activity network structure, wherein the construction operations can applied any number of times to build arbitrarily complex stochastic networks. Finally, we present an algorithm for computer implementation to construct the "reduced" stochastic process without requiring the generation of the complete marking space.

The remainder of this paper is organized as follows. In the following section, we briefly review the basic definitions concerning stochastic networks and then define the types of variables to be employed in the construction process. These variables, as mentioned earlier, can be used to estimate both transient and steady-state system characteristics. The third section describes the construction operations used, contains theorems stating the validity of method, and presents a procedure for generating the "reduced" base model stochastic process for a given stochastic activity network and performance variable. This is followed by some examples which both illustrate the method and demonstrate its effectiveness in reducing the size of a state space.

II NETWORK-LEVEL MODELS AND VARIABLES

A Stochastic Activity Networks

Stochastic activity networks (SANs) [18, 19] incorporate features of both stochastic Petri nets and queueing models. Structurally, SANs have primitives consisting of *activities,*

places, input gates, and output gates. Activities ("transitions" in Petri net terminology) are of two types, *timed* and *instantaneous.* Timed activities represent activities of the modeled system whose durations impact the system's ability to perform. Instantaneous activities, on the other hand, represent system activities which, relative to the performance variable in question, complete in a negligible amount of time. Cases associated with activities permit the realization of two types of spatial uncertainty. Uncertainty about which activities are enabled in a certain state is realized by cases associated with intervening instantaneous activities. Uncertainty about the next state assumed upon completion of a timed activity is realized by cases associated with that activity. Places are as in Petri nets. The use of gates permits greater flexibility in defining enabling and completion rules.

The stochastic nature of the nets is realized by associating an *activity time distribution function* with each timed activity and a *probability distribution* with each set of cases. Generally, both distributions can depend on the global marking of the network. The execution of stochastic activity networks is discussed in detail in several places, including [20]. Informally, though, SANs execute in time through completions of activities that result in changes in markings. Activities complete some period of time after they are *activated*, depending on their activity time distribution function. More specifically, an activity is chosen to *complete* in the current marking based on the relative priority among activities (instantaneous activities have priority over timed activities) and the activity time distributions of *enabled* activities. A case of an activity chosen to complete is then selected based on the probability distribution for that set of cases. These two choices determine uniquely the next marking of the network, which is then obtained by executing the input gates connected to the input of the activity chosen and the output gates connected to the chosen case. This procedure is repeated by considering the activities enabled in the new marking.

In order to solve a stochastic activity network by analysis for a particular variable, a stochastic process must be constructed that both supports the variable and is solvable. Informally, SANs can be solved via analytic methods when all activity time distributions are exponential and activities are reactivated often enough to ensure that their rates depend only on the current state. Traditionally, for both SANs and other variants of stochastic Petri nets, this has been accomplished by taking the possible stable (tangible) markings as states (see for example, [20] where this is called the "marking behavior"). This approach results in a large state space that can support variables which can be written in terms of possible markings of the net. A second process, known as the "activity-marking behavior,"

was considered in [20]. In this case, the states of the process (known as *am-states* in that which follows) are taken to be pairs of the form (a, μ), where a is a timed activity and μ is a next stable marking which may be reached upon completion of a. More precisely then, the *activity-marking behavior* (or *am-behavior*) of a SAN is a stochastic process $(R, T, L) = \{R_n, T_n, L\}$ where R_n is the state reached after the nth timed activity completion, T_n is the time of the nth activity completion and L is the total number of the transitions of the process. This process, although typically larger than the marking behavior for a given SAN, can capture activity as well as marking behaviors (this fact is proved in [20]).

While the activity-marking behavior can conceptually support an extremely large class of performance variables, solution of the probability of occurrence of arbitrary sets of trajectories is often intractable. Accordingly, we opt for solutions based on less detailed process models which support more restricted classes of variables. To make these distinctions more precise, a stochastic process which supports a large class of variables (e.g., activity-marking behavior and marking behavior) is referred to as a *detailed base model*; a stochastic process which is constructed specifically to support a designated performance variable is a *reduced base model*.

B Example CSMA/CD Network Model

To illustrate the use of SAN primitives, and to provide a vehicle to explain the techniques developed in this paper, consider a variant of non-persistent CSMA/CD[1]. When a station has something to send, it waits an exponentially distributed amount of time before attempting the transmission. After this delay, it senses the channel. If the station detects an idle channel, it begins to transmit. One of two events will then occur: either 1) the channel was actually idle, so the transmission begins normally, or 2) a collision occurs, since another station had begun transmitting, but its signal had not propagated to the first station at the time the channel was sensed. If a collision occurs, it is cleared after an exponentially distributed amount of time. If the station detects a busy channel, it returns to the wait state and after the delay period once again attempts to gain access to the channel.

A stochastic activity network representing a single station connected to the channel is

[1]Note that the goal here is not to develop a detailed CSMA/CD model, but to illustrate reduced base model construction methods. Many other CSMA/CD models exist, including other SPN and SAN models. The only requirement for the use of the method is that activity times be exponentially distributed; hence its application to standard CSMA/CD would be relatively straightforward.

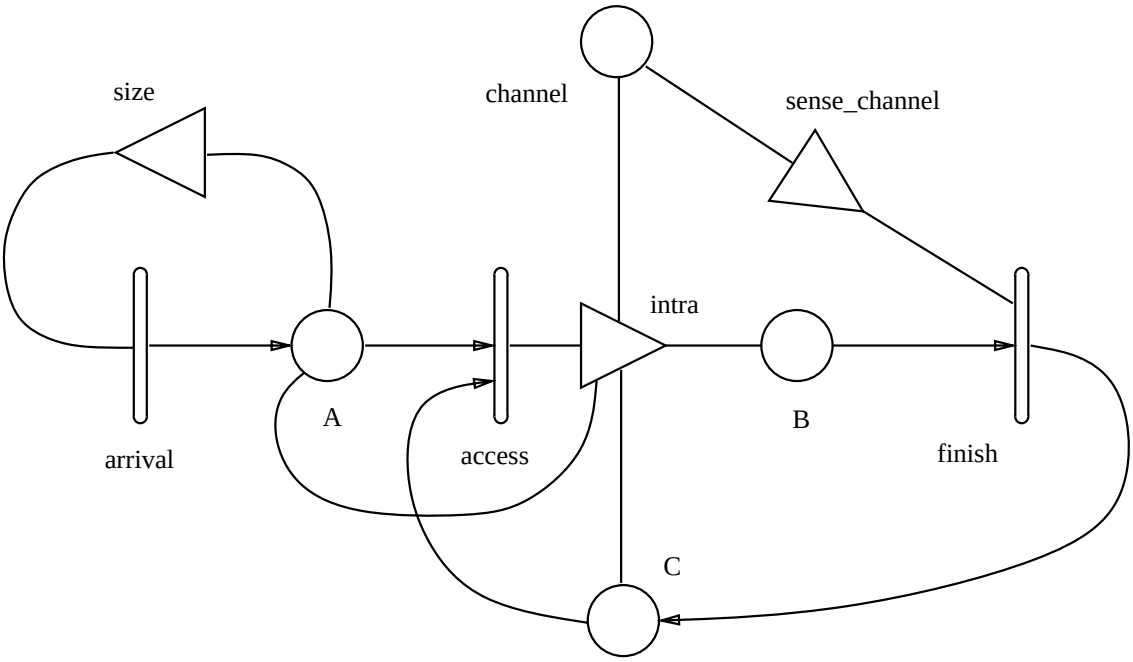


Figure 1: Station Submodel

given in Figure 1. The gate and activity parameter values for the model are given in Tables 1 and 2, respectively. Each completion of activity *arrival* represents the arrival of a message to the station queue. The station queue is represented by place A. The marking of place A represents the number of messages waiting to be transmitted. The finiteness of the queue is represented by gate *size*. Gate *size* specifies (via its predicate) that activity *arrival* is enabled only when the number of messages in the queue is less than the system's capacity.

Completion of activity *access* signals that the station is sensing the channel. This results in one of four outcomes, as stated above, which are implemented by gate *intra*. If the channel is idle (i.e. the marking of *channel* is 0), the marking of place *channel* is set to one to indicate the start of a transmission. In addition, the marking of B is set to one to indicate that a transmission is in progress for the station. If the channel is in use, but the signal has not received sufficient time to propagate (i.e. the marking of *channel* is 1), the marking of place *channel* is set to three to indicate that a corrupted message is now on the channel and a token is added to place A to indicate that a retransmission is necessary. If a corrupted message which has not been cleared is present (i.e. the marking of *channel* is 3), a token is added to place A to indicate that a retransmission is necessary and the station goes back

Gate	Type	Enabling Predicate	Function
size	input	$MARK(A) < 2$	identity

intra	output	–	if (MARK(*channel*)==0) {
			MARK(*channel*) = 1;
			MARK(*B*) = 1; }
			else if (MARK(*channel*)==1) {
			MARK(*channel*) = 3;
			MARK(*C*) = 1;
			MARK(*A*) = MARK(*A*) + 1; }
			else if (MARK(*channel*)==2) {
			MARK(*C*) = 1;
			MARK(*A*) = MARK(*A*) + 1; }
			else if (MARK(*channel*)==3) {
			MARK(*C*) = 1;
			MARK(*A*) = MARK(*A*) + 1; }

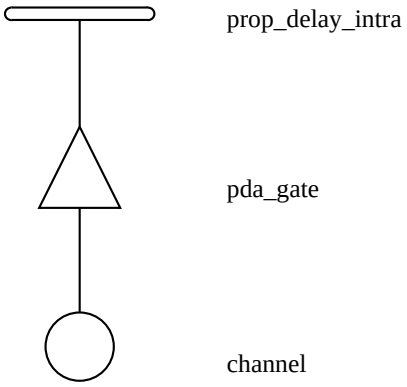
sense_channel	input	MARK(*channel*)==2) || MARK(*channel*)==3	MARK(*channel*) = 0;

Table 1: Gates for Station Submodel

Activity	Distribution Type	Parameter (Rate)
arrival	exponential	varied
access	exponential	10 (normal prio. station) 100 (high prio. station)

finish	exponential	if (MARK(*channel*)==2)
		rate = 1
		else
		rate = 5

Table 2: Activity Parameters for Station Submodel



Gate	Type	Enabling Predicate	Function
pda_gate	input	MARK(*channel*)=1	MARK(*channel*)=2;

Activity	Distribution Type	Parameter (Rate)
prop_delay_intra	exponential	20

Figure 2: Network Submodel

into the wait state. If the transmission of a message that has been active long enough to propagate to all stations is in progress (i.e. the marking of *channel* is 2), a token is returned to place A to indicate that no transmission is possible at this time and the marking of place C is set to one to indicate that that station is again entering the wait state.

The time required to transmit messages (either corrupted or not) is controlled by activity *finish*, whose activity time distribution depends on the marking of place *channel*. This distribution depends on whether a propagated or corrupted message is on the channel. If a propagated message is on the channel, the distribution reflects the message transmission time. If a corrupted message is on the channel (detected by gate *sense_channel*), the distribution reflects the time to clear the collision. The completion of a transmission occurs when activity *finish* completes. When this happens, a token is added to place C.

The time to propagate a message is represented by the activity *prop_delay_intra* in the network submodel. The network submodel is shown in Figure 2. Gate *pda_gate* senses the presence of an unpropagated message (via its predicate) and, when it holds, activates activity *prop_delay_intra*. After a prescribed (exponentially distributed) propagation delay, activity *prop_delay_intra* completes, signaling that the propagation of the signal to all stations has occurred. The subsequent execution of gate *pd_gate* sets the marking of place

channel to 2 thus indicating the presence of a propagated message.

These stochastic activity networks will be used in the following sections to illustrate the state-space savings that can be acheived by using reduced base model construction methods. Before this can be done, however, we need to review the variables that will be used in the modeling process.

C Performance Variable Specification

As outlined in the introduction, our method makes use of knowledge regarding the desired performance variables as well as the structure of the net to aid in the construction process. In order to make this possible, it is necessary to formally specify the "types" of performance variables that may be considered. Previous work regarding "reward models" [21] (and associated "reward variables") provides an instructive example in this regard. Informally, a reward model consists of a stochastic process and a "reward structure". The reward structure relates possible behaviors of the process to a specified performance variable. Typically, this is done by associating a "reward rate" with each state, its interpretation being the rate at which reward accumulates while the process is in that state. In this case, the performance variable is taken to be the reward accumulated over some utilization interval (either finite or infinite). By associating different reward rates to states, one can construct performance variables with many different interpretations.

The network-level structure of a stochastic activity network (or other stochastic Petri net extension) allows us to define the reward structure at the network level, rather than the state level. This approach has several distinct advantages over the state-level approach when used with SANs. First, the assignment of rewards and interpretation of solutions are more natural, because it is done at the level at which the modeler thinks. Second, since rewards are assigned at the network level, they can be used in the construction procedure. Further advantages of this approach, along with a careful development of the underlying theory, can be found in [22] where we construct a general reward structure at the SAN level and systematically generate variables from that reward structure. For the purposes of this paper, however, it suffices to briefly list the variables defined so that they can be used in the construction procedures developed in the next section.

In particular, the variables considered are organized according to reward structure type, category within a reward structure type, and variable type within a category. At the highest level, the variables are distinguished by the choice of reward structure type. By type we

mean one or more classes of functions that have a particular interpretation in terms of the SAN. For a given reward structure type, variables can be further distinguished by the interval of time on which they depend. Three categories of variables are distinguished at this level. The first category, *instant-of-time variables*, represents the status of the SAN at either a particular time t or in steady state. The second category, the *interval-of-time* variables, represent particular aspects of the behavior of a SAN over some finite or infinite interval of time. The final category, the *time-averaged interval-of-time* variables, represent particular aspects of the behavior of a SAN over some interval divided by the length of the interval.

Three types of variables are considered for the interval-of-time and time-averaged interval-of-time variable categories. The first type represents the total or time-averaged reward (relative to a particular reward structure) accumulated during some interval $[t, t+l]$. The second type corresponds to an interval of length l as t goes to infinity, and is useful in representing the reward that is accumulated during some interval of finite length in steady-state. The final variable type corresponds to the total or time-averaged reward accumulated during an interval starting at t and of length l as $l \to \infty$.

A SAN-level reward structure, together with eight variable types, has been defined that quantifies benefits associated with activity completions and particular numbers of tokens in places. In particular, we define an "activity-marking oriented reward structure" as follows:

Definition 1 *An* activity-marking oriented reward structure *of a stochastic activity network, with places P and activities A, is a pair of functions:*

$\mathcal{C} : A \to \mathbb{R}$ *where for $a \in A$, $\mathcal{C}(a)$ is the reward obtained due to completion of activity a, and*

$\mathcal{R} : \mathcal{P}(P, \mathbb{N}) \to \mathbb{R}$ *where for $\nu \in \mathcal{P}(P, \mathbb{N})$, $\mathcal{R}(\nu)$ is the rate of reward obtained when, for each $(p, n) \in \nu$, there are n tokens in place p,*

where $\mathbb{N}$ is the set of natural numbers and $\mathcal{P}(P, \mathbb{N})$ is the set of all partial functions between P and $\mathbb{N}$.

Informally, impulse rewards are associated with activity completions (via $\mathcal{C}$) and rates of reward are associated with numbers of tokens in sets of places (via $\mathcal{R}$). An element $\nu \in \mathcal{P}(P, \mathbb{N})$ is referred to as a *partial marking*. The marking is partial in the sense that natural numbers are assigned to some subset of P, namely the domain of the partial

function ν. This assignment is made in a manner identical to the way a (total) marking assigns natural numbers to all the places in the set P. Although $\mathcal{R}$ has a countably infinite domain, the number of elements ν that are of interest to the modeler and, hence, deserving of a non-zero reward assignment will generally be small compared to, say, the number of reachable stable markings of the SAN. Similarly, it will usually be the case that only a fraction of the SAN's activities will have non-zero rewards associated with their completions. We thus adopt the convention that rewards associated with activity completions and partial markings are taken to be zero if not explicitly assigned otherwise.

Variables are then defined by writing functions in terms of the reward structure and behavior of the network, and fall into the three categories discussed above. In particular, two instant-of-time variables are defined. The first quantifies the behavior of a stochastic activity network at a particular time t. More precisely, if we let V_t denote this variable type then

$$V_t = \sum_{\nu \in \mathcal{P}(P, \mathbb{N})} \mathcal{R}(\nu) \cdot I_t^{\nu} \; + \; \sum_{a \in A} \mathcal{C}(a) \cdot I_t^{a},$$

where

I_t^{ν} is an indicator random variable representing the event that the SAN is in a marking such that, for each $(p, n) \in \nu$, there are n tokens in p at time t, and

I_t^{a} is an indicator random variable representing the event that activity a is the activity that completed most recently at time t.

When I_t^{ν} and I_t^{a} converge in distribution for all ν and a with non-zero rewards as t approaches ∞, the "steady-state" reward obtained at an instant of time can be studied. If we denote the random variable with this steady-state distribution as $V_{t \to \infty}$, its value can be expressed as

$$V_{t \to \infty} = \sum_{\nu \in \mathcal{P}(P, \mathbb{N})} \mathcal{R}(\nu) \cdot I_{t \to \infty}^{\nu} \; + \; \sum_{a \in A} \mathcal{C}(a) \cdot I_{t \to \infty}^{a},$$

where

$I_{t \to \infty}^{\nu}$ is an indicator random variable representing the event that the SAN is in a marking such that, for each $(p, n) \in \nu$, there are n tokens in p in steady-state, and

$I_{t \to \infty}^{a}$ is an indicator random variable representing the event that activity a is the activity that completed most recently in steady-state.

As mentioned earlier, three variable types are considered for the interval and time-averaged-interval variables corresponding to an interval of length l starting at time t ($[t, t+l]$), an interval of length l as $t \to \infty$ ($[t, t+l], t \to \infty$), and an interval starting at t as $l \to \infty$ ($[t, t+l], l \to \infty$). Variable types of the interval category are denoted by "Y" while variables types of the time-averaged category are denoted by "W", each with the appropriate subscript. In particular, let

$$Y_{[t,t+l]} = \sum_{\nu \in \mathcal{P}(P, \mathbb{N})} \mathcal{R}(\nu) \cdot J^{\nu}_{[t,t+l]} \; + \; \sum_{a \in A} \mathcal{C}(a) \cdot N^{a}_{[t,t+l]}, \text{ and}$$

$$W_{[t,t+l]} = \frac{Y_{[t,t+l]}}{l}$$

where

$J^{\nu}_{[t,t+l]}$ is a random variable representing the total time that the SAN is in a marking such that, for each $(p, n) \in \nu$, there are n tokens in p during $[t, t+l]$, and

$N^{a}_{[t,t+l]}$ is a random variable representing the number of completions of activity a during $[t, t+l]$.

Similarly, if the required variables converge in distribution, we can define

$$Y_{[t,t+l],t\to\infty} = \sum_{\nu \in \mathcal{P}(P, \mathbb{N})} \mathcal{R}(\nu) \cdot J^{\nu}_{[t,t+l],t\to\infty} + \sum_{a \in A} \mathcal{C}(a) \cdot N^{a}_{[t,t+l],t\to\infty},$$

and

$$W_{[t,t+l],t\to\infty} = \frac{Y_{[t,t+l],t\to\infty}}{l},$$

where

$J^{\nu}_{[t,t+l],t\to\infty}$ is a random variable representing the total time that the SAN is in a marking such that, for each $(p, n) \in \nu$, there are n tokens in p during a interval of length l in steady-state, and

$N^{a}_{[t,t+l],t\to\infty}$ is a random variable representing the number of completions of activity a during an interval of length l in steady-state, and

$$Y_{[t,t+l],l\to\infty} = \sum_{\nu\in\mathcal{P}(P,\mathbb{N})} \mathcal{R}(\nu)\cdot J^{\nu}_{[t,t+l],l\to\infty} \; + \; \sum_{a\in A} \mathcal{C}(a)\cdot N^{a}_{[t,t+l],l\to\infty},$$

and

$$W_{[t,t+l],l\to\infty} = \lim_{l\to\infty} \frac{Y_{[t,t+l]}}{l}$$

where

$J^{\nu}_{[t,t+l],l\to\infty}$ is a random variable representing the total time that the SAN is in a marking such that, for each $(p,n)\in\nu$, there are n tokens in p during $[t,\infty)$, and

$N^{a}_{[t,t+l],l\to\infty}$ is a random variable representing the number of completions of activity a during $[t,\infty)$.

D Example Variables

These variable types can be specialized, through appropriate choices of reward structures, to cover a wide variety of specific performance, dependability, and performability measures (for a detailed discussion, see [22]). In terms of the CSMA/CD network considered earlier, we now define several variables to illustrate the specification method.

In particular, the first performance variable studied is the expected queue length in steady-state at each type of station. This can be obtained using the reward structure

$$\mathcal{C}_{st}(a) = 0 \quad \forall a\in A$$

$$\mathcal{R}_{st}(\nu) = \begin{cases} i & \text{if } \nu = \{(A,i)\} \\ 0 & \text{otherwise} \end{cases}$$

and variable $E[V_{t\to\infty}]$. Since markings and activity completions in the network submodel do not contribute to this variable, the reward structure for this submodel $(\mathcal{C}_Z,\mathcal{R}_Z)$ is defined such that $\mathcal{C}_Z(a)=0,\ \forall\ a\in A$, and $\mathcal{R}_Z(\nu)=0,\ \forall\nu$.

Performance variables representing the fractions of time various actions are present on the channel are constructed using a null reward structure for each station submodel (i.e, $(\mathcal{C}_Z,\mathcal{R}_Z)$) and a reward structure for the network submodel such that:

$$\mathcal{C}_N(a) = 0 \quad \forall a\in A$$

$$\mathcal{R}_N(\nu) = \begin{cases} 1 & \text{if } \nu = \{(channel, i)\} \\ 0 & \text{otherwise} \end{cases}$$

where i is the marking of the channel when the action is taking place and $E[V_{t\to\infty}]$ is the variable. For example, to determine the fraction of time a propagated packet is on the channel, i would be taken to be two. The fractions of time that the bus is idle, an unpropagated message is on the bus, and a corrupted message is on the bus are determined in a similar manner.

Other variables can be constructed in a similar manner, and can capture system dependability and performability, as well as performance [22]. For the purpose of our discussion, however, these are sufficient to illustrate the construction methods developed in the following section.

III REDUCED BASE MODEL CONSTRUCTION

Given a performance variable of the type described above, let us now consider the problem of constructing a SAN-derived stochastic process that i) supports solution of the variable in question and ii) is solvable by practical means. As stated in the introduction, for stochastic extensions of Petri nets, the state space of such a process is typically taken to be the set of reachable, stable markings of the net, i.e., the resulting process (base model) is the net's *marking behavior*. Certain variables, however, cannot be supported by this process, e.g., variables referring to activity completions (or, in SPN terms, transition firings) which, in general, cannot be uniquely inferred from knowledge of a marking trajectory. Accordingly, a wider variety of performance variables can be supported if, along with the marking, the state includes the name of the most recently completed activity; in this case, the resulting base model is referred to as the *activity-marking behavior*. While the latter choice suffices to support the types of variables described in the previous section, their use with realistic systems can result in state spaces that are very large. For certain classes of systems, however, reduced base models can be constructed that are much smaller (in number of states) and still preserve system information necessary to solve for the chosen performance variables. This section discusses the theory and application of reduced base model construction methods. These methods are useful for hierarchical systems where, at certain levels in the hierarchy, subsystems are replicated and where contributions to total reward (as defined by an activity-marking variable) are the same for identical subsystems. Such hierarchical systems are becoming increasingly more prevalent and, in a computer

context, include both multiprocessor systems (with many identical processors and busses) and computer networks (with many identical stations).

As with the conditions on system structure, the conditions on the performance variables are not severe. In fact, since the structure of replicated systems is identical, it is natural to assume that possible behaviors of identical subsystems would be treated in an equivalent manner. An example of this would be a computer network consisting of many types of stations, where one is interested in the queue length at a particular type of station, not at an individual station within a type. The next subsection introduces operations that aid in the construction of stochastic activity networks that meet these conditions.

A Construction Operations

In order to insure that models built have characteristics that permit construction of a reduced base model, we construct them in a bottom-up manner using two operations. Before we can do this, however, a formal definition of the operands of the construction operations is needed. Specifically, we introduce the notion of a "SAN-based reward model".

Definition 2 *A SAN S with places P, together with a reward structure ($\mathcal{C}$,$\mathcal{R}$) and set of distinguished places $P_D \subseteq P$, is a* SAN-based reward model *(S,$\mathcal{C}$,$\mathcal{R}$, P_D).*

A SAN-based reward model (SBRM) provides us with a structure that can be used to construct larger models that have compact state-space representations. Since the operations discussed below are defined on SAN-based reward models, they operate on the reward structure as well as the network. This insures that the defined variable is supported by the reduced model.

The first construction operation *replicates* a SAN-based reward model. This operation is useful when a system has two or more identical subsystems. The effect of the operation depends on a set of places which is a subset of the distinguished places of the input SBRM. These places allow communication between the replicated submodels and are not replicated when the operation is carried out. Informally, the result of the replicate operation on a SAN-based reward model is another SAN-based reward model, where

1. The SAN component is constructed by replicating the original SAN a certain number of times, holding some subset of the distinguished places of the input SBRM common to all replicate SANs, and

2. The reward structure is constructed by assigning an impulse reward to each replicate activity equal to its reward in the original model and reward rates to each partial marking in the new model equal to the rate assigned to the corresponding partial marking in the original model, and

3. The set of distinguished places is the set of places used in the construction operation.

The replicate operation allows us to construct SAN-based reward models that consist of several identical component SBRMs. This permits the representation of systems that consist of a single replicated structure. Typically, however, distributed systems consist of several different structures, each of which may be replicated. The combination of several different structures is accomplished using the join operation. As with the replicate operation, the *join* operation both acts on and produces SAN-based reward models. Informally, the effect of the operation is to produce a new SAN-based reward model which is a combination of the individual subnetworks and reward structures. Again, certain places play an important role in the construction operation. In this case, however, a *list* of places is associated with each component SBRM in a manner such that the cardinality of each list of places is the same.

In the joined model, the corresponding places in each list form a single place, with connections to gates and activities as in the individual models. These lists of places thus allow communication between joined subnetworks. As with the replicate operation, each list of places must be a subset of the set of distinguished places of the component SBRM. The reward structure for the new SBRM is constructed in a similar manner to that used for the replication operation. Specifically, the reward structure is constructed by 1) assigning a reward to each activity in the new model equal to the reward of the corresponding activity in the original model and 2) assigning a reward rate for each partial marking in the new model equal to the rate assigned to the corresponding partial marking in the original model.

These operations can be applied iteratively to build models of systems that are organized in a hierarchical manner. We refer to a SAN-based reward model constructed in this way as a *composed SAN-based reward model*. In order to describe the structure of a composed SAN-based reward model, we make use of a directed tree with three types of nodes: *model/reward structure nodes*, *replicate nodes*, and *join nodes*. Model/reward structure nodes have a degree of zero. These nodes define distinct subnetworks and reward structures in the composed model, and serve as a basis on which to apply the construction operations. Replicate nodes have a degree of one with an outgoing arc that points to the

SAN-based reward model that is replicated. Join nodes have a degree equal to the number of SBRMs that are joined together. In this case, each arc points to a SBRM that is joined.

Information contained inside each node describes the parameters of the particular operation. In particular, each model/reward structure node contains a triple, $(S, \mathcal{C}, \mathcal{R})$, specifying the subnetwork and reward structure for the node. The SBRM associated with the node consists of the SAN and reward structure of the node together with a set distinguished places equal to the set of places of the SAN. Replicate nodes contain an integer denoting the number of times the associated SBRM is to be replicated and a set of places to be used in the replicate operation. Join nodes contain an integer (denoting the number of SBRMs to be joined) and a list of places for each SBRM to be joined.

To illustrate the construction of a composed SAN-based reward model, consider the model of the CSMA/CD station and bus described earlier. Two scenarios are considered. In the first, all stations in the network are identical. In the second, one station is considered to be a high-priority station, and all other stations are considered to be normal priority. Here the priority of a station is determined by the mean time it waits before trying to access the bus; the high-priority station waits (on the average) one-tenth the time of the other stations.

The SAN models and performance variables specified earlier provide the information necessary to construct composed SAN-based reward models for each scenario. This is done by replicating the normal and high-priority stations the required number of times and adjoining them with the network submodel and reward structure. The particular reward structures that are used depend on the particular performance variable being solved. For example, to determine the expected queue length at a station for the homogeneous system (scenario 1), the SAN-based reward model given in Figure 3 can be used. In this figure, $S_{STATION}$ is the station submodel, $S_{NETWORK}$ is the network submodel, and n is the number of stations in the network. Within a replicate node (denoted R in the figure) the superscript (n) refers to the number of times the SBRM below is to be replicated, and the subscript refers to the set of distinguished places for the operation, i.e. those places that are to not be replicated when the operation is carried out. Similarly, within a join node (denoted J in the figure), the the ordered lists of places refer to the places that are to be identified with one another during the join operation. More specifically, in this example, the two places named *channel* (one in each sub-SBRM) are to be made a single place in the resulting SBRM.

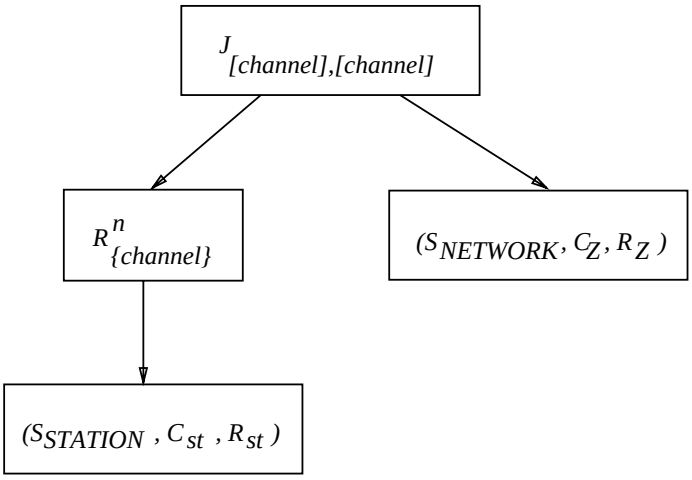


Figure 3: Homogeneous Network

If we define S_{HI} to be a high-priority station submodel, then a SAN-based reward model which can be used to determine the expected queue length at the high-priority station is given in Figure 4, where n is now the number of normal priority stations in the model. SAN-based reward models for the other variables can be constructed in a similar manner.

Before we can solve the composed models, we must consider the mapping between the model and reduced base model representation. This is done in the next section.

B Representations of State

We now consider possible state space and, in turn, base model, representations for composed SAN-based reward models. One approach would be to use activity-marking or marking states as the state representation, and a detailed base model as the process representation. This, however, leads to excessively large state spaces for many models. The aim, then, is to find one or more alternative representations for state that lead to reduced base models. If support of the variable in question were the only concern, the set of possible values of the variable could be taken as the states of the process. However, the state behavior of this process is not generally Markovian or discrete state, and thus very difficult to solve. On the other hand, if "solvability" were the only issue, the states of the detailed base model could be lumped into a single state in a manner such that the behavior of the reduced base model was easy to deduce. This will not suffice, however, since this process will not generally support a selected variable. Thus both support and solvability are important considerations in selecting a reduced base model representation.

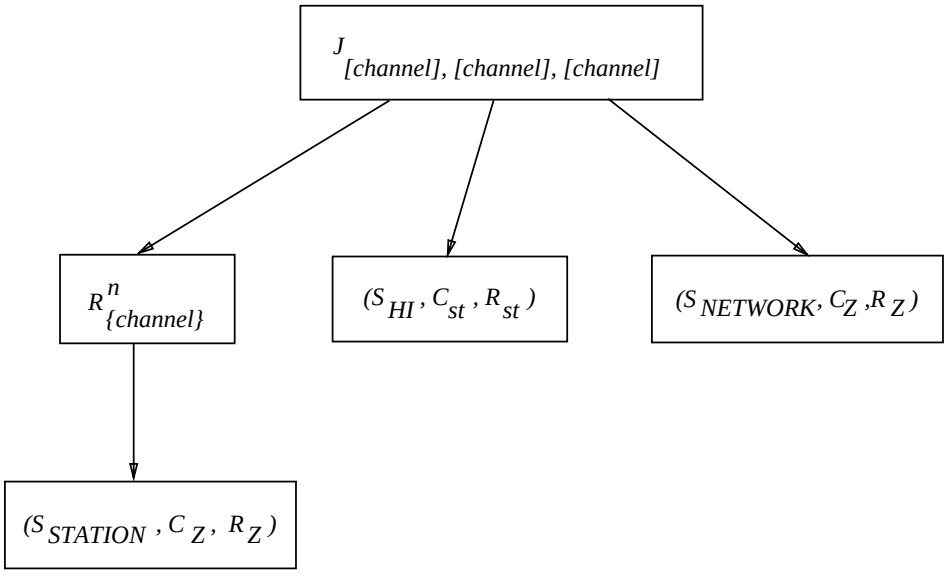


Figure 4: Normal/High Priority Network

Since we consider a process to be solvable if it is Markov, results regarding lumpings of a Markov process [23] and, equivalently, functions of a Markov process that are Markov [24] are relevant. These results give conditions on a Markov process and a lumping of that process (or equivalently, a functional of that process) such that the new process is also Markov. These conditions are limited, however, in two respects. First, they consider only the solvability of the resulting process; no requirement is made that the resulting process support any particular variable. Second, a direct application would require that the detailed process be generated and that a suitable lumping be found. Machine memory size limitations typical preclude generation of the detailed process and, even if the process could be generated, there is no easy way (without additional information regarding the system being modeled) to determine a suitable lumping from the detailed model.

Stochastic activity networks and the construction operations introduced in the previous subsection provide us with a way to avoid both of these difficulties, due to the way in which the construction operations were defined. In particular, the order of application of the operations (as specified by the directed tree representation introduced in the previous subsection) determines a notion of state which both supports the variable in question and is Markov whenever the detailed base model representation is Markov. Informally, this notion of process state is determined such that for each replicate operation, the number of replicate SBRMs in each possible submodel "state" is recorded and for each join operation, a vector

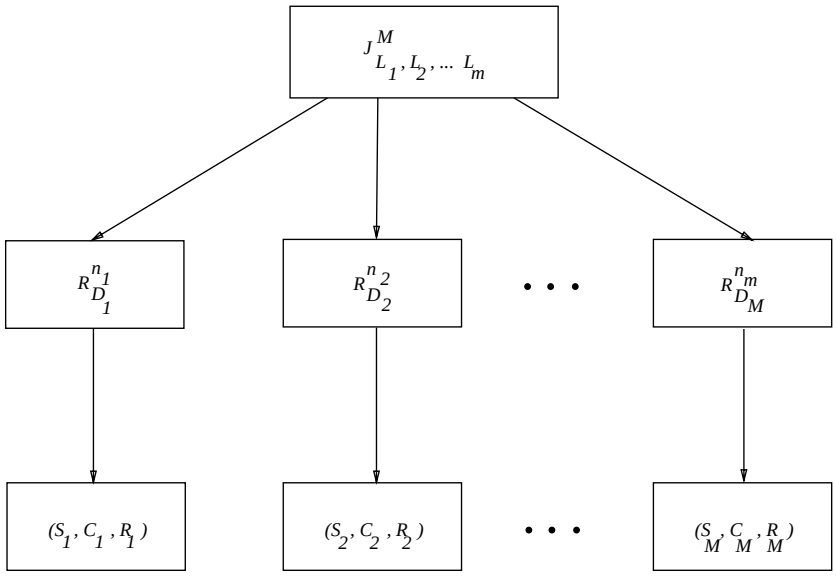


Figure 5: Example Composed SAN-Based Reward Model

of the "state" of each joined submodel is kept. At the lowest level, the "state" of a SBRM model is the marking of the submodel. The complete state for the composed model is then taken to be a pair where the first component is the impulse reward of an activity in the model and the second component is a state formed as described above.

To give a more precise example of how the notion of state would be formulated for a particular application of the replicate and join operations, consider the composed SAN-based reward model of Figure 5, where the lowest level nodes are arbitrary models and reward structures. Now let E be the state space of the activity-marking behavior for this model, and M to be the set of marking components of this am-space. Then define projections to examine the markings of submodel types and individual submodels. Specifically, for each $\mu \in M$ define μ^j to be the projection of (global marking) μ on the places of submodels of type j, $j = 1$ to m. Similarly, define $\mu^{j,k}$ to be the projection of (global marking) μ on the places of submodel k of type j (where places within a submodel have the same ordering).

Furthermore, for each μ^j let $B_{\mu^j} =< \mu^{j,1}, \mu^{j,2}, \ldots, \mu^{j,n_j} >$ be the bag of projected markings for submodels of type j in marking μ. The bag formalism [25] is useful here, since it allows us to compactly characterize the different configurations various replicates of a submodel are in, without distinguishing an ordering among them. As will be seen in the following, this representation allows us to specify a functional which, due to the nature of the replicate and join operations, can serve as a reduced base model for the system in

question. If, further, we let

$$f(a, \mu) = (\mathcal{C}(a), B_{\mu^1}, B_{\mu^2}, \ldots, B_{\mu^m}) \tag{1}$$

then this mapping defines a functional of the am-behavior associated with the SBRM of Figure 5. Intuitively, for this example, the functional defines a notion of state that keeps track of the number of submodels of each type in particular markings, and the impulse reward of the activity that led to this marking state. Different composed model structures would result in different notions of state. This functional is defined in terms of two processes derived from the am-behavior: the minimal process and the discrete-time embedded process. Following the convention used for Markov renewal processes, the *minimal behavior* $Z = \{Z_t \mid t \in I\!R^+\}$ of the am-behavior is a process defined such that Z_t is the am-state of the network at time t. Likewise, the *discrete-time embedded process* associated with the am-behavior is a process $R = \{R_n \mid n \in I\!N\}$ defined such that R_n is the am-state reached after the *nth* timed activity completion. Given these processes, the minimal and discrete-time embedded behaviors of a reduced base model can be defined as

$$U = \{f(Z_t) |\ t \in I\!R^+\}$$

and

$$T = \{f(R_n) \mid n \in I\!N\}$$

respectively.

While the functional notion of Equation (1) is convenient for two levels of operations, it becomes cumbersome if extended directly to more levels. Instead, we use a set of equations relating SAN-based reward models at one level to those at the next lower level. The nature of this relationship depends on the specific operation used at the particular level. As with the two-level example just given, each replicate node corresponds to a "number of" relation in the state space, and is represented as a bag of submodels at the next lower-level. On the other hand, the "state" of each joined SBRM must be represented explicitly, and hence the operation is represented as a vector of joined submodels. At the lowest level are the submarkings of the individual SAN-based reward models. As with the two-level case discussed above, activities are named in a way which designates the particular submodel that they belong to, and projections of markings on particular submodels are designated by superscripts on the marking (i.e., $\mu^{n_1,n_2,\ldots,n_h}$ denotes the projection of global marking μ on the places of the n_1th SBRM of the highest-level operation, the n_2th SBRM of the next-level operation, and so on).

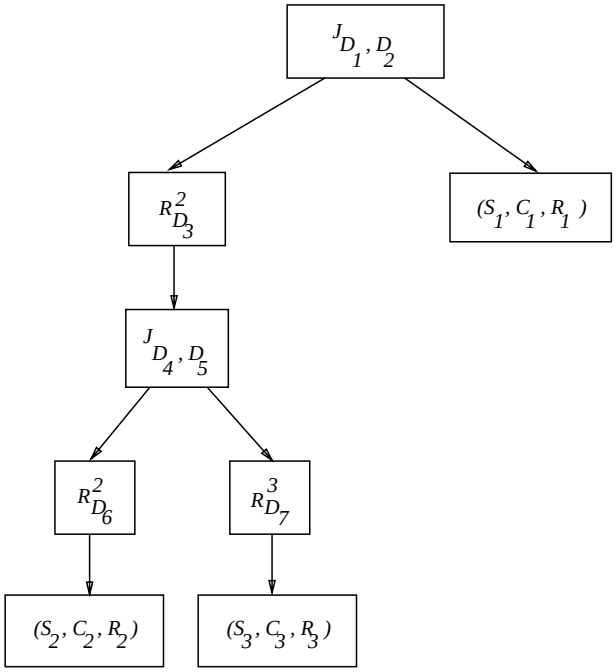


Figure 6: Example Composed SAN-based Reward Model

For example, consider the SAN-based reward model of Figure 6. The functional describing the mapping from am-states to reduced states is written as:

$$\begin{aligned}
f(a,\mu) &= (\mathcal{C}(a), V) \\
V &= (B_1, \mu^2) \\
B_1 &= < V_{11}, V_{12} > \\
V_{11} &= (B_{111}, B_{112}) \\
V_{12} &= (B_{121}, B_{122}) \\
B_{111} &= < \mu^{1111}, \mu^{1112} > \\
B_{112} &= < \mu^{1121}, \mu^{1122}, \mu^{1123} > \\
B_{121} &= < \mu^{1211}, \mu^{1212} > \\
B_{122} &= < \mu^{1221}, \mu^{1222}, \mu^{1223} > .
\end{aligned}$$

Using this notation, the letter B denotes a bag of SBRMs at the next lower level and V denotes a vector. The superscripts on markings denote the marking of a particular submodel.

As alluded to earlier, in order for reduced base models to be useful, they must support the variables of interest and be solvable. Sufficient conditions for this to occur are known, and are now given. In particular,

1. The reduced base model of a composed SBRM supports the instant-of-time, interval-of-time, and time-averaged interval-of-time activity-marking oriented variables defined herein,

2. The minimal behavior (U) of the reduced base model of a composed SBRM is Markov whenever its minimal am-behavior is Markov, and

3. The discrete-time embedded behavior (T) of the reduced base model of a composed SBRM is Markov whenever its am-behavior is Markov.

Theorems and accompanying proofs that establish these results can be found in [20], along with theorems and proofs that show when the detailed base models are Markov. Intuitively, however, these models are solvable because replicate submodels of a particular type behave in an equivalent manner when in identical markings. Similarly, since the performance variable was constructed in a bottom up manner as part of the composed SAN-based reward model, it is "symmetric" with respect to different replicates of a given submodel type and thus supported by the reduced base model. Since we can generate these processes without first generating a detailed base model, we have avoided both of the problems associated with classical lumping methods.

Taken together, these theorems show that both the minimal and discrete-time embedded behaviors of the reduced base model of a composed SBRM are Markov whenever the corresponding activity-marking behavior is Markov. Since these processes also support the variable of interest, they can be used to solve for the variable's probabilistic behavior. Construction procedures which generate a reduced base model are now investigated.

C Construction Procedure

While a reduced base model is indeed solvable whenever the corresponding detailed base model is and, moreover, it supports the specified performance variable, these facts do not suggest a manner to generate a reduced base model. In fact, the proof of solvability given in [20] relies on lumping arguments. Clearly, one would hope that a procedure could be formulated that did not rely on the prior generation of a detailed base model. This is, in fact, possible, and the resulting procedure is more efficient than those derived for detailed base models.

The procedure operates on reduced states and "representative" markings. A *representative marking of a reduced state* is any marking that corresponds (via the functional defined

earlier) to the marking portion of the reduced state. Once a state has be expanded, the representative marking is no longer needed; hence representative markings need be associated with states only until they are expanded.

In the following procedure, f is the functional defined earlier which maps am-states to reduced states, U is the set of reduced states, X is the set of reduced states yet to be expanded and representative markings for these states, T is the set of transition rates from marking portions of reduced states to reduced states. Furthermore, Y is a set of triples denoting possible next reduced states, representative markings for these states and rates to these states from the marking portion of a reduced state. The marking portion of a reduced state u is denoted as u^m. The rate from a marking portion of a reduced state, u^m, to a reduced state u' is written as $\lambda_{u^m,u'}$.

Procedure 1 *(Procedure to generate the reduced base model of a stochastic activity network with initial marking μ_0.)*

Let $U = \{f(\bigtriangledown, \mu_0)\}$.
Let $X = \{(f(\bigtriangledown, \mu_0), \mu_0)\}$.
Let T *equal the null set.*
While $X \neq$ null*:*
    Let $(u, \mu) = some\_element\{X\}$.
    Let $X = X - \{(u, \mu)\}$.
    Let $Y = poss\_U(\mu)$.
    For each $(y_u, y_\mu, y_\lambda) \in Y$*:*
        If $y_u \in U$ *then*
            Let $T = T \cup \{(u^m, y_u)\}$.
            Let $\lambda_{u^m,y_u} = y_\lambda$.
        else
            Let $U = U \cup \{y_u\}$.
            Let $T = T \cup \{(u^m, y_u)\}$.
            Let $\lambda_{u^m,y_u} = y_\lambda$.
            If there does not exist a $u \in U$ *such that* $u^m = y_u^m$
                Let $X = X \cup \{(y_u, y_\mu)\}$.
    Next $(y_u, y_\mu, y_\lambda) \in Y$.
While end.

Algorithm 1 ($poss\_U$) generates the set of next possible reduced states and rates to these states from a particular marking portion of a reduced state. In this algorithm, the activity that is completed is referred to as the *representative activity*. The number of activities that are in the set of replicate activities is denoted by n. After possible next stable markings

and probabilities of these states for each representative activity are computed using an algorithm given in [20], rates to these states are then found by multiplying the rate due to a single activity by the number of replicate activities in the particular submodel marking. The following algorithm describes this in a more precise manner.

Algorithm 1 *(Generates the set, Y, of reduced states, representative markings, and rates to these states from a representative marking μ of a particular reduced state.)*

Let Y equal the null set.
Let D be the set of pairs (a, n) where a is a representative activity and n is cardinality of the set of replicate activities of a in μ.
For each $(a, n) \in D$:
    Compute the set of possible next stable markings reached upon completion of a in μ (i.e. $NS(a, \mu)$), the probability of reaching each of these markings (i.e. $h_{\mu,a}$), and check whether this distribution is invariant over possible sets of activity choices
    If the distribution is not invariant over possible sets of activity choices then
        Signal SAN is not well specified and abort algorithm.
    For each $\mu' \in NS(a, \mu)$:
        If $\exists (y_u, y_\mu, y_\lambda) \in Y$ such that $f(a, \mu') = y_u$ then
            $(y_u, y_\mu, y_\lambda) = (y_u, y_\mu, y_\lambda + r_a(\mu) * n * h_{\mu,a}(\mu'))\}$.
        else
            Let $Y = Y \cup \{(f(a, \mu'), \mu', r_a(\mu) * n * h_{\mu,a}(\mu'))\}$.
    Next $\mu' \in NS(a, \mu)$.
Next $(a, n) \in D$.

Note that in order for this algorithm to run successfully to completion, the SAN in question must be "well specified." Informally, a SAN is well specified if its probabilistic behavior is completely specified, i.e. an unambiguous probabilistic representation can be derived from the SAN. This may not be the case if more than one instantaneous activity is enabled in a reachable marking. An algorithm to check whether a specific SAN is well specified is given in [20].

D Example Results

Using the developed theory and algorithms, we are now prepared to construct reduced base model representations for our running examples. Recall that we have considered two scenarios, one where all nodes in the network were homogeneous and we wish to determine expected queue length at a node and the fraction of time various types of packets are on the network, and a second where there are two classes of nodes (high and low priority) and we wish to determine the queue length at a high priority node and the time various packet types are on the network.

Space does not permit us to explicitly describe the reduced base model generated in each case, but an informal discussion of the nature of the spaces may be useful. For the homogeneous system, since the expected queue length at an individual station can be determined from the expected number of packets at all stations, it is not necessary to keep track (in the state-space) of the queue lengths of each station, but only the *number* of stations with each particular queue length. In terms of the SAN representing the network submodel, this means that the notion of state need only keep track of the *number* of submodels in a particular marking, the the marking of each submodel. Furthermore, since the defined performance variables do not depend on the knowledge of activities that complete, this information can also be abstracted from the detailed (activity-marking) state space.

In the case of the second scenario, since there are two classes of stations, and the aim is to determine, among other things, the expected number of packets queued at a high priority station, we must have a more refined notion of behavior than in the first scenario. In particular, we must distinguish the two classes of stations in the state space, and in this case, keep track of the *number of stations in each class* of in a particular marking. Again, since the performance variable does not depend on knowledge of activities that complete, this information is not needed in the state representation. As can be seen from these two examples, the notion of "state" varies among different SAN-based reward models, depending on the structure of the specific SAN and performance variables desired.

State-level representations for each scenario were constructed using an extension to METASAN[2] [26] that permits the construction and solution of reduced base model representations. Four state space representations were contrasted: the activity-marking state space, the marking state space, and the reduced base model state-space size for each scenario.

The results are presented in Table 3. One can see that for each network configuration considered, the reduced base model construction technique generated significantly fewer states compared to the marking and activity-marking states. Moreover, the rate of growth of the state-space experienced by increasing the number of stations in the network was much smaller for the reduced base model construction technique. In fact, the number of states generated using this technique remained quite small even when the generation of state-spaces using standard techniques became intractable.

[2]METASAN is a registered Trademark of the Industrial Technology Institute.

Number of Stations	Detailed Base Model		Reduced Base Model	
	am-states	*m-states*	*Scenario 1*	*Scenario 2*
2	135	57	30	57
3	841	261	61	147
4	4277	1041	102	276
5	17820	3873	153	444
6	80000	13883	214	651
7	$*^a$	$*^a$	285	897
8	$*^a$	$*^a$	366	1182
9	$*^a$	$*^a$	457	1506
10	$*^a$	$*^a$	558	1869
11	$*^a$	$*^a$	669	2271
12	$*^a$	$*^a$	790	2712

[a] state-space too large to be computed

Table 3: State-Space Sizes Obtained Using Various Construction Techniques

IV CONCLUSIONS

As argued in the introduction, stochastic Petri nets and extensions are a truly useful model class for representing distributed systems, in general, and large-scale computer-communication networks, in particular. However, traditional model construction and solution methods for these models limit their usefulness, due to the extremely rapid growth of the size of the stochastic process used in model solution. This paper reports on the results of one approach to solve this problem, namely, through the use of reduced base model construction methods. Reduced base model construction methods make use of both the structure of a particular model and the performance variables desired in order to choose an appropriate notion of state, typically reducing the number of states that must be considered for an analytic solution. Furthermore, unlike traditional lumping methods, this decision is made *a priori* and does not require the generation of a detailed state space. A procedure, ready for computer implementation, was given that embodies the these concepts and can be used to generate reduced base model representations for SAN-based reward models. This work thus makes an important contribution towards alleviating the state space explosion problems associated with traditional model construction methods.

Further work is underway with this same general aim. In particular, we are working on a implementation of these and other recent advances in model construction and solution

methods in software form. This will allow us to test the methods developed herein on large multi-level examples. In addition, we are also investigating extensions to the theory that should result in methods applicable to wider classes of SANs (e.g., those where submodels are similar, but not identical). Finally, completion of the software tool mentioned above should allow us to apply the methods in many application areas, hopefully resulting in new insights regarding particular system designs.

REFERENCES

[1] P. Heidelberger and S.S. Lavenberg, "Computer performance evaluation methodology", *IEEE Trans. on Computers*, vol. C-33, no. 12, pp. 1195–1220, December 1984.

[2] L. Kleinrock, *Queueing Systems, Volume I: Theory*, New York: John Wiley, 1975.

[3] S. Natkin, "Reseaux de Petri stochastiques", Thèse de Docteur-Ingénieur, CNAM-PARIS, June 1980.

[4] M. Molloy, *On the integration of delay and throughput measures in distributed processing models*, PhD thesis, UCLA, 1981.

[5] *Proc. of the First International Workshop on Timed Petri Nets*, IEEE Press, Torino, Italy, July 1985.

[6] *Proc. of the International Workshop on Petri Nets and Performance Models*, IEEE Press, Madison, WI, Aug. 1987.

[7] J. F. Meyer, "On evaluating the performability of degradable computing systems", *IEEE Trans. on Computers*, vol. C-22, pp. 720–731, August 1980.

[8] M. A. Marsan, G. Balbo, G. Chiola, S. Donatelli, "On the product-form solution of a class of multiple-bus multiprocessor system models", *The Journal of Systems and Software*, vol. 1, no. 2, pp. 117–124, 1986.

[9] A. A. Lazar and T. G. Robertazzi, "Markovian Petri net protocols with product form solution", in *Proc. INFOCOM '87*, San Francisco, CA, Mar. 1987.

[10] A. Zenie, "Colored stochastic Petri nets", in *Proc. International Workshop on Timed Petri Nets*, pp. 262–271, Torino, Italy, July 1985.

[11] C. Lin and D. C. Marinescu, "Stochastic high-level Petri nets and applications", *IEEE Trans. on Computers*, vol. C-37, no. 7, pp. 815–825, July 1988.

[12] C. Dutheillet and S. Haddad, "Regular stochastic Petri nets", *Proc. Tenth European Workshop on Application and Theory of Petri Nets*, Bonn, W. Germany, June 1989.

[13] G. Chiola and G. Franceschinis, "Colored GSPN models and automatic symmetry detection", *Proc. Third International Workshop of Petri Nets and Performance Models*, Kyoto, Japan, Dec. 1989.

[14] G. Chiola, C. Dutheillet, G. Franceschinis, and S. Haddad, "On well-formed coloured nets and their symbolic reachability graph", *Proc. Eleventh International Conference on Application and Theory of Petri Nets*, Paris, France, June 1990.

[15] G. Balbo, S. Bruell and S. Ghanta, "Combining Queueing network and generalized stochastic Petri net models for the analysis of some software blocking phenomena," *IEEE Trans. on Software Engineering*, vol. SE-12, No. 4, pp; 561–576, 1986.

[16] G. Balbo, S. Bruell and S. Ghanta, "Combining Queueing network and generalized stochastic Petri net models for the solution of complex models of system behavior," *IEEE Trans. on Computers*, vol. 37, No. 10, pp; 1251–1268, 1988. 1986.

[17] W. H. Sanders and J. F. Meyer, "Variable driven construction methods for stochastic activity networks," in *Applied Mathematics and Performance/Reliability Models of Computer/Communication Systems* (ed., G. Iazeolla), North-Holland, 1987.

[18] A. Movaghar and J. F. Meyer, "Performability modeling with stochastic activity networks," in *Proc. 1984 Real-Time Systems Symp.*, Austin, TX, Dec. 1984.

[19] J. F. Meyer, A. Movaghar, and W. H. Sanders, "Stochastic activity networks: Structure, behavior, and application," in *Proc. International Workshop on Timed Petri Nets*, Torino, Italy, July 1985, pp. 106–115.

[20] W. H. Sanders, "Construction and solution of performability models based on stochastic activity networks," Computing Research Laboratory Technical Report CRL-TR-9-88, The University of Michigan, Ann Arbor, MI, August 1988.

[21] R. A. Howard, *Dynamic Probabilistic Systems, Vol II: Semi-Markov and Decision Processes*, New York: Wiley, 1971.

[22] W. H. Sanders and J. F. Meyer, "A unified approach for specifying measures of performance, dependability, and performability," to be presented at the *International Working Conference on Dependable Computing for Critical Applications*, Santa Barbara, CA, August 21-23, 1989.

[23] J. C. Kemeny and J. L. Snell, *Finite Markov Chains*, Princeton: D. Van Nostrand Co., Inc., 1969.

[24] M. Rosenblatt, *Markov processes. Structure and Asymptotic Behavior*, Berlin: Springer-Verlag, 1971.

[25] J. L. Peterson, *Petri Net Theory and the Modeling of Systems*, Englewood Cliffs: Prentice-Hall, 1981.

[26] W. H. Sanders and J. F. Meyer, "METASAN: A performability evaluation tool based on stochastic activity networks", *Proc. ACM-IEEE Comp. Soc. 1986 Fall Joint Comp. Conf.*, Dallas, TX, November 1986.