Journal of Zhejiang University SCIENCE A
ISSN 1673-565X (Print); ISSN 1862-1775 (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

Stochastic individual predicate/transition nets[*]

Yao YUE[†], Chun-ming ZHANG, Hai-xin WANG, Guo-qiang BAI[†‡], Hong-yi CHEN
(*Institute of Microelectronics, Tsinghua University, Beijing 100084, China*)
[†]E-mail: yue-y05@mails.tsinghua.edu.cn; baigq@tsinghua.edu.cn

Received Apr. 8, 2008; Revision accepted June 5, 2008; Crosschecked Dec. 26, 2008

Abstract: We analyze the drawbacks of generally distributed time transition stochastic Petri nets (GDTT_SPN) in evaluating the performance of parallel systems, and propose a more general model, stochastic individual predicate/transition nets (SIPTN). SIPTN has higher modeling power and could provide more realistic models compared to GDTT_SPN, because in SIPTN the sojourn time distribution is determined not only by the transition, but also by the individuals. It is further proved that GDTT_SPN is a subset of SIPTN. As SIPTN introduces folding techniques from predicate/transition nets, SIPTN models have simpler and more intuitive graphic notations and accordingly higher usability, and thus are suitable for constructing simulation models for parallel systems.

Key words: Formal model, Parallel system, Performance analysis, Simulation, Stochastic Petri nets
doi:10.1631/jzus.A0820268 **Document code:** A **CLC number:** TN47; TP301.2

INTRODUCTION

In the research field of performance evaluation, two main methods are often adopted. One method is to use queues and queuing networks, while the other is to use generally distributed time transition stochastic Petri nets (GDTT_SPN) (Ajmone Marsan *et al*., 1998; Herzog, 2002). Queuing theory has a long history and is deeply studied (Jain *et al*., 2006; Zimmermann, 2007), but when blocking and synchronization are important characters to be modeled, which is the case in parallel systems, descriptions using queuing networks are not natural. In contrast, the underlying Petri nets (PN) of GDTT_SPN could offer a language in which synchronization, blocking and splitting are native in their formalism and semantics. Moreover, some results from PN research could easily be applied to the qualitative analysis of GDTT_SPN models after modification (Ajmone Marsan *et al*., 1998). GDTT_SPN incorporates nearly all existing methods that combine PN and stochastic processes to evaluate system performance, such as stochastic Petri nets (SPN), generalized stochastic Petri nets (GSPN), semi-Markov SPN, phase type SPN, and Markov regenerative SPN (Ajmone Marsan *et al*., 1998). This paper shows that although some simple models, such as SPN and GSPN, exhibit good analytical properties, they lack ability in modeling parallel systems. It is further shown that even GDTT_SPN is inadequate for modeling complex parallel systems. Therefore, stochastic individual predicate/transition nets (SIPTN), a superset of GDTT_SPN, is proposed in this paper to solve the problems in GDTT_SPN.

DRAWBACKS OF GDTT_SPN

PN was originally developed and used for the study of the qualitative properties of systems exhibiting concurrency, synchronization and conflicts. The simplest PN has four elements: places, transitions, arcs and an initial marking, which are often represented by circles, rectangles, directed arcs either from places to transitions or from transitions to places, and a distribution of tokens in the places in its graphic

[‡] Corresponding author
[*] Project supported by the National Natural Science Foundation of China (No. 60576027), and the Hi-Tech Research and Development Program (863) of China (No. 2006AA01Z415)

notation. Inhibitor arc, arc weight and place capacity are often added to enhance the modeling capacity of PN. A typical PN could be formally defined as follows:

A PN is an 8-tuple: $PN=(P, T, F, H, W, W_I, K, M_0)$, where $P=\{p_1, p_2, \ldots, p_n\}$ is the place set; $T=\{t_1, t_2, \ldots, t_m\}$ is the transition set; $P\cap T=\varnothing$, $P\cup T\neq\varnothing$; $F\subseteq(P\times T)\cup(T\times P)$ is the arc set; $H\subseteq P\times T$ is the inhibitor arc set; $F\cap H=\varnothing$; W: $F\rightarrow\mathbb{N}$ is the weight function for arcs; W_I: $I\rightarrow\mathbb{N}$ is the weight function for inhibitor arcs; K: $P\rightarrow\mathbb{N}^*\cup\{+\infty\}$ is the capacity function for places; $M_0=(m_{0,1}, m_{0,2}, \ldots, m_{0,n})$ is the initial marking, and $\forall p\in P$, $M_0(p)\leq K(p)$.

Transition t is firable if and only if: (1) $\forall p\in P$, $W(p, t)\leq M(s)\leq K(p, t)-W(p, s)$, which means that enough tokens exist at each input place of t and that all output places of t have enough capacity to accommodate the extra tokens added by t; (2) $\forall p\in P$, $M(s)<W_I(p, t)$, which means that t is not disabled by inhibitor arcs.

In order to use PN-based techniques for performance evaluation, it is necessary to introduce temporal specifications within the basic, untimed models. Associating sojourn time probability density function (pdf) with transitions is the most common method for incorporating time into PN. If the sojourn time pdf could be general, even degenerate to allow constant time delays, this type of timed PN is referred to as GDTT_SPN (Ajmone Marsan *et al*., 1998). In GDTT_SPN, immediate transitions, whose sojourn time pdf degenerates to Dirac function $\delta(t)$, have higher priority than other transitions, and are represented by black bars.

Ajmone Marsan *et al*.(1989) pointed out the importance of memory policy on the behavior of SPN. In GDTT_SPN, three memory policies are considered (Ajmone Marsan *et al*., 1998):

(1) Resampling: The timer of the transition is reset to a new value at any change in the marking. The new value is sampled from the sojourn time pdf associated with the transition.

(2) Enabling: If the transition is still enabled in the new marking, the value of the timer is kept; it is reset to a new value if the transition is disabled.

(3) Age: The timer value is kept, whether the transition is still enabled or not in the new marking.

A GDTT_SPN is a 10-tuple: $GDTT\_SPN=(P, T, F, H, W, W_I, K, M_0, \omega, \varepsilon)$, where $(P, T, F, H, W, W_I, K, M_0)$ is the underlying PN; $\omega\rightarrow\{pdf\}$ is the operator to assign sojourn time pdfs for transitions; ε: $T\rightarrow$\{Resampling, Enabling, Age\}, where Resampling, Enabling and Age stand for different memory policies.

GDTT_SPN could easily be used to model system architectures for parallel systems. The GDTT_SPN model of the high performance network security accelerator (HPNSA) proposed in (Wang *et al*., 2007) serves as an example. This example will be used in further discussion.

In the GDTT_SPN model of HPNSA shown in Fig.1, the sojourn time pdf type associated with each transition is provided. In this model, tokens represent data packages, places represent storage units such as FIFO (first-in-first-out) and DMA (direct memory access), and transitions represent events including task arrival, bus transfer and data processing. GEN is a place for all tokens waiting to be processed and RET is a place for all tokens that have been processed by HPNSA. New task arrivals are simulated by transition

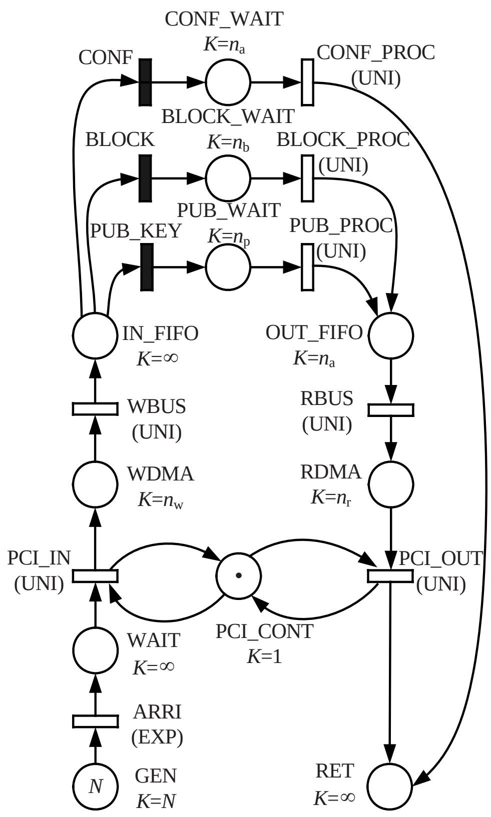

Fig.1 GDTT_SPN model of HPNSA (Wang *et al*., 2007)
UNI: uniform distribution; EXP: exponential distribution; n_a, n_b, n_p, n_r, n_w, and N: the capacity of CONF_WAIT, BLOCK_WAIT, PUB_WAIT, RDMA, WDMA, and GEN, respectively

ARRI, whose sojourn time obeys exponential distribution, so that the arrival of new tasks obeys the Poisson process, which is often the truth. PCI_IN, PCI_OUT and PCI_CONT simulate the bidirectional PCI BUS, while WBUS and RBUS simulate unidirectional data buses. WDMA and RDMA serve as input and output caches respectively. Immediate transition CONF, BLOCK and PUB_KEY are added to distinguish three different branches: configuration, block cipher processing, and public-key cipher processing. For configuration no result is required to be sent out, and the tokens are directly transferred back to RET after processing by CONF_PROC. Block cipher algorithms and public-key cipher algorithms are separated because of their different processing rates. As HPNSA is a parallel computing system, multiple-server semantics is used for CONF_PROC, BLOCK_PROC and PUB_PROC.

The simplest type of GDTT_SPN is SPN, which uses exponential distribution for all transitions. Because if the pdf associated with X_i is exponential:

$$f_i(x)=\begin{cases}\lambda_i \mathrm{e}^{-\lambda_i x}, & \text{for } x\ge 0,\\ 0, & \text{otherwise,}\end{cases}$$

where $i\in\mathbb{N}^*$, the following two formulae hold:

$$P\left\{\min_{i=1,2,\ldots,n} X_i \le x\right\}=\begin{cases}1-\exp\left(-\sum_{i=1}^{n}\lambda_i x\right), & \text{for } x\ge 0,\\ 0, & \text{otherwise,}\end{cases}$$

$$P\{X_i-X_j\le x \mid X_i>X_j\}=\begin{cases}1-\mathrm{e}^{-\lambda_i x}, & \text{for } x\ge 0,\\ 0, & \text{otherwise.}\end{cases}$$

The above two formulae show that: (1) the sojourn time in any marking M is also exponential; (2) it makes no difference which memory policy is assumed since once a transition is fired, the residual sojourn time pdfs of all other transitions are the same as their original pdfs. SPN models have good analytical properties because their marking processes prove to be continuous time Markov chains (CTMC) (Molloy, 1981; Balbo, 2001).

GSPN extends SPN by allowing for immediate transitions and priorities. GSPN models also exhibit good analytical properties because their marking processes prove to be stochastic point processes (SPP) (Ajmone Marsan *et al*., 1984; Balbo, 2001).

In spite of their elegant analytical properties, SPN and GSPN have limited applications in performance evaluation due to the fact that actual systems rarely satisfy the condition that the sojourn time pdfs of all timed transitions obey exponential distributions. This problem is especially severe when modeling parallel systems because it might give misleading results. Consider the GDTT_SPN model in Fig.1 for example. If the sojourn time of all transitions is assumed to obey exponential distribution, the GDTT_SPN model reduces to an SPN model. The analytical results of the SPN model show that it makes no difference to use one bidirectional data bus to replace WBUS and RBUS and, what is more, using DMAs as caches decreases rather than increases the data transfer rate. The SPN model produces these absurd results because, due to the memoryless property of exponential distribution, the timed transition could be viewed as fired in serial. Therefore, although no place like PCI_CONT is used between WBUS and RBUS, the two unidirectional data buses act as just one bidirectional data bus. The key to the second absurd result is that as adding DMAs brings in more timed transitions, namely WBUS and RBUS, the firing probability for each transition is reduced. This leads to a decrease in the system throughput. From the analysis above, it is apparent that both SPN and GSPN lack power in modeling parallel systems due to their serial nature.

Now return to the GDTT_SPN model of HPNSA in Fig.1. To model parallel systems, the memory policy must allow concurrency among timed transitions, and thus Enabling and Age memory policies serve as candidates. Since in the HPNSA model no other transition will be disabled when a transition finishes its execution, Enabling policy and Age policy are equivalent for this model. As concurrency could be properly modeled in GDTT_SPN, it at least casts some light on performance evaluation. However, the GDTT_SPN model is very hard, if not impossible, to solve. Furthermore, the sojourn time of the same token for different timed transitions is irrelevant, which contradicts with reality and might lead to erroneous results, especially in transient analysis. For example, the sojourn time of a token for PCI_IN and the sojourn time of the same token for WBUS are independent in the model in Fig.1, but in reality, if a data package sojourns 200 clock cycles in PCI_IN, it must sojourn 133 clock cycles in WBUS, because PCI BUS and WBUS work under clock frequencies 133 MHz and 200 MHz respectively, and both of the two buses are 64 bits in width and do not allow transfer interruption.

In the real HPNSA system, once a task is generated its sojourn time in each transition could be readily determined by its processing type and data length. Therefore, the randomicity in HPNSA arises from the random type and random data length of the arriving tasks or, in other words, the random characteristics of tokens. But in GDTT_SPN, randomicity is purely modeled with the random sojourn time associated with each transition. Therefore, the GDTT_SPN model in Fig.1 is far from accurate. Assume only one stochastic task is being modeled and the data length obeys uniform distribution. The probability that its data length is very short is much greater than the probability that the sampled sojourn time of all transitions is very short. Thus, it is problematic to assume that the sojourn time of each transition obeys uniform distribution just because the length of the task data obeys uniform distribution, although if a large number of tasks are being modeled, this assumption is close to reality. The fact that reasonable sojourn time pdfs depend on the initial marking will bring great trouble to the modelers.

To obtain more accurate results, GDTT_SPN must use a more complicated structure because it could only model "token characteristics" by separating tokens structurally, which is awkward and boring. To simplify the discussion, only the input part of HPNSA is considered to further illustrate this point and the coupling between PCI_IN and PCI_OUT is ignored. Fig.2 shows the original model and the refined model of the input part of HPNSA when n_1 tokens of data length l_1 and n_2 tokens of data length l_2 ($l_1 \neq l_2$) are to be transferred. In the original model, the sojourn time pdf of PCI_IN is $n_1\delta(t-t_1)/(n_1+n_2)+n_2\delta(t-t_2)/(n_1+n_2)$ and the sojourn time pdf of WBUS is $n_1\delta(t-0.665t_1)/(n_1+n_2)+n_2\delta(t-0.665t_2)/(n_1+n_2)$, where $t_1=l_1/R_{\text{PCI\_tr}}$, $t_2=l_2/R_{\text{PCI\_tr}}$, and $R_{\text{PCI\_tr}}$ is the PCI transfer rate. It is apparent that the sojourn time pdf type is closely related with the tasks to be processed, and that even if all sojourn time pdfs are properly set, the model could give us many different results on overall processing time, among which only one equals the actual processing time.

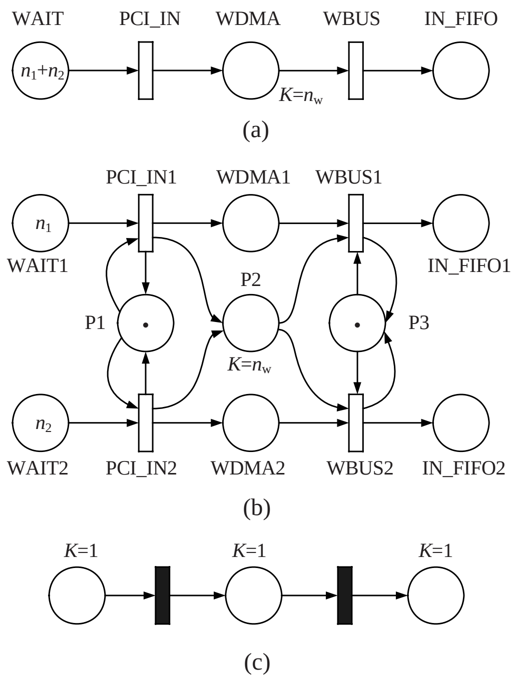

Fig.2 Refinement of the input part of HPNSA
(a) Original model; (b) Refined model; (c) Further refinement of WDMA when the capacity of P2, n_w, equals 3

Therefore, the original model in Fig.2a is insufficient when high modeling accuracy is required. According to the discussion above, GDTT_SPN must use extra structure to separate different tokens, and this will result in the refined model in Fig.2b. This refined model contains not only two copies of each resource to separate the different tokens, but also three extra places, namely P1, P2 and P3, to obtain proper system behavior. P1 and P3 are added to guarantee that PCI_IN and WBUS transfer only one data package at a time, and the capacity of WDMA is modeled with the capacity of P2. The sojourn time pdfs of PCI_IN1, PCI_IN2, WBUS1, and WBUS2 are $\delta(t-t_1)$, $\delta(t-t_2)$, $\delta(t-0.665t_1)$ and $\delta(t-0.665t_2)$, respectively. In this model all sojourn time pdfs are irrelevant to the initial marking, and the whole processing time given by this model exactly equals the actual processing time. This model could also be further refined by allowing for arbitration policies in WDMA, such as first-in-first-out realized as shown in Fig.2c, which could easily be incorporated into the two models shown in Figs.2a and 2b.

Although the refined model in Fig.2b has satisfactory accuracy, the refinement process increases the number of places from 3 to 9, the number of transitions from 2 to 4, and the number of arcs from 4 to 20. What is more, this refined model considers only two types of tokens. If more types are considered and the whole HPNSA system is being modeled, it is easy to imagine that the resulting model will be very complex, which brings trouble to modelers.

From the discussion above, it could be concluded that:

(1) Although the original model in Fig.2a is simpler and more intuitive, the sojourn time pdf type is closely related to the tasks to be processed, and most of its feasible states could not be reached in reality.

(2) The refined model in Fig.2b could gives satisfactory results, but it is much more complex and less intuitive compared to the original model.

Therefore, both these two models bring problems to the modelers, and it is meaningful to find an extension of GDTT_SPN that could keep the model simple while obtaining good modeling accuracy. This topic will be discussed in the next section.

INFORMAL INTRODUCTION TO SIPTN

The fundamental factor that complicates the refined model is that GDTT_SPN is awkward when the randomicity of tasks is being modeled. It is forced to use copies of the same structure to separate different tasks, and in turn required to introduce extra resources to control the several copies so as to achieve the desired unified behavior of the whole model. Therefore, a reasonable extension of GDTT_SPN should add new mechanisms that are flexible in distinguishing different tasks. As GDTT_SPN is very suitable for modeling the uncertainty of the sojourn time in transitions, the extension should inherit this merit. In conclusion, the extension should be flexible in modeling two types of randomicity: the uncertainty of sojourn time in transitions, and the randomicity in token characteristics.

Enlightened by the idea of high level Petri nets (HLPN) (Jensen, 1996; Genrich and Lautenbach, 1979) and stochastic high level Petri nets (SHLPN) (Lin and Marinescu, 1988), it is reasonable to introduce new semantics to fold the refined model in Fig.2 to eliminate the extra copies of the same resources. In this way the extra controlling places P1, P2 and P3 are not needed any more, and the folded refined model looks somewhat like the original model after folding. To emphasize token characteristics, predicate/ transition nets (Pr/T nets) (Genrich and Lautenbach, 1979) is chosen to be the underlying net. In Pr/T nets, tokens could carry their own characteristics and become “individuals”, while places represent changing properties of, or relations between, individuals and become “predicates” with variable extension.

SIPTN extends Pr/T nets to accommodate temporal features in GDTT_SPN. In SIPTN the transitions are divided into normal transitions (NTs), which are represented with rectangles, and property assigning transitions (PATs), which are represented with double-edged rectangles. PATs are responsible for allocating the sojourn time information on NTs of individuals. The sojourn time of an individual on a PAT is solely determined by the sojourn time pdf associated with the transition, just as transitions in GDTT_SPN. When a PAT sends out an individual, it randomly updates the sojourn time pdfs of the individual on all NTs based on its sampling result on a probability space. An NT resolves the final sojourn time pdf from the sojourn time pdfs associated with this transition for all input individuals. In this way, both the influence of individual characteristics and the influence of sojourn time uncertainty could easily and properly be modeled for NTs. PATs are useful when modeling random individual arrival processes.

The SIPTN model for HPNSA, shown in Fig.3, is constructed as an example to illustrate the concept of SIPTN. This model differentiates configuration and crypto processing before they arrive, because the data length of configuration is much shorter than that of crypto processing. What is more, the “guard” at the two immediate transitions guarantees that individuals go to the right branch, and choosing the right probability space associated with OP_ARRI could bifurcate block cipher processing and public-key cipher processing. Another merit of this SIPTN model is that CONF_ARRI and OP_ARRI could assign many different, even infinite types of sojourn time information to individuals, but this does not complicate the graphic notation at all. As the individual “<a>” is merely responsible for PCI BUS control, its sojourn time information is ignored by both PCI_IN and PCI_OUT when resolving the final sojourn time pdfs.

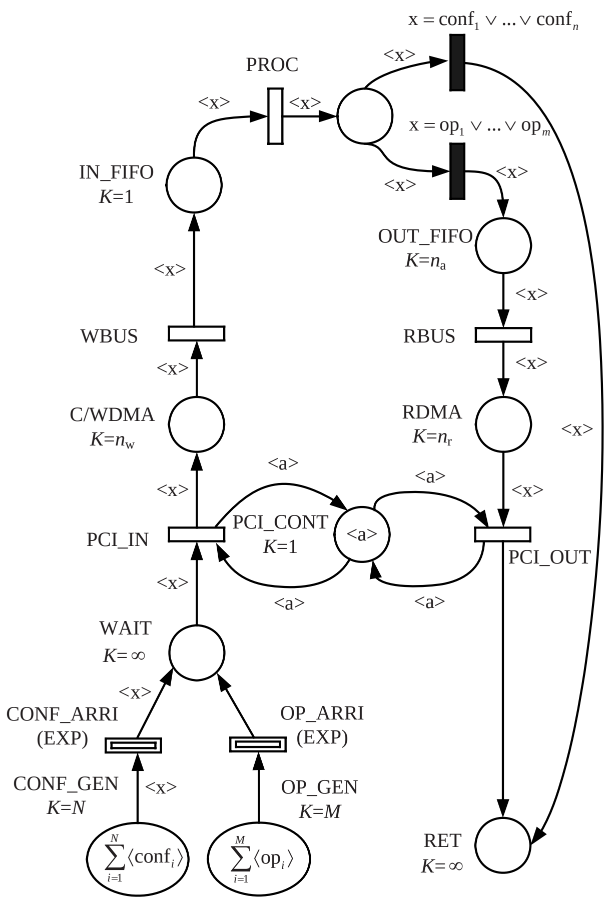

Fig.3 SIPTN model for HPNSA

n_a, n_b, n_p, n_r, n_w and N: the capacity of CONF_WAIT, BLOCK_WAIT, PUB_WAIT, RDMA, WDMA, and GEN, respectively. <conf$_i$> (i=1, 2, …, N), <op$_i$> (i=1, 2, …, M) and <a> represent individuals, and <x> could represent either <conf$_i$> or <op$_i$>

FORMAL DEFINITION OF SIPTN

An SIPTN is a 17-tuple: $SIPTN=(P, T, F, H, D, V, A_P, A_T, A_F, A_I, K, M_0, P_a, P_n, A_l, D_e, \varepsilon)$, where

- $P=\{p_1, p_2, \ldots, p_n\}$ is the predicate (first order place) set
- $T=T_n \cup T_a$ is the transition set satisfying $T_n \cap T_a=\varnothing$
- $T_n=\{t_{n1}, t_{n2}, \ldots, t_{nm}\}$ is the normal transition set
- $T_a=\{t_{a1}, t_{a2}, \ldots, t_{aq}\}$ is the property assigning transition set
- $P \cap T=\varnothing$, $P \cup T \neq \varnothing$
- $F \subseteq (P\times T)\cup(T\times P)$ is the arc set
- $H \subseteq P\times T$ is the inhibitor arc set, and $F \cap H=\varnothing$
- D is a finite, nonempty set. It is called the individual set of SIPTN. Ω is defined as the set of operators for D
- V is the variable set defined over D
- A_P: $P\rightarrow\Pi$, where Π denotes the variable predicate set on D. $\forall p\in P$, if $A_P(p)$ is an n-ary predicate, then p is called "n-ary predicate"
- A_T: $T\rightarrow f_D$, where f_D is the formula set on D. $\forall t\in T$, $A_T(t)$ could contain only static predicates and operators in Ω
- A_F: $F\rightarrow f_s$, where f_s is the symbolic sum set on D. $\forall p\in P$, if $(t, p)\in F$, or $(p, t)\in F$, $A_F(t, p)$ or $A_F(p, t)$ is an n-ary symbolic sum; otherwise, $A_F(t, p)=A_F(p, t)=\varnothing$
- A_I: $I\rightarrow f_s$. $\forall p\in P$, if $(p, t)\in I$, $A_I(p, t)$ is an n-ary symbolic sum; otherwise, $A_I(p, t)=\varnothing$. $\forall t\in T$, free variables in formula $A_T(t)$ must be free variables on arcs or inhibitor arcs connected to t
- K: $P\rightarrow\mathbb{N}^*\cup\{+\infty\}$ is the capacity function, where $\forall p\in P$, $K(p)$ is the maximum number of aries for predicate p
- M_0: $P\rightarrow f_s$ is the initial marking, which satisfies: (1) $\forall p\in P$, $M_0(p)$ is an n-ary predicate; (2) $\forall p\in P$, $|M_0(p)|\leq K(p)$
- P_a: $T_a\rightarrow\{pdf\}$, $t_{ai}\mapsto pdf_i^{(a)}$ is the sojourn time pdf allocation operator for PATs, where $pdf_i^{(a)}$ is fixed $(1\leq i\leq q)$
- P_n: $D\rightarrow\{pdf\}^m$, $d\mapsto(pdf_{d,1}^{(n)}, pdf_{d,2}^{(n)}, \ldots, pdf_{d,m}^{(n)})$ memorizes the sojourn time pdfs of individual d for all NTs. P_n could be modified by A_l
- A_l: $T_a\rightarrow R$, $R=\{r_1, r_2, \ldots, r_q\}$, where r_i is a random variable defined on the probability space $\{\{D\rightarrow\{pdf\}^m\}, F, P\}$. When individual d is sent out by the jth firing of t_{ai}, $P_n(d)$ is redefined as $Alloc_i(j)$ $(1\leq i\leq q)$
- D_e: $T_n\rightarrow S$, $S=\{s_1, s_2, \ldots, s_m\}$, where s_i: $\{pdf\}^k\rightarrow pdf$ determines the final sojourn time distribution for T_{ni} $(1\leq i\leq m)$, and k stands for the number of input individuals of the transition T_{ni}
- ε: $T\rightarrow$\{Resampling, Enabling, Age\}, where Resampling, Enabling and Age stand for different memory policies

For SIPTN, $(P, T, F, H, D, V, A_P, A_T, A_F, K, M_0)$ is the underlying Pr/T net (Genrich and Lautenbach, 1979).

FURTHER DISCUSSION

The following two theorems show that SIPTN has higher modeling power than GDTT_SPN.

Theorem 1 GDTT_SPN is a subset of SIPTN.

Proof For a GDTT_SPN, if its underlying PN is viewed as a simple Pr/T net, it is an SIPTN with $T=T_a$.

Theorem 2 When the probability space of all PATs is finite in an SIPTN, it is isomorphic with a GDTT_SPN.

Simple Proof Let $T_a=\{t_{a1}, t_{a2}, \ldots, t_{aq}\}$, and the number of elements in the probability space associated with t_{ai} be $f(t_{ai})$. Define the GDTT_SPN obtained by replacing predicates with places, and all transitions with the basic transition in SIPTN as the basic structure of SIPTN. Then, using the refining technique in Fig.2 to control $f(t_{a1})f(t_{a2})\cdots f(t_{aq})$ copies of basic structures could give the GDTT_SPN model that is isomorphic with the original SIPTN model.

When at least one of these probability spaces is infinite, the above technique could not work any more, and generally this SIPTN could not be reduced to GDTT_SPN.

Although SIPTN has high modeling power, general SIPTN models are very difficult to solve analytically because its subset, GDTT_SPN models, are already difficult to solve (Ajmone Marsan *et al*., 1998). Therefore, simulation methods should be adopted to do quantitative analysis, and results from Pr/T nets could easily be used to do qualitative analysis, although some results should be modified because the introduction of time may change some qualitative properties of the underlying Pr/T nets. It is more reasonable to use complicated simulation models such as SIPTN to give design directions for complex parallel systems rather than to use oversimplified models, such as the SPN/GSPN model for HPNSA, to obtain analytical results. It is meaningful to develop general SIPTN simulation software, which could not only help designers access and compare the performance of different system architectures prior to the actual design, but also facilitate the search for the optimum design parameters in order to achieve the required performance at the lowest cost.

CONCLUSION

SIPTN, as a superset of GDTT_SPN, has higher modeling ability, more intuitive graphic notations, and could provide more realistic models. Therefore, it is very suitable for constructing simulation models for parallel systems. Developing general SIPTN simulation software is meaningful work as it could aid performance evaluation in behavioral level design. One method for further increasing the modeling accuracy is to introduce queuing policy to SIPTN.

References

Ajmone Marsan, M., Conte, G., Balbo, G., 1984. A class of generalized Petri nets for the performance evaluation of multiprocessor systems. *ACM Trans. Comput. Syst.*, **2**(2):93-122. [doi:10.1145/190.191]

Ajmone Marsan, M., Balbo, G., Bobbio, A., Chiola, G., Conte, G., Cumani, A., 1989. The effect of execution policies on the semantics and analysis of stochastic Petri nets. *IEEE Trans. Software Eng.*, **15**(7):832-846. [doi:10.1109/32.29483]

Ajmone Marsan, M., Bobbio, A., Donatelli, S., 1998. Petri nets in performance analysis: an introduction. *LNCS*, **1491**:211-256.

Balbo, G., 2001. Introduction to stochastic Petri nets. *LNCS*, **2090**:117-148. [doi:10.1007/3-540-44667-2_3]

Genrich, H.J., Lautenbach, K., 1979. The analysis of distributed systems by means of predicate/transition-nets. *LNCS*, **70**:123-146. [doi:10.1007/BFb0022467]

Herzog, U., 2002. Formal methods for performance evaluation. *LNCS*, **2090**:1-37. [doi:10.1007/3-540-44667-2_1]

Jain, J.L., Mohanty, S.G., Böhm, W., 2006. A Course on Queueing Models. Chapman & Hall/CRC, London, New York.

Jensen, K., 1996. Colored Petri Nets: Basic Concepts, Analysis Methods and Practical Use (2nd Ed.). Volume 1, Springer-Verlag Berlin, New York, p.65-85.

Lin, C., Marinescu, D.C., 1988. Stochastic high level Petri nets and applications. *IEEE Trans. Comput.*, **37**(7):815-825. [doi:10.1109/12.2227]

Molloy, M.K., 1981. On the Integration of Delay and Throughput Measures in Distributed Processing Models. PhD Thesis, UCLA, Los Angeles, CA.

Wang, H.X., Yue, Y., Zhang, C.M., Bai, G.Q., Chen, H.Y., 2007. A Novel Unified Control Architecture for a High-performance Network Security Accelerator. Proc. Int. Conf. on Security and Management, p.538-544.

Zimmermann, A., 2007. Stochastic Discrete Event Systems: Modeling, Evaluation, Applications. Springer, Berlin Heidelberg New York, p.65-78.