Stochastic Petri Nets and Their Applications to Performance Analysis of Computer Networks *

Kishor S. Trivedi, Hairong Sun
{ kst,hairong@ee.duke.edu}
Center for Advanced Computing and Communications
Department of Electrical and Computer Engineering
Duke University
Durham, NC 27708

Abstract

Continuous-time Markov chains are used extensively to analyze the performance of various computer networks. However, constructing and solving continuous-time Markov chain is a tedious and error-prone procedure, especially when the studied systems are complex. Stochastic Petri nets and the corresponding software packages provide automated generation and solution to continuous-time Markov chains. This paper gives an overview of stochastic Petri nets. Two examples in ATM networks are presented and studied to illustrate how to use stochastic Petri nets for performance analysis of computer networks.

Index Terms: Stochastic Petri Nets, Stochastic Reward Nets, Computer Networks, ATM Networks

*This research was supported in part by the National Science Foundation under Grant No. EEC9418765.

1 Introduction

From ARPAnet to Internet and to Internet 2, from Ethernet to fast Ethernet and to gigabit Ethernet, from packet switching to Asynchronous Transfer Mode (ATM) switching and to label switching, computer networks have been evolving dramatically from 1960s. As a consequence of this rapid progress, thorough understanding and evaluation of the behavior of the computer networks is critical to meet cost and performance requirements through out the life cycle of the system. As a performance evaluation methodology, stochastic modelling is preferable to deterministic evaluation (e.g., actual measurement and benchmarking), because the latter can be resorted to only after the computer network is implemented while the former can be used during the design phase. Stochastic modelling can be classified into discrete-event simulation and analytical solution methods. Each one of these has its own merits and disadvantages and should be used in a complementary manner. In this paper, we will focus on analytical solution methods especially based on Markov models.

First introduced by A. A. Markov in 1907, Markov chains have been in use in performance analysis since around 1950. A Markov chain consists of a set of states and a set of labeled transitions between the states. A state of the Markov chain can model various conditions of interest in the system being studied. For example, these could be the number of packets in the buffer, the number of active users in the network, etc. After a sojourn in a state, the Markov chain will make a transition to another state. Such transitions are described either with *transition probability matrix* $\mathbf{P}$ (in case of discrete-time Markov chain, $DTMC$) or *infinitesimal generator matrix* $\mathbf{Q}$ (in case of continuous-time Markov chain, $CTMC$). Steady-state dynamics of Markov chains, if it exists, can be studied by using a system of linear equations with one equation for each state, i.e., $\mathbf{V}\ \mathbf{P} = \mathbf{V}$ and $\mathbf{V1} = 1$ for $DTMC$ or $\mathbf{\Pi}\ \mathbf{Q} = 0$ and $\mathbf{\Pi 1} = 1$ for $CTMC$, where $\mathbf{V}$ and $\mathbf{\Pi}$ are the steady-state

probability vectors for $DTMC$ and $CTMC$ respectively. Transient behavior of a $CTMC$ gives rise to a system of first-order, linear, ordinary differential equations, i.e., $\frac{d\,\mathbf{\Pi(t)}}{dt} = \mathbf{\Pi(t)}\ \mathbf{Q}$.[19, 13]

In the area of computer networks and telecommunications, the most commonly used method is abstracting the physical system at first, then constructing CTMC or DTMC, and then setting up ordinary differential equations (for transient solution) or linear equations (for steady-state solution) manually, and finally writing a program for the numerical solution to the above equations. It is a rather tedious and error-prone procedure, especially when the number of states becomes very large. There are some efforts on developing software package to solve $CTMC$ or $DTMC$ automatically, while still requiring to construct the $CTMC$ by hand. On the other hand, the Markov model of a system is sometime far removed from the shape and general feel of the system being modeled. System designer may have difficulty in directly translating their problem into a Markov chain. Since late 1980s, some scientists have been devoting to develop new modelling formalism and software packages for the automated genaration and solution of Markovian stochastic systems. The efforts led to the emergence of a new formalism called stochastic Petri nets (SPN) which is more concise in its specification and whose form is closer to a designer's intuition about what a model should look like. Some software packages such as $SPNP$ [10, 12], $DSPNexpress$ [5, 6], $TimeNet$ [27], $UltraSAN$ [33], and $GreatSPN$ [8], are available, which can translate the SPN model to $CTMC$ and then solve it automatically. The users' effort is now just in building an SPN model from the real system.

In this paper, we will show how to build an SPN model to study the performance issue in computer networks. In section 2, the concept of SPN, $GSPN$ and SRN are described. As an example, we construct a stochastic Petri net model for the Early Packet

Discard algorithm in ATM networks in section 3. In section 4, a performance model for Ethernet/ATM bridge is built. In section 5, we overview the recent advances in SPN. The conclusions are drawn in section 6.

2 Stochastic Petri Nets

2.1 Petri Nets

Petri nets were originally introduced by C. A. Petri in 1962. Formally, a Petri net (PN) is a 5-tuple $PN = (P, T, F, W, M)$, where

- $P = \{p_1, p_2, ..., p_m\}$ is a finite set of *places* (drawn as circles);
- $T = \{t_1, t_2, ..., t_n\}$ is a finite set of *transitions* (drawn as bars);
- $F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs connecting P and T;
- $W : F \to \{1, 2, 3, ...\}$ is weight function;
- $M : P \to \{0, 1, 2, ...\}$ is the *marking* which denote the number of tokens (drawn as black dots) in place P and the initial marking is denoted as M_0.

Graphically, Petri net is a directed graph with two disjoint types of nodes: *places* and *transitions*. A directed arc connecting a place (transition) to a transition (place) is called an input (output) *arc* of the transition. A positive integer called multiplicity can be associated with each arc. Places connected to a transition by input arcs are called the input places of this transition, and those connected by means of output arcs are called the output places. Each place may contain zero or more tokens. A transition is *enabled* if each of its input places has at least as many tokens as the multiplicity of the corresponding input arc. A

transition can *fire* when it is enabled, and upon firing, a number of tokens equal to the multiplicity of the input arc is removed from each of the input places, and a number of tokens equal to the multiplicity of the output arc is deposited in each of its output places. Therefore, the firing of a transition may transform a PN from one marking into another. With respect to a given initial marking M_0, the *reachability set* is defined as the set of all markings reachable through any possible firing sequences of transitions, starting from the initial marking [13, 30].

Fig. 1 is a simple example of a PN [13]. In Fig. 1 a, only transition t_1 is enabled because place P_1 contains two tokens, t_2 is disabled because P_2 is empty. If the firing takes place, one token is removed from input place P_1 and one token is deposited in both output places P_2 and P_3 (see Fig. 1 b). In Fig. 1 b, both transitions t_1 and t_2 are enabled. If t_1 fires, the PN will reaches Fig. 1 d, while Fig. 1 c will be reached if t_2 fires. The reachability set in this example is given by {(2, 0, 0, 0), (1, 1, 1, 0), (0, 2, 2, 0), (0, 0, 1, 1)}.

PN can be used to capture the behavior of many real-world situations including sequencing, synchronization, concurrency, and conflict. In computer networks, it can be used to describe and verify the communication protocols. However, the concept of time is not explicitly given in the original definition of Petri nets, while for the performance evaluation of dynamical systems, it is necessary and useful to introduce time delays associated with transitions in the Petri net models. This intuition has led to the emergence of stochastic Petri nets.

2.2 Stochastic Petri Nets

Stochastic Petri nets are Petri nets where exponentially distributed firing time is attached to each transition. In Generalized Stochastic Petri nets ($GSPN$) [23], transitions are allowed

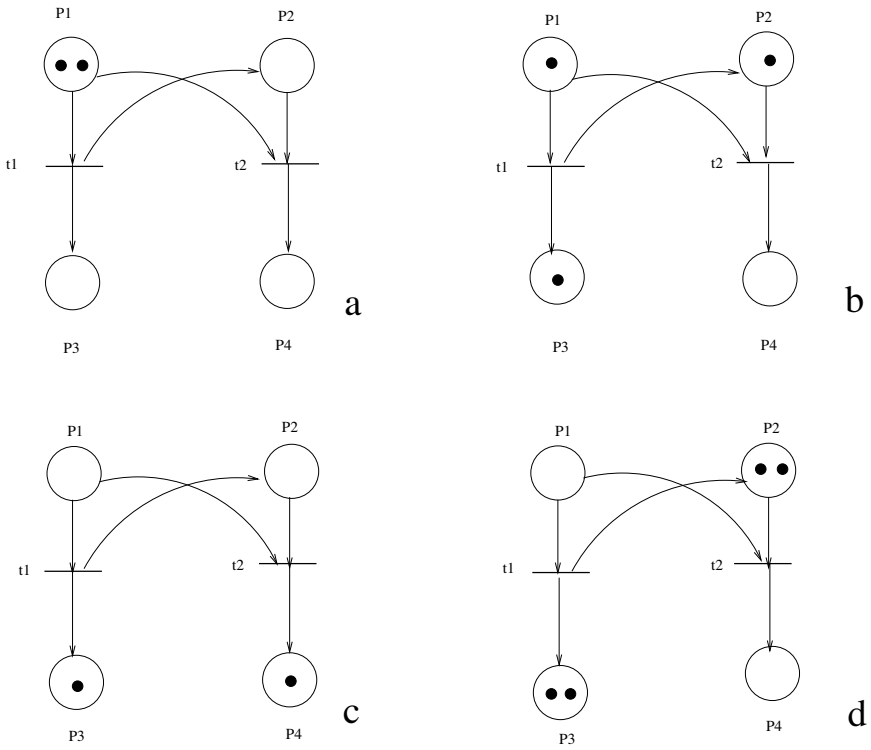


Figure 1: An Example of Petri Net

to be either *timed* (exponentially distributed firing time, drawn as rectangular boxes) or *immediate* (zero firing time, represented by thin black bars). Immediate transitions always have priority over timed transitions to fire. If several immediate transitions compete for firing, a specified probability mass function is used to break the tie.

A marking of a $GSPN$ is called *vanishing* if at least one immediate transition is enabled in the marking and *tangible* otherwise. $GSPN$ also introduces *inhibitor arc* connecting a place to a transition. Inhibitor arcs have small hollow circles instead of arrows at their terminating ends. A transition with an inhibitor arc can not fire if the input place of the inhibitor arc contains more tokens than the multiplicity of the arc.

It has been proved that exactly one CTMC corresponds to a given $GSPN$ under condition that only a finite number of transitions can fire in finite time with non-zero probability [23].

The $GSPN$ analysis can be decomposed into four steps [13]:

- Genarating the extended reachablility graph which contains the markings of the reachablity set as nodes and some stochastic information attached to the arcs, thereby all the markings are related to each other with stochastic information.
- Eliminating the vanishing markings with zero sojourn times and the corresponding transitions. This procedure generates the $CTMC$.
- Analyzing the steady-state, transient and cumulative behavior of the $CTMC$.
- Outputing the measures, such as the average number of tokens in each place and the throughput of each timed transition.

When SPN is applied to performance evaluation of computer networks, *places* can be used to denote the number of packets or cells in the buffer or the number of active users or

flows in the system, while the arrival and departure of packets, cells, users or flows can be represented by *transitions*.

Fig. 2 shows a $GSPN$ model of an $M/E_m/1/N$ queueing system. Transition T_1 represents the arrival of a customer with firing rate λ. An inhibitor arc with weight N from place P_{on} to T_1 represents the capacity of the queueing system.The transition T_1 is disabled when the number of tokens in the system equals N. The immediate transition t_1 fires when there is token in place P_{on} and P_{ser} is empty, and m tokens will be deposited in place P_{ser} after the firing of t_1.

Fig. 2 also shows the extended reachability graph of the $GSPN$ model, where (i, j) represents that the number of tokens in place P_{on} and P_{ser} are i and j, respectively. After the extended reachablility graph is generated, the vanishing markings will be deleted, which is mapped to a CTMC. In this example, there are N vanishing markings which are represented by dashed circles, i.e., (1,0), (2,0), ..., (N,0). Then the CTMC can be evaluated with well-known numerical solutions.

2.3 Stochastic reward nets

Stochastic reward nets ($SRNs$) are based on $GSPN$ but extend it further [11]. In SRN, every tangible marking can be associated with a reward rate. It can be shown that an SRN can be mapped into a Markov reward model. Thus a variety of performance measures can be specified and calculated using a very convenient formalism. SRN also allows several other features that makes specification convenient:

- Each transition may have an enabling function (also called a guard) so that a transition is enabled only if its marking-dependent enabling function is true. This feature provides a powerful means to simplify the graphical representation and to make SRNs

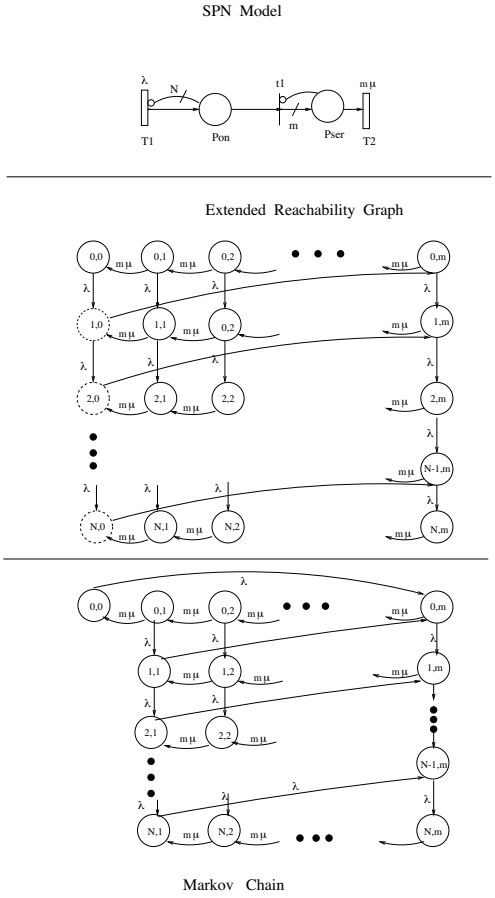


Figure 2: An Example of GSPN

easier to be understood.

- Marking dependent arc multiplicities are allowed. This feature can be applied when the number of tokens to be transferred depends on the current marking.
- Marking dependent firing rates are allowed. This feature allows the firing rate of the transitions to be specified as a function of the number of tokens in any place of the Petri net.
- Transitions can be assigned different priorities, and a transition is enabled only if no other transition with a higher priority is enabled.
- Besides the traditional output measures obtained from a $GSPN$, such as throughput of a transition and the mean number of tokens in a place, more complex reward function can be defined.

We will show in the following sections how to use these features to build concise models.

3 Example 1: SRN Model for Early Packet Discard in ATM Networks

3.1 Problem description

While ATM was originally conceived as a carrier of integrated traffic, the current most popular application of ATM is data communication where ATM networks are used as backbones to interconnect legacy Local Area Networks (LAN) and servers, and support legacy applications such as ftp, e-mail, and Web browsing. Such legacy applications operate on TCP/IP platform, which requires that ATM must support TCP/IP in the protocol stack.

Because of the popularity of TCP/IP, even some multimedia applications are based upon $TCP/IP/ATM$ platform. Therefore, TCP/IP and ATM will coexist and interoperate in the future.

When we run TCP/IP over ATM, TCP/IP packets are segmented at the ATM layer into fixed-size cells. When an ATM switch drops a cell because buffer overflows, the rest of the cells belonging to the same packet of the discarded cell will still be transmitted. After the cells arrive at the destination, the destination fails to reassemble the packet to which the lost cell belonged. TCP has a mechanism to request the source to retransmit the corrupted packet. Once one or more cells constituting a packet are lost, the whole packet will be retransmitted. Hence, the loss of one cell is amplified to the loss of a packet. And a portion of network resources are wasted in transmitting the corrupted and useless packets. This phenomenon was observed by Romanow and Floyd, and Early Packet Discard (EPD) was proposed in [4] to enhance the efficiency of TCP over ATM.

We assume that the buffer capacity is N (cells), and that the threshold of EPD algorithm is K (cells). From the description of EPD algorithm [20, 15] in Fig. 3, we can see that the packets arriving into the network face three eventualities: successfully transmitted, totally dropped when the first cell of the packet found the queue length exceeding the threshold, and partially discarded when some middle cell of the packet sees buffer overflow. We assume the total number of packets arriving during time period of duration t as $P_{in}(t)$, the number of successfully transmitted packets as $P_{out}(t)$, the number of totally dropped packets as $P_1(t)$, and the number of partially discarded packets as $P_2(t)$, then

$$P_{in}(t) = P_1(t) + P_2(t) + P_{out}(t) \tag{1}$$

```
when a cell arrives at an ATM switch:
if (this is the first cell of the packet) then
{
    if (queue length >= K) then discard the cell;
else accept the cell into the buffer;
}
else
{
    if (any cell of the packet has been discarded) then discard the cell;
    else
    {
        if (queue length == N) then discard the cell;
        else accept the cell into the buffer;
     }
 }
```

Figure 3: The EPD Algorithm

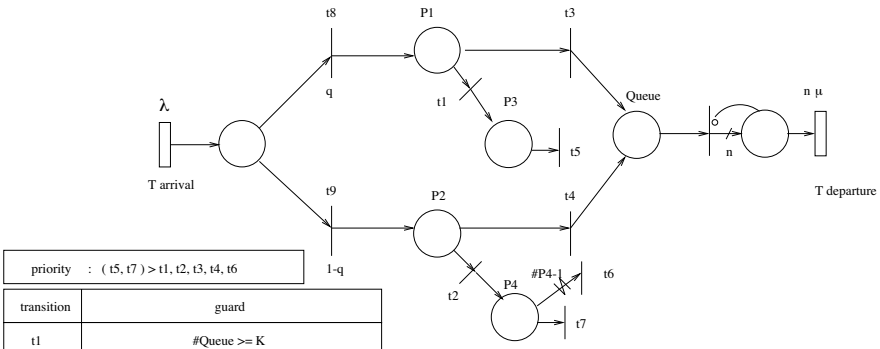
Then the effective throughput TH defined in [4] is:

$$TH = \lim_{t \to \infty} \frac{P_{out}(t)}{P_1(t) + P_{out}(t)} \tag{2}$$

The goodput G as defined in [34] is:

$$G = \lim_{t \to \infty} \frac{P_{out}(t)}{P_{in}(t)} \tag{3}$$

In order to get the TH and G, a series of complicated derivations were introduced in [34]. We will show in the following subsections that TH and G can be easily obtained by using SRN.



priority : (t5, t7) > t1, t2, t3, t4, t6	
transition	guard
t1	#Queue >= K
t2	#Queue = N or #P3 > 0 or #P4 > 0
t3	#Queue < K
t4	#Queue < N and #P3 = #P4 = 0
t5	#P1 > 0
t6	#P4 > 1
t7	#P1 > 0

guards for no-control syetem

transition	guard
t1	#Queue =N
t2	#Queue = N
t3	#Queue < N
t4	#Queue < N

Figure 4: Stochastic Reward Net Model for EPD

3.2 SRN model

We assume the cell arrival process is Poisson and the length of packet is geometrically distributed in terms of cells. The cell arrival rate is λ. An arriving cell is the head of the packet with probability q. Then the mean packet length and packet arrival rate are $\frac{1}{q}$ and λq, respectively. The behavior of the buffer can be described with the SRN model shown in Fig. 3.

The arrival of a cell is represented by timed transition $T_{arrival}$ with rate λ, and the departure of a cell is represented by transition $T_{departure}$. In ATM networks, the cell is transmitted with a deterministic delay. Because the SRN and underlying $CTMC$ are based on the exponential assumption of the distribution of the holding time in a given state, the deterministic delay can not be modeled by SRN directly. In this paper, we use an n-stage Erlang distribution whose rate is $n\mu$ to approximate it.

The firing of the immediate transition t_8 represents the event that the arriving cell is the head of a packet. If it finds the queue length (the number of tokens in place $Queue$) is less the threshold K, the immediate transition t_3 fires and the cell enters place $Queue$. The cell transmission time is deterministic, which is approximated by a 3-stage Erlang distribution with rate 3μ. If the head of the packet finds the queue length exceeding the threshold K, then it will be dropped, the immediate transition t_1 fires and one token is deposited in place P_3 which is used to remember that the entire packet is discarded.

The firing of the immediate transition t_9 represents the event that the arriving cell is not the head of a packet. If it finds buffer full, or the head of the packet has been discarded (i.e., there is one token in place P_3), or one of its predecessor cell in the same packet has been discarded (i.e., there is one token in place P_4), then the cell is discarded, otherwise it enters the buffer. If the cell is discarded, one token will be deposited in the place P_4. Immediate

```
double partial(){
    return (mark(''P3'')==0 && mark(''P4'')>0);
}
double total(){
    return (mark(''P3'')>0);
}
```

Figure 5: Reward function

transition t_6 and the variable cardinality of arc from P_4 to t_6 are used to guarantee the number of tokens in P_4 be less than 2, which is used to reduce the number of states of the system.

Once a new packet's head enters place P_1, the tokens in the places P_3 and P_4 are cleared by immediate transitions t_5 and t_7, which ensures that whether the current packet is discarded (damaged) or not has no relationship with the successive packet. The correct operation is guaranteed by giving t_5 and t_7 higher priority than t_3.

Obviously, any time we look at the state of the SRN in Fig. 4, a token in P_3 represents that the packet is totally discarded, and a token in P_4 represents that the packet is partially damaged. In other word, when the head of a packet arrives, the state of P_3 and P_4 gives the information about the previous packet's fate, successfully transmitted or totally discarded or partially damaged. SRN provides an index named *reward rate* to get the probabilities of the events we are interested in. Reward functions we use are shown in Fig. 5. According to the $PASTA$ (Poisson Arrivals See Time Averages) theorem [19], the reward function *partial( )* gives the probability of a packet being partially damaged, and *total( )* presents the probability of a packet being totally discarded. Then the performance indices TH and G can be obtained from the reward functions.

$$TH = \frac{1 - q \times partial() - q \times total()}{1 - q \times partial()} \tag{4}$$

$$G = 1 - q \times partial() - q \times total() \tag{5}$$

For SRN model of the queueing system without EPD control, we just need modify the guards of immediate transition t_1, t_2, t_3 and t_4 in Fig. 4 (see the table in the dashed rectangle).

4 Example 2: SRN model for $Ethernet/ATM$ bridge

4.1 Problem description

Consider an $Ethernet/ATM$ bridge, which has N *Ehternet* interface and one ATM interface. The packets arrival from one *Ehternet* interface follows a 2-state Markov Modulated Poisson Process ($MMPP$). Assume the traffic on the *Ehternet* interfaces are homogeneous, then the aggregation of the traffic may be described by an N-state $MMPP$. The aggregated packet arrival process is Poisson with rate r_i while it is at state i. The transitions between these states form a homogeneous, continuous time Markov chain. The rate from state i to $(i+1)$ is $i\delta$, to $(i-1)$ is $(N-i)\gamma$.

After the frames arrive into the buffer, they will be segmented into cells and each cell is serviced with a deterministic service time, i.e., 2.7 microseconds (we assume the port rate of the ATM interface is 155.520 Mb/s). From the previous measurements of real networks, it has been observed that the frame sizes of data traffic in Ethernet have three predominant values: 46 bytes (with probability c_1), 144 bytes (with probability c_2) and 1500 bytes (with probability c_3), which corresponds to 2 cells, 4 cells and 32 cells, respectively, after including

the $LANE$ (LAN Emulation) overhead (see [35, 25]). It is found that $c_1 = 0.342$, $c_2 = 0.093$, $c_3 = 0.565$.

These three different sizes are the consequence of three different application classes. Short frames are transmitted during terminal-to-host communication, whereas applications based on network-file system protocol (NFS) generate short frames in one direction followed by medium-sized frames in the reverse direction. The maximum frame size in Ethernet traffic is 1512 bytes, used mostly during file transfer applications. In other words, the data frames in the LAN consist of large packets, medium packets and small packets. This observation allows us to analyze and design the buffer more effectively.

4.2 SRN model

The traffic is modelled by an N-state MMPP, which is represented by the subnet in the dotted rectangle in Fig. 6. The firing rate of T_{on} depends on the number of tokens in place P_{on}, i.e., $(N - \#P_{on})\gamma$. The firing rate of T_{off} depends on the number of tokens in place P_{on} as well , i.e., $\#P_{on}\delta$. The firing rate of transition $T_{arrival}$ depends on the number of tokens in P_{on}, say $\lambda\#P_{on}$.

The frames generated according to the MMPP have three possible sizes, 2 cells (with probability c_1), 4 cells (with probability c_2), and 32 cells (with probability c_3). The split between the large, medium and small frame is represented by the immediate transitions t_1,t_2, and t_3. When a packet finds there is enough space in the buffer whose capacity is M cells, the packet enters the buffer, which is represented by immediate transitions t_7, t_8, and t_9. If the packet finds that there is not enough space in the buffer to accomodate the entire packet, the immediate transitions t_7, t_8, and t_9 are inhibited by inhibitor arcs from P_{buffer} to t_7, t_8, and t_9. Then the entire packet will be discarded, which is represented by

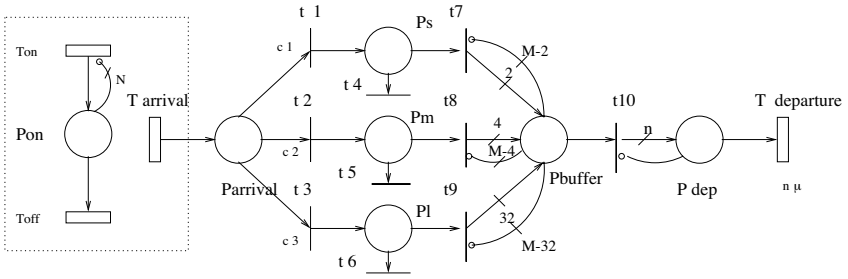


Figure 6: Stochastic Reward Net Model for Ethernet/ATM Bridge

Table 1: Rates and Guards for the Transitions in Fig. 6

transition	T_{on}	T_{off}	$T_{arrival}$	$T_{departure}$
rate	$(N - \#P_{on})\gamma$	$\#P_{on}\delta$	$\#P_{on}\lambda$	$n\mu$

immediate transitions t_4, t_5, and t_6. In this paper, discard is frame-based, not cell-based. Therefore, partial packet loss will not exist in our study, and loss amplification caused by partial packet loss is prevented [4]. The deterministic service time of the multiplexer, which can accomodate M cells, is approximated with an n-stage Erlang distribution.

The marking dependent firing rates for the timed transitions are listed in Table 1. The guards for the immediate transitions are listed in Table 2.

Table 2: Rates and Guards for the Transitions in Fig. 6

transition	t_4	t_5	t_6
guard	$\#\ P_{buffer} + sign(\#P_{dep}) > M - 2$	$\#\ P_{buffer} + sign(\#P_{dep}) > M - 4$	$\#\ P_{buffer} + sign(\#P_{dep}) > M - 32$

5 Recent Research on SPN

5.1 Tools development

Concurrent with the stochastic evolution of Petri nets, effort has been devoted to defining and implementing automatic solution tools for PN models. For example,

- *SPNP* (Stochastic Petri Net Package) and its *GUI* version *iSPN* can provide steady state, transient and cumulative measures, and advanced features such as marking-dependent rates, marking-dependent arc multiplicities and guards [10, 12].
- *GreatSPN* provides graphical input, steady state and transient solution to *GSPN* [8].
- *DSPNexpress* can deal with Deterministic and Stochastic Petri Nets (*DSPN*) under the assumption that only one deterministic transitions can be enabled[5, 6]. The solution to *DSPN* with two concurrently enabled deterministic transitions has been presented recently.
- *ESP* implements an approximate solution based on the use of phase type distributions [1].
- *UltraSAN* is based on a class of SPN models known as stochastic activity networks and provides steady state solution as well as transient solution [33].

5.2 Solutions Dealing with Largeness

Since *SPNs* are based on the solution of *CTMC*, the state space and reachable markings grows exponentially as the number of transitions, places and tokens increase in the *SPNs*, which poses a higher requirement to the memory and capacity of the computers,

and limits the applicability of $SPNs$ to deal with real life applications. For example, in the $Ethernet/ATM$ bridge, the number of state is about nMN. If we choose 4-stage Erlang to approximate the deterministic transition, and assume that the bridge has 8 Ethernet interfaces and the ATM port can accomodate 20,000 cells (i.e., about 1Mbytes), then the number of states is about 640,000. A large effort have been devoted to overcome or alleviate the *largeness*. For example, we can decompose the SPN model into multiple smaller models which can be solved separately [18]. Actually, the decomposition is based on representing the infinitesimal generator matrix in a compact form as a combination of smaller component matrices with Kronecker operators [29, 24]. However, sometimes the decomposition might not be very "clean", i.e., there are interactions among the submodels: the input parameters in submodel A depend on the behavior of submodel B, and the input parameters in submodel B depend on the behavior of the submodel A. In this case, fixed-point iteration can be used [31, 21]. For a proof of existence of a fixed point, see [31]. The proof of uniqueness, however, needs to be worked out on a case by case basis.

5.3 Solutions Dealing with Stiffness

The wide difference among the rates of the transitions in $SPNs$ is called stiffness, which will cause numerical difficulties while solving the Markov chain. In ATM networks, there are various processes operating on widely different time scales. For example, the cells may be transmitted within 3 microseconds with the ATM port rate is 155Mb/s, while the cells arrive in bursts whose length is at order of magnitude of millisecond, and the connection duration in ATM networks might be from several seconds to tens of minutes. Then if we want to study the performance of ATM networks at cell level, bursty level and connection level within one model, we have to deal with the stiffness in the model. One method to deal

with the stiffeness is based on hierarchical decomposition [18, 2], which decomposes the stiff model into multiple submodels so that there might be large difference in the magnitudes of the transition rates in different submodels, but the transition rates within each submodel do not differ significantly. The submodels can be solved separately and reward functions can be defined to get the final performance indices. Fixed-point iteration is still useful if the decompostion is not very "clean".

Besides the hierarchical decomposition, uniformization has been recently improved to handle stiff problems [17] .

5.4 Extensions to SRN

5.4.1 Non-Markovian Stochastic Petri Nets

The analytical tractability of $SRNs$ and underlying $CTMC$ are based on the exponential assumption of the distribution of the holding time in a given state. However, many activities in real life systems do not have an exponentially distributed duration, e.g., the transmission of cell in ATM networks has deterministic delay, self-similarity and heavy-tailed distributions were found in computer networks which demonstrates rather different properties from the Markovian processes [28, 22, 32]. Although non-exponential distribution can be approximated by phase type distributions, the price for introducing phase type distribution is an enlargement of the state space. In recent years, several classes of SPN models have been elaborated which incorporate some non-exponential characteritics in their definition [16, 7, 26, 3, 6].

5.4.2 Fluid Stochastic Petri Nets [14]

Fluid Stochastic Petri Nets ($FSPN$) extends the $SPNs$ by introducing real (positive) tokens to special continuous places. The places are partitioned into a set of discrete places containing an integer number of tokens and a set of fluid (or continuous) places containing a real fluid level. The state space of an $FSPN$ is partially discrete and partially continuous. The discrete part is an integer vector accounting for the number of tokens in the discrete places. The continuous part is a vector of real numbers accounting for the fluid levels in the continuous places. Conceptually, $FSPN$ is based on stochastic fluid flow models which have been used extensively to evaluate the performance of high-speed networks. In high-speed networks, the stochastic processes can be viewed as continuous-state as the network speed increases (e.g., the cell transmission delay is 3 microseconds in 155Mb/s ATM networks). Therefore $FSPN$ is very useful in high-speed networks. Besides, $FSPN$ can be used to deal with the problem of largeness.

Numerical analysis of FSPNs with single fluid place and discrete event simulation with multiple fluid places have been integrated in the $SPNP$ [9].

6 conclusion

As a high-level description language, $SPNs$ is very concise in its specification and its form is closer to a designer intuition about what a model should look like. Software packages developed by the researchers enable the automated genaration and solution of Markovian stochastic systems represented by $SPNs$. It really relieves the performance analyzer from the tedious task of building a Markov chains and solving linear equations. The aim of this paper is to encourage the researchers in communication area to use $SPNs$ as a tool of

performance analysis, therefore we omitted rigorous mathematical proofs for the procedures of automated generation and solution to $CTMC$. We focused on how to use $SPNs$ as a tool of performance analysis and two examples in ATM networks were presented and studied in this paper. We hope the "transition" from traditional methods to $SPNs$ will be "enable"ed eventually. We do believe the "transition" is "deterministic " , if not "immediate".

References

[1] A. Cumani, "ESP- A Package for the Evaluation of Stochastic Petri Nets with Phase-type Distributed Transition Times", *Proc. IEEE Int. Workshop on Timed Petri Nets*, pp.433-440, 1995.

[2] A. Bobbio, K. S. Trivedi, " An Aggregation Technique for the Transient Analysis of Stiff Markov Chain," *IEEE Trans. on Computers*, vol. 35, pp. 803-814, 1986.

[3] A. Bobbio, M. Telek, " A Benchmark for PH Estimation Algorithm: Results for Acyclic-PH", *Stochastic Models*, vol. 10, pp. 661-667, 1994.

[4] A. Romanow, S. Floyd, "Dynamics of TCP Traffic over ATM Networks", *IEEE J-SAC*, pp. 633-641, May 1995.

[5] C. Lindemann, "DSPNexpress: A Software Package for the Efficient Solution of Deterministic and Stochastic Petri Nets", *Performance Evaluation*, vol. 22, pp. 3-21, 1995.

[6] C. Lindemann, *Performance Modelling with Deterministic and Stochastic Petri Nets*, John Wiley, 1998.

[7] D. Logothetis, K. S. Trivedi, "Transient Analysis of the Leaky Bucket Rate Control Scheme Under Poisson and ON-OFF Sources", *Proceedings of the IEEE INFOCOM 94*, Toronto, Canada, June 1994.

[8] G. Chiola, " GreatSPN 1.5 Software Architecture", *Computer Performance Evaluation*, pp. 121-136, Elsevier Science Publisher, 1992.

[9] G. Ciardo, D. Nicol, K. S. Trivedi, "Discrete-event Simulation of Fluid Stochastic Petri Nets",*Proc. 7th Int. Workshop on Petri Nets and Performance Models (PNPM'97)*, pp. 217–225.

[10] G. Ciardo, J. Muppala and K. Trivedi, "SPNP: Stochastic Petri Net Package", *International Conference on Petri Nets and Performance Models*, Kyoto, Japan, December 1989.

[11] G. Ciardo, J. Muppala, K. S. Trivedi, "Analyzing Concurrent and Fault-tolerant Software Using Stochastic Reward Nets", *Journal of Parallel and Distributed Computing*, 15, p255-269, 1992.

[12] G. Ciardo, K. S. Trivedi, "Manual for SPNP: Stochastic Petri Net Package", version 5.0, CACC, ECE Department, Duke University, 1996.

[13] G. Bolch, S. Greiner and H. de Meer, Kishor S. Trivedi, *Queueing Networks and Markov Chains,*John Wiley, 1998.

[14] G. Horton, V. Kulkarni, D. Nicol, K. S. Trivedi, "Fluid stochastic Petri nets: Theory, application, and solution", *European Journal of Operations Research,* vol. 105, no. 1, pp. 184–201, Feb. 1998.

[15] H. Y. Li, K .Y. Siu, H. Y. Tzeng, C. Ikeda, H. Suzuki, "On TCP Performance in ATM Networks with Per-VC Early Packet Discard Mechanisms", *Computer Communications*, v.19, n.13, Nov 1996.

[16] H. Choi, V. Kulkarni, K. S. Trivedi, "Markov Regenerative Stochastic Petri Nets", *Performance Evaluation*, vol. 20, no. 1-3, pp. 337-357, 1994.

[17] J. Muppala, M. Malhotra, K. S. Trivedi,"Stiffness-Tolerant Methods for Transient Analysis of Stiff Markov Chains", *Microelectronics and Reliability,* vol. 34, no. 11, pp. 1825-1841, 1994.

[18] J. K. Muppala, K. S. Trivedi, "Composite Performance and Availability Analysis using a Hierarchy of Stochastic Reward Nets", *Proc. of the Fifth International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, Torino, 1991.

[19] K. S. Trivedi, *Probability and Statistics with Reliability, Queueing, and Computer Science Applications*, Prentice-Hall, Englewood Cliffs, NJ, 1982.

[20] K. Y. Sui, H. Y. Tzeng, "Performance of TCP over ATM with Time-Varing Available Bandwidth", *Computer Communications*, v.19, n.11, Sep. 1996.

[21] L. Tomek,K. S. Trivedi, "Fixed-Point Iteration in Availability Modeling", *Informatik-Fachberichte, Vol. 283: Fehlertolerierende Rechensysteme,* M. Dal Cin (ed.), pp. 229-240, Springer-Verlag, Berlin, 1991.

[22] M. E. Crovella, A. Bestavros, "Self-similarity in World Wide Web Traffic- Evidence and Possible Causes", *Sigcomm'96*, pp. 160-169.

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

[24] P. Kemper, "Numerical Analysis of Superposed GSPNs", *IEEE Trans. on Software Engineering*, vol. 22, 1996.

[25] Raif O. Onvural, *Asynchronous Transfer Mode Networks: Performance Issues,* Artech House, 1994.

[26] R. German, C. Lindermann, "Analysis of Stochastic Petri Nets by the Method of Supplementary Variables", *Performance Evaluation*, vol. 20, pp. 317-335, 1994.

[27] R. German, C. Kelling, A. Zimmerman, G. Hommel, " TimeNET - A Toolkit for Evaluating Non-Markovian Stochastic Petri Nets", *Performance Evaluation*, 1998

[28] S. Deng, "Empirical Model of WWW Document Arrivals at Access Link", *ICC'96*, pp1797-1802.

[29] S. Donatelli, " Superposed Stochastic Automata: A Class of Stochastic Petri Nets Amenable to Parallel Solution", *Performance Evaluation*, voo. 18, pp. 21-36, 1993.

[30] T. Murata, "Petri Nets: Properties, Analysis and Applications", *Proc. of IEEE*, vol. 77, no. 4, pp. 541-560, 1989.

[31] V. Mainkar, K. S. Trivedi, "Sufficient Conditions for the Existence of a Fixed Point in Stochastic Reward Net-Based Iterative Models", IEEE Trans. on Soft. Eng., vol. 22, Sept., pp. 640-653, 1996.

[32] V. Paxson, S. Floyd, "Wide Area Traffic: The Failure of Poisson Modeling," *IEEE/ACM Trans. on Networking*, pp. 227-444, June 1995.

[33] W. H. Sanders, W. D. Obal II, M. A. Qureshi, F. K. Widjanarko, "UltraSAN Modeling Environment", *Performance Evaluation*, 1998

[34] Y. Lapid, R. Rom, M. Sidi, "Analysis of Discard Policies in High-Speed Networks", *IEEE J-SAC*, pp. 764-777, June 1998

[35] "LAN Emulation over ATM Specification Version 1.0", *ATM Forum, af-lane-0021.000*, Jan. 1995.