Reachability Analysis of Real-Time Systems Using Time Petri Nets

Jiacun Wang, Yi Deng, *Member, IEEE*, and Gang Xu

***Abstract*—Time Petri nets (TPNs) are a popular Petri net model for specification and verification of real-time systems. A fundamental and most widely applied method for analyzing Petri nets is reachability analysis. The existing technique for reachability analysis of TPNs, however, is not suitable for timing property verification because one cannot derive end-to-end delay in task execution, an important issue for time-critical systems, from the reachability tree constructed using the technique. In this paper, we present a new reachability based analysis technique for TPNs for timing property analysis and verification that effectively addresses the problem. Our technique is based on a concept called *clock-stamped state class* (CS-class). With the reachability tree generated based on CS-classes, we can directly compute the end-to-end time delay in task execution. Moreover, a CS-class can be uniquely mapped to a traditional state class based on which the conventional reachability tree is constructed. Therefore, our CS-class-based analysis technique is more general than the existing technique. We show how to apply this technique to timing property verification of the TPN model of a command and control (C2) system.**

***Index Terms*—Reachability analysis, real-time systems, real-time verification, time Petri nets.**

I. Introduction

A REAL-TIME system is one whose logical correctness is based both on the correctness of the outputs and on their timeliness [8], [10]. It must satisfy explicit (bounded) response-time constraints or risk severe consequences, including system failure. Consequently, a key requirement to real-time systems is the end-to-end delay in task execution, a critical issue in the design and analysis of these time critical systems [9], [14], [15]. This paper aims to address the timing property analysis and verification of real-time systems.

Real-time systems, such as aircraft navigation, command and control, and power plant monitoring systems, are often reactive or embedded control systems that must constantly react to their environment and interact among components within the systems. Consequently, concurrency, resource sharing, synchronization, and deadlock resolution are among essential issues in the design and implementation of such systems. Petri nets, for their ability to model these properties, become a suitable model for modeling and analysis of this class of systems [3], [12]. Several extended models of Petri nets were proposed to deal with the timing issues [16]. These models include timed Petri nets [13], [20], stochastic timed Petri nets [7], and time Petri nets (TPNs) [11]. Among these models, TPNs are a most widely used model for real-time system specification and verification [3], [6], [15], [17], [18]. In TPNs, event synchronization is represented in terms of a set of pre- and post-conditions associated with each individual action of the modeled system, and timing constraints are expressed in terms of minimum and maximum amount of time elapsing between the enabling and the execution of each action. This allows a compact representation of the state space and an explicit modeling of concurrency and parallelism.

A fundamental and most widely applied method for analyzing Petri nets' models, like for many other formal models, is reachability analysis. It permits the automatic translation of behavioral specification models into a state transition graph made up of a set of states, a set of actions, and a succession relation associating states through actions [3], [5]. This representation makes explicit such properties as deadlock freedom and reachability [19], and allows the automatic verification of ordering relationships among task execution times [4], [15].

However, the existing techniques for reachability analysis of TPNs is not well suited for dealing with the end-to-end timing issues that are critical to real-time systems. Reachability analysis of TPNs is currently based on the concept of state classes [2], [3], a mechanism that groups time-dependent system states into equivalence sets in terms of the same discrete states characterized by TPN markings, so as to reduce the state explosion problem. A state class is composed of a marking and a firing domain, which is defined as the union of the firing domains of all states in the state class. The firing domain in a state class is relative to the moments at which transitions are enabled. However, the value of the time at which the transitions are enabled is unknown. The problem with this relative-time firing domain is that there is no clear way to calculate the end-to-end time needed for or required of task execution in the system being modeled. In particular, linear addition of the transitions' firing intervals between two adjacent state classes in the reachability tree of a TPN cannot result in the time span between the two state classes. Consequently, reachability tree based on such a state class concept does not give sufficient information required for the timeliness analysis or timing verification of modeled systems. (See Section II for more details.) In order to derive the time span information, one has to repeat the reachability analysis in

Manuscript received August 31, 1999; revised June 19, 2000. This work was supported in part by the Army Research Office under Grant DAAG55-98-1-0428, and by the National Science Foundation under Grant HDR-9707076. This paper was recommended by Associate Editors M. A. Jafari and M. Zhou.

J. Wang is with the School of Computer Science, Florida International University, Miami, FL 33199 USA.

Y. Deng was with the School of Computer Science, Florida International University, Miami, FL 33199 USA. He is now with the Department of Computer Science, University of Texas at Dallas, Richardson, TX 75083 USA (e-mail: deng@cs.fiu.edu).

G. Xu is with AT&T Labs, Lincroft, NJ 07738 USA.

Publisher Item Identifier S 1083-4419(00)08300-X.

a different way based on the obtained reachability tree [3]. In ensuing discussion, we refer to a state class used in [2] as a *traditional* state class.

In this paper, we present a new reachability analysis technique for TPN models that effectively addresses the problem described above. Our technique is based on a concept called *clock-stamped state class* (CS-class), which not only groups system states into compact representation of state classes, but also records the time, relative to the beginning of system execution, when such states are reached. In particular, a CS-class consists of three parts: 1) a marking that represents a logical state of the modeled system; 2) a "global" firing domain corresponding to firing intervals, whose values are counted relative to the beginning of the net's execution, of all *firable* transitions in the state class; and (3) a clock stamp that corresponds to the moment when the state class is reached with the clock value relative to the beginning of the execution. We present an algorithm to construct the reachability tree of the TPN based on the CS-class concept. With the reachability tree generated based on CS-classes, we can straightforwardly compute the time span between any two reachable CS-classes, thus the end-to-end time delay in task execution. Moreover, a CS-class can be uniquely mapped into a traditional state class, but not vice versa. Therefore, the CS-class based analysis method is more general than the traditional state class-based analysis method given in [2].

The rest of the paper is organized as follows. In Section II, we present our CS-class based reachability analysis technique for TPN models. In Section III, we illustrate the application of this technique to timing property verification of the TPN model of a command and control (C2) system. Section IV concludes the paper.

II. Timeliness Analysis of TPN Models

In this section, we first give a brief introduction to TPNs and the traditional state class concept. After pointing out the problem associated with this concept, we proceed to present the formal definition of CS-class for TPNs and define transition firing rules, which guide the construction of the reachability trees of TPNs based on CS-classes. We then show how the reachability trees can be used for timing property analysis of TPN. In addition, the following interval arithmetic will be used later: Let $I_1 = [a_1, b_1]$ and $I_2 = [a_2, b_2]$, with $0 \leq a_i \leq b_i \leq +\infty$. Then we define $I_1 + I_2$ to be the interval $[a_1 + a_2, b_1 + b_2]$ and $I_1 - I_2$ to be $[a_1 - a_2, b_1 - b_2]$.

A. Time Petri Nets

A TPN is a tuple (P, T, B, F, M_0, SI) where

1) $P = \{p_1, p_2, \ldots, p_m\}$ is a finite nonempty set of *places*.
2) $T = \{t_1, t_2, \ldots, t_n\}$ is a finite nonempty set of *transitions*.
3) $B : P \times T \to N$ is the *backward incidence function*.
4) $F : T \times P \to N$ is the *forward incidence function*.
5) M_0 is the *initial marking*. (P, T, B, F and M_0 together define a Petri net.)
6) *SI* is a mapping called *static interval*. $\forall t \in T$, $SI(t) = [SEFT(t), SLFT(t)]$, where $SEFT(t)$ is the *static earliest firing time* and $SLFT(t)$ the *static latest firing time*.

A *state* of a TPN is a pair $S = (M, I)$ where

1) M is a marking.
2) I is a firing interval set which is a vector of possible firing times. The number of entries in this vector is given by the number of the transitions enabled by marking M.

A state is reached from the initial state by a given sequence of firing times corresponding to a firing sequence. Since any reachable marking may be reached from the initial marking by different sequence of firing times corresponding to the same firing sequence, the state space may be infinite.

A *state class* [2] of a TPN is an aggregated pseudostate associated with a firing sequence, which represents *all* states reachable from the initial state by firing all feasible firing values corresponding to the same firing sequence. A state class is a pair $C = (M, D)$ in which:

1) M is the marking of the class: all states in the class have the same marking;
2) D is the firing domain of the class, which is defined as the union of the firing domain of all the states in the class.

A transition t_i is *firable* from class $C = (M, D)$ if t_i is *enabled* by marking M, and may fire before the minimum of all *LFT*'s related to all enabled transitions. For detailed firing rules see [2].

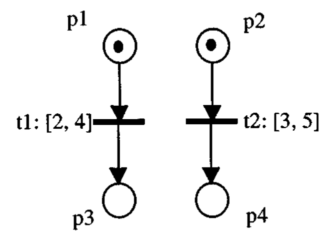
The concept of state class helps generate reachability tree [2]. In a state class, the firing domain is relative to the epoch when the marking of the class is reached, whose value is not known in the reachability tree. Because of its relative domain, such a state class concept, however, cannot directly provide any information about time span between any two classes C_i and C_j where C_j can be reached from C_i. This is because linear addition of the firing intervals of the transitions that lead the net from state class C_i to state class C_j cannot result in the time span between the two state classes. Let us take the simple TPN shown in Fig. 1(a) as an example. As we see, transitions $t1$ and $t2$ are concurrent, and starting from the initial state the net may take any time in interval $[3, 5]$ to reach the marking M_3: $(0\,0\,1\,1)$. Fig. 1(b) shows the reachability tree based on classic state classes. There are two schedules leading the net to marking M_3 that are characterized by firing sequences $t1t2$ and $t2t1$. Under the first firing schedule, $D_0(t1)$ plus $D_1(t2)$ gives $[2, 7]$, and under the second schedule, $D_0(t2)$ plus $D_2(t1)$ gives $[3, 6]$: none of the two results gives the correct time interval that leads the net from the initial marking M_0 to M_3.

In the next section, we present the concept of CS-class that leads us to solve the problem.

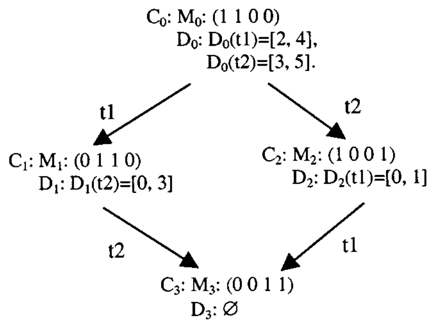
B. Clock Stamped State Classes

A *clock stamped state class* (CS-class) is a 3-tuple $C = (M, D, ST)$ where

1) M is a marking.
2) D is a *firing domain*, i.e., a set of constraints on the values of the time to fire for transitions enabled by current marking M. For an enabled transition t_i, $D(t_i)$ represents its firing interval. Let $EFT(t_i)$ be the left bound of $D(t_i)$ (the earliest firing time) and $LFT(t_i)$ be the right bound of $D(t_i)$ (the latest firing time).
3) *ST* is the time stamp of the CS-class, which is a (global) time interval.



(a)



(b)

Fig. 1. (a) Simple TPN. (b) Reachability tree based on classic state classes.

As will be pointed out in Theorem 1, the transition firing rules defined later will put meanings on the firing domain D and the time stamp *ST* as follows: 1) For an enabled transition $t_i, D(t_i)$ gives the global firing time interval of t_i, where by "global" we mean the values are counted relative to the beginning of the net's execution from the initial CS-class C_0—the initial CS-class is defined as $C_0 = (M_0, D_0, ST_0)$ where M_0 is the initial marking, D_0 contains all the static firing time intervals of the transitions enabled at M_0, and $ST_0 = [0,0]$; 2) *ST* gives the global time delay interval in which the net runs from the initial CS-class C_0 to current CS-class C.

Now we consider the firing rules that guide the generation of all reachable CS-classes of a TPN. An enabled transition t_j is said to be *firable* at CS-class C_k if $EFT_k(t_j) \le \min\{LFT_k(t_i), t_i \in E(C_k)\}$, where $E(C_k)$ is the enable set at C_k. Let $Fr(C_k)$ be the set of firable transitions at CS-class C_k, and let

$$MLFT(C_k) = \min\{LFT_k(t_i), t_i \in Fr(C_k)\} \tag{1}$$

where $MLFT(C_k)$ defines the minimum of latest firing times of all firable transitions in $Fr(C_k)$. We divide the firable transitions in $Fr(C_k)$ into two groups: 1) *inherited firable transitions* that were firable before C_k is reached and 2) *new firable transitions* that begin firable at C_k. The firing of transition $t_f \in Fr(C_k)$ changes the CS-class to C_{k+1}. Let CS-class $C_k = (M_k, D_k, ST_k)$ and $C_{k+1} = (M_{k+1}, D_{k+1}, ST_{k+1})$.

1) Transition Firing Rules:

Step 1) Calculate $\tilde{D}_k(t_f)$, the feasible firing intervals of the firing transition t_f. It is achieved by shifting right bound of $D(t_f)$ to $MLFT(C_k)$ while keeping its left bound unchanged, i.e.,

$$\tilde{D}_k(t_f) = [EFT_k(t_f), MLFT(C_k)]. \tag{2}$$

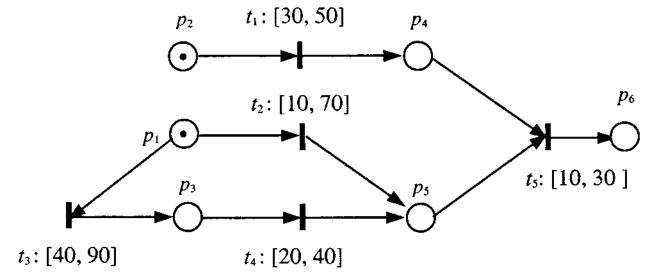


Fig. 2. Simple TPN model with concurrency, competence, and synchronization.

Let

$$ST_{k+1} = \tilde{D}_k(t_f). \tag{3}$$

Step 2) Calculate firing intervals of inherited firable transitions in CS-Class C_{k+1}.

2.1 Let $M'_{k+1} = M'_k - B(t_f)$ and collect (inherited) firable transitions at M'_{k+1}.

2.2 Let $D_{k+1} = D_k$ and delete from D_{k+1} all entries whose corresponding transitions are disabled by M'_{k+1}.

2.3 For each inherited firable transition t_j $(t_j \neq t_f)$ at M'_{k+1}, let

$$EFT_{k+1}(t_j) = \max(EFT_k(t_j), EFT_k(t_f)). \tag{4}$$

Step 3) Calculate the firing intervals of new firable transitions after firing t_f.

3.1) Let $M_{k+1} = M'_{k+1} + F(t_f)$ and collect new firable transitions: they are firable at M_{k+1} but not firable at virtual marking M'_{k+1}.

3.2) Add into D_{k+1} entries that corresponding new firable transitions at M_{k+1}: if t_j $(t_j \neq t_f)$ is a new firable transition at M_{k+1}, then

$$D_{k+1}(t_j) = SI(t_j) + ST_{k+1}. \tag{5}$$

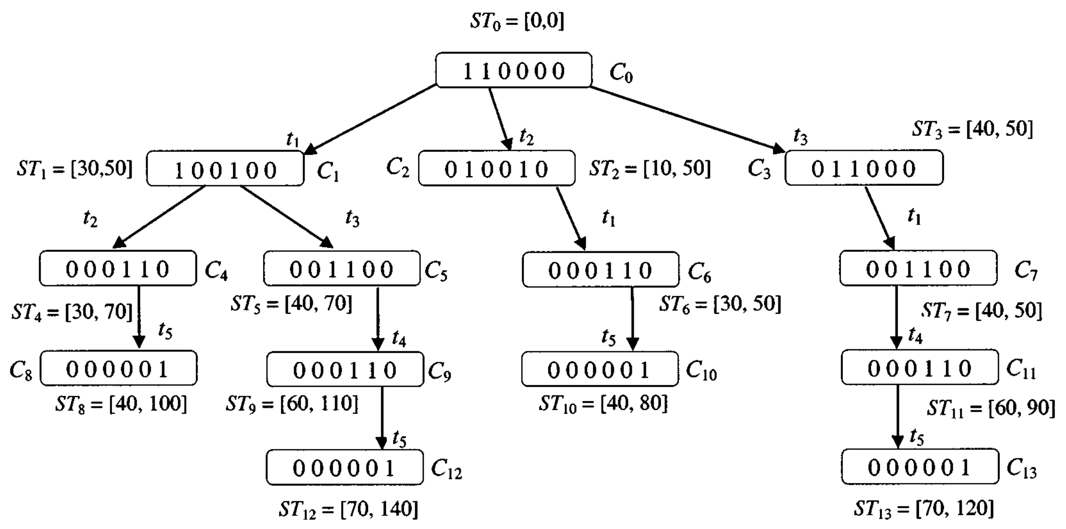
3.3) If t_f is still firable at M_{k+1} after its own firing, let

$$D_{k+1}(t_f) = SI(t_f) + ST_{k+1}. \tag{6}$$

Note that by Step 3.3), a transition that is still enabled after its own firing, is always treated as a newly enabled one. This simplifies the treatment of states in which a transition has sufficient tokens in its input places to permit multiple firings. The treatment of this condition, usually referred to as *multiple enabledness* [2], requires that multiple firing intervals be associated with a single transition and involves a number of semantic subtleties that are not relevant to the objective of this paper.

Example 1: Consider a simple TPN model, shown in Fig. 2. The initial CS-class is $C_0 = (M_0, D_0, ST_0)$ where

$$\begin{aligned} ST_0 &= [0,0].\\ M_0 &= (1\,1\,0\,0\,0\,0)^T,\\ D_0 &= \{D_0(t_1)\colon [30,50], D_0(t_2)\colon [10,70],\\ &\quad D_0(t_3)\colon [40,90]\}. \end{aligned}$$



ST_0 = [0,0]
1 1 0 0 0 0 C_0
t_1
t_2
t_3
ST_1 = [30,50] 1 0 0 1 0 0 C_1
C_2 0 1 0 0 1 0 ST_2 = [10, 50]
C_3 0 1 1 0 0 0 ST_3 = [40, 50]
0 0 0 1 1 0 C_4 ST_4 = [30, 70]
0 0 1 1 0 0 C_5 ST_5 = [40, 70]
0 0 0 1 1 0 C_6 ST_6 = [30, 50]
0 0 1 1 0 0 C_7 ST_7 = [40, 50]
t_4
t_5
C_8 0 0 0 0 0 1 ST_8 = [40, 100]
0 0 0 1 1 0 C_9 ST_9 = [60, 110]
0 0 0 0 0 1 C_{10} ST_{10} = [40, 80]
0 0 0 1 1 0 C_{11} ST_{11} = [60, 90]
0 0 0 0 0 1 C_{12} ST_{12} = [70, 140]
0 0 0 0 0 1 C_{13} ST_{13} = [70, 120]

Fig. 3. Reachability tree of the TPN in Fig. 1.

Moreover, it follows from D_0 that

$$MLFT(C_0) = \min\{EFT_0(t_1), EFT_0(t_2), EFT_0(t_3)\} = \{50, 70, 90\} = 50.$$

Firing t_1 will result in CS-class $C_1 = (M_1, D_1, ST_1)$. Then it follows from Step 1) that

$$ST_1 = [EFT_0(t_1), MLFT(C_0)] = [30, 50].$$

And it follows from Step 2) that

$$\begin{aligned} M_1' &= M_0 - B(t_1) = (1\,0\,0\,0\,0\,0)^T. \\ D_1(t_2) &= [\max(EFT_0(t_2), EFT_0(t_1)), LFT_0(t_2)] \\ &= [30, 70]. \\ D_1(t_3) &= [\max(EFT_0(t_3), EFT_0(t_1)), LFT_0(t_3)] \\ &= [40, 90]. \end{aligned}$$

Finally, it follows from Step 3) that

$$M_1 = M_1 + F(t_1) = (1\,0\,0\,1\,0\,0)^T.$$

In M_1, there is no new enabled transition. Thus, we get $C_1 = (M_1, D_1, ST_1)$ as

$$\begin{aligned} ST_1 &= [30, 50], \\ M_1 &= (1\,0\,0\,1\,0\,0)^T, \\ D_1 &= \{D_1(t_2)\colon [30, 70], D_1(t_3)\colon [40, 90]\}. \end{aligned}$$

Similarly at C_0, firing t_2 will result in CS-class $C_2 = (M_2, D_2, ST_2)$, where

$$\begin{aligned} ST_2 &= [EFT_0(t_2), MLFT(C_0)] = [10, 50], \\ M_2 &= M_2 - B(t_2) + F(t_2) = (0\,1\,0\,0\,1\,0)^T, \\ D_2 &= \{D_2(t_1)\colon [30, 50]\}. \end{aligned}$$

At C_0, firing t_3 will result in CS-class $C_3 = (M_3, D_3, ST_3)$ where

$$\begin{aligned} ST_3 &= [40, 50], \\ M_3 &= (0\,1\,1\,0\,0\,0)^T, \\ D_3 &= \{D_3(t_1) = [40, 50]\}. \end{aligned}$$

Notice that t_4 is enabled but not firable at C_3 because

$$EFT_3(t_4) = 60 > MLFT(C_3) = 50.$$

At C_1, firing t_2 results in CS-class C_4 (M_4, D_4, ST_4) where

$$\begin{aligned} ST_4 &= [30, 70], \\ M_4 &= (0\,0\,0\,1\,1\,0)^T, \\ D_4 &= \{D_4(t_5)\colon SI(t_5) + ST_4 = [10, 30] + [30, 70] \\ &= [40, 100]\}. \end{aligned}$$

Following this way, we can generate the reachability tree of the example TPN, which is depicted in Fig. 3.

C. Timeliness Analysis of TPN Models

We have described the transition firing rules that guide the evolution of CS-classes in Section II-B. Below, we are to show the benefits that the new concept of state classes brings us. First, Theorem 1 shows that what the firing domain D and the time stamp *ST* in a CS-class exactly stand for. Then, Corollary 1 shows what we can gain from the generation of the reachability tree, and Theorem 2 shows the relationship between CS-classes and traditional state classes. To facilitate our description, we denote the firing schedule that leads the TPN from initial state class C_0 to state class C_n by firing $t_0\, t_1 \ldots t_{n-1}$ by

$$(C_0, t_0)(C_1, t_1) \ldots (C_i, t_i) \ldots (C_{n-1}, t_{n-1})C_n.$$

Theorem 1: Let $C_i = (M_i, D_i, ST_i)$ be a reachable state class from $C_0 = (M_0, D_0, ST_0)$. Then

(C1) ST_i is the global time (interval) when CS-class C_i is visited

(C2) If $t_j \in Fr(C_i)$, then $D_i(t_j)$ is the global firing time interval of t_j.

Proof: From the preconditions, we know there must be a firing schedule starting with C_0 and ending with C_i, i.e.,

$$(C_0, t_0)(C_1, t_1)\dots(C_{i-1}, t_{i-1})C_i.$$

The proof of the theorem is carried out by induction on i. For the basis case $(i = 0)$, C_0 is the initial class. Obviously we have: 1) $\forall t_j \in Fr(D_0), D_0(t_j)$ is exactly the static firing time interval of t_j, which is also the global firing time, and 2) $ST_0 = [0, 0]$, which is the arriving time of C_0. Therefore, the assertion of the theorem holds for $i = 0$.

Now assume that the assertion holds for $i \leq k$. Consider $i = k + 1$. It follows from (2) and (3) that

$$ST_{k+1} = [EFT_k(t_k), MLFT(C_k)]. \tag{7}$$

In (7), $EFT_k(t_k)$ is the earliest global firing time of t_k, and $MLFT(C_k)$, according to (1), is the minimum latest firing time of all firable transitions in $Fr(C_k)$, hence the actual latest global firing time of t_k. Therefore, ST_{k+1} is the global firing time interval of t_k. Because firing a transition takes no time, ST_{k+1} is also the global arriving time of state class C_{k+1}, which is reached by firing t_{k+1}. So (C1) is proven.

Suppose that a transition t_j is firable at C_{k+1}. Now we prove (C2) according to three different cases of t_j.

Case 1. t_j is a newly enabled transition at M_{k+1} and $t_j \neq t_k$.

It follows from (1) and (7) that

$$\begin{aligned} &D_{k+1}(t_j) \\ &\quad = SI(t_j) + ST_{k+1} \\ &\quad = [SEFT(t_j) + EFT_k(t_k), SLFT(t_j) + MLFT(C_k)] \end{aligned}$$

where, $EFT_k(t_k)$ is the earliest (global) arriving time of state class C_{k+1}, $SEFT(t_j)$ the static (relative) earliest firing time when t_j is enabled at C_{k+1}, so $LEFT(t_j) + EFT_k(t_k)$ is the earliest global firing time of transition t_j; $MLFT(C_k)$ is the latest (global) arriving time of state class $C_{k+1}, SLFT(t_j)$ the static (relative) latest firing time when t_j is enabled at C_{k+1}, so $SLFT(t_j) + MLFT(C_k)$ is the latest global firing time of transition t_j. Therefore, $D_{k+1}(t_j)$ is the global firing time interval of t_j.

Case 2. $t_j = t_k$.

Because we ignore multiple-enabledness, so t_k is viewed as a new enabled transition at M_{k+1}. Thus the conclusion drawn in Case 1 also applies to this case.

Case 3. t_j is an inherited transition.

In this case, it follows from *Step* 2 that

$$D_{k+1}(t_j) = [\max(EFT_k(t_j), EFT_k(t_k)), LFT_k(t_j)].$$

According to the assumption made for $i \leq k$, $[EFT_k(t_k)), LFT_k(t_j)]$ is the global firing time interval of transition t_j at state class C_k. The latest global firing time of t_j at C_{k+1} should be the same as it is at C_k; however, the earliest global firing time of t_j at C_{k+1} must take the larger value of $EFT_k(t_j)$ and $EFT_k(t_k)$, because t_j is supposed to fire after C_{k+1} is reached. So, $D_{k+1}(t_j)$ is the global firing time interval of transition t_j at state class C_{k+1}.

Hence, the theorem holds. □

It is worth pointing out that the proof also shows the soundness and completeness of the global time interval ST_i. By soundness we mean that if sequence $t_0\ t_1 \dots t_{i-1}$ fires, then the current time must be in ST_i; by completeness we mean for any time τ in ST_i, it is possible to fire sequence $t_0\ t_1 \dots t_{i-1}$, and end at time τ and class C_i. So ST_i gives the exact global time interval when CS-class C_i is visited.

Corollary 1: Let $C_i = (M_i, D_i, ST_i)$ and $C_j = (M_j, D_j, ST_j)$ be two reachable CS-classes of a TPN where C_j is reachable from C_i. If $\forall t_j \in Fr(D_i), t_j$ is a newly enabled transition, then the time span that the TPN runs from C_i to C_j is $ST_j - ST_i$.

Proof: Because all firable transitions at C_i are newly enabled, the future behavior of the TPN starting from C_i is reached is independent of the history before C_i is reached. Suppose that if the TPN starts running from class C_i at time 0, it will reach class C_j at global time interval $[x, y]$. Then we know that if the TPN starts running from class C_i at time a, it will reach class C_j at global time interval $[x + a, y + a]$. Futhermore, if the TPN starts running from class C_i at time interval $ST_i = [a, b]$, it will reach class C_j at global time interval $ST_i = [\min\{x+a, x+b\}, \max\{y+a, y+b\}] = [x+a, y+b]$. Because the time span that the TPN runs from C_i to C_j is independent of the starting time, it follows from Theorem 1 that the time span is $[x, y]$, or $ST_j - ST_i$. □

The conclusion of Corollary 1 can be directly used for timeliness analysis. As mentioned in [15], the key issue of timeliness analysis is to verify whether a marking can be reached with timing constraints. Corollary 1 shows that the concept of CS-classes helps establish quantitative timing relationship between any two reachable classes when creating the reachability tree.

Example 2: Let us go back to Fig. 2. Since $ST_0 = [0, 0]$, the time span from C_0 to C_8 is ST_8, from C_0 to C_{10} is ST_{10}, from C_0 to C_{12} is ST_{12}, and from C_0 to C_{13} is ST_{13}. Suppose that we have a timing constraint that asks the absorbing marking $(0\,0\,0\,0\,1)^T$, in which classes C_8, C_{10}, C_{12}, and C_{13} stay, must be reached within 150 time units from the initial marking, then we know that the constraint is satisfied. □

Example 3: Applying the CS-class reachability analysis method to the TPN shown in Fig. 1, we get the reachability tree as shown in Fig. 4. From the reachability tree we find that the marking $(0\,0\,1\,1)$ can be reached from $(1\,1\,0\,0)$ via two firing sequences: $(t1, t2)$ (left branch) and $(t2, t1)$ (right branch). The time span via the first sequence is $[3, 4]$ and the second sequence $[3, 5]$. Therefore, marking $(0\,0\,1\,1)$ can be reached within $[3, 5]$. Obviously, it gives the correct end-to-end time delay. □

The next theorem shows that a CS-class can be uniquely mapped into a traditional state class. It is an interesting property, because it reveals two facts: 1) CS-class-based analysis method is as effective as the traditional method given in [2] in constructing reachability tree, because no extra node is introduced in the tree; and 2) CS-class-based analysis method has all advantages that traditional state class based analysis method has gained.

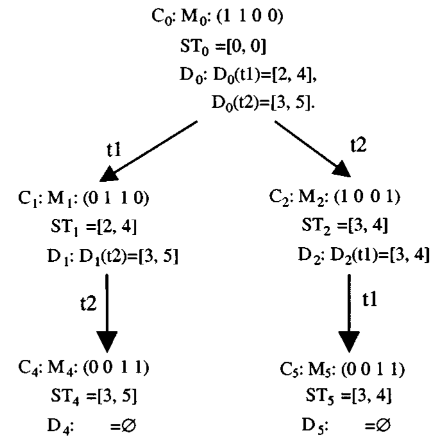


Fig. 4. Reachability tree of the TPN in Fig. 1 based on CS-state classes.

Theorem 2: Let $\mathbf{C}$ and $\mathbf{C}'$ be the CS-class set and the traditional state class set of a given TPN model, respectively. Let $ST_k = [EST_k, LST_k]$ for $C_k = (M_k, D_k, ST_k)$ and suppose that the firing of transition t_f leads the TPN from C_k to C_{k+1}. Then, for any $C_k \in \mathbf{C}$, there is a corresponding $C'_k \in \mathbf{C}'$ such that [see (8) at the bottom of the page].

Proof: We prove this theorem by checking CS-classes in any given firable schedule

$$(C_0, t_0)(C_1, t_1)\ldots(C_i, t_i)\ldots$$

by induction on i. For the basis case $(i = 0)$, C_0 is the initial CS-class $C_0 = (M_0, D_0, ST_0)$, where M_0 is the initial marking, D_0 contains all the static firing time intervals of the transitions enabled at M_0, and $ST_0 = [0, 0]$. Correspondingly, we have $C'_0 = (M'_0, D'_0)$, where M'_0 is the initial marking, D'_0 contains all the static firing time intervals of the transitions enabled at M'_0. Obviously, we have $M'_0 = M_0$, and $D'_0 = D_0$. Because $ST_0 = [0, 0]$, $D'_0 = D_0 - ST_0$. It satisfies (8).

Now, assume that the assertion holds for $i \le k$. Consider $i = k + 1$. Because $M'_k = M_k$ and same rules are used to infer new markings in both traditional state class generation method and CS-class generation method, so we have $M'_{k+1} = M_{k+1}$. Thus, for any transition $t_j \in Fr(C_{k+1})$ holds $t_j \in Fr(C'_{k+1})$. We are going to prove the theorem in three cases.

Case 1: t_j is a new firable transition at C_{k+1}. It follows from (5) or (6) that $D_{k+1}(t_j) = SI(t_j) + ST_{k+1}$; on the other hand, according to the firing rules of traditional state classes, we have $D'_{k+1}(t_j) = SI(t_j)$. Therefore, $D'_{k+1}(t_j) = D_{k+1}(t_j) - SI(t_j)$.

Case 2: t_j is an inherited transition at C_{k+1}, and t_f is a new firable transition at C_k. In this case, the relative firing time interval of transition t_f is $[EST_{k+1} - EST_k, LST_{k+1} - LST_k]$. So, the relative earliest firing time of transition t_j at C_{k+1} is $\max\{EFT'_k(t_j) - (LST_{k+1} - LST_k), 0\}$, and the relative latest firing time of transition t_j at C_{k+1} is $LFT'_k(t_j) - (EST_{k+1} - EST_k)$. It gives

$$D'_{k+1}(t_j) = [\max\{EFT'_k(t_j) - (LST_{k+1} - LST_k), 0\},\\ LFT'_k(t_j) - (EST_{k+1} - EST_k)].$$

Case 3: t_j is an inherited transition at C_{k+1}, and t_f is an inherited transition at C_k. In this case, no matter when C_k is reached, transition t_f may fire at any time in the global interval $[\max\{EFT_k(t_f), EST_k\}, MLFT(C_k)]$, and leads the TPN to C_{k+1} at $[EST_{k+1}, LST_{k+1}]$. So, the relative firing time interval of transition t_f is $[\max\{EST_{k+1} - EST_{k+1}, 0\}, LST_{k+1} - EST_k]$. Then, the relative earliest firing time of transition t_j at C_{k+1} is $\max\{EFT'_k(t_j) - (LST_{k+1} - EST_k), 0\}$, and the relative latest firing time of transition t_j at C_{k+1} is $LFT'_k(t_j) - \max\{(EST_{k+1} - EST_k), 0\}$. It gives

$$D'_{k+1}(t_j) = [\max\{EFT'_k(t_j) - (LST_{k+1} - EST_k), 0\},\\ LFT'_k(t_j) - \max\{(EST_{k+1} - EST_k), 0\}].$$

□

Example 4: Consider the TPN shown in Fig. 2 again. According to Theorem 2 and based on the model's CS-class-based reachability tree shown in Fig. 3, we can easily compute the traditional state class-based reachability tree, which is shown in Fig. 5. This traditional state class-based reachability tree can also be computed by the use of state class enumeration method given in [2]. □

To facilitate the timeliness analysis of TPNs, we implemented a software tool named TPNm&a. The tool is coded in C++ and developed with UIM/X (professional edition), and presently runs on Solaris.

III. Timing Property Verification of a C2 System

In this section, we show the application of CS-class-based reachability analysis to the verification of timing properties of a

$$M'_k = M_k.$$

$$D'_{k+1}(t_j) = \begin{cases} D_{k+1}(t_j) - ST_{k+1}, \\ \quad t_j \text{ is a new firable transition at } C_{k+1}; \\ [\max\{EFT'_k(t_f) - (LST_{k+1} - LST_k), 0\}, LFT'_k(t_f) - (EST_{k+1} - EST_k)], \\ \quad t_j \text{ is an inherited transition at } C_{k+1}, \text{ and } t_f \text{ is a new firable transition at } C_k; \\ [\max\{EFT'_k(t_f) - (LST_{k+1} - EST_k), 0\}, LFT'_k(t_f) - \max\{(EST_{k+1} - LST_k), 0\}], \\ \quad t_j \text{ is an inherited transition at } C_{k+1}, \text{ and } t_f \text{ is an inherited transition at } C_k \end{cases} \tag{8}$$

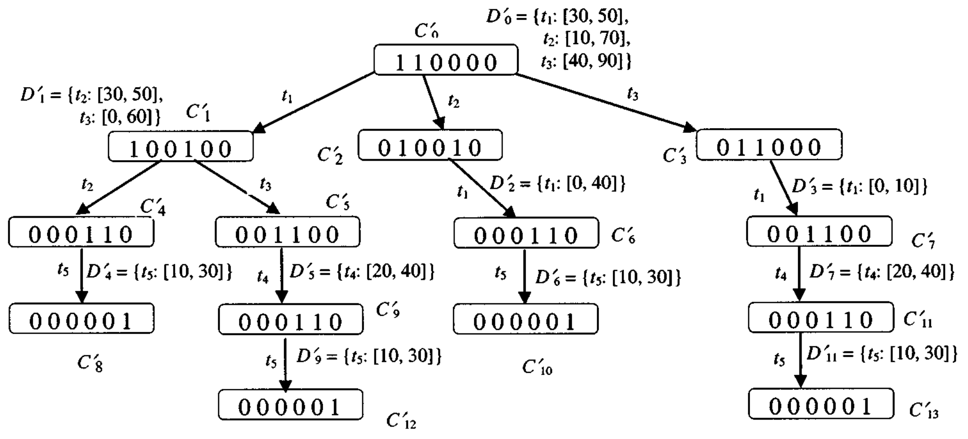


Fig. 5. Traditional reachability tree of the TPN in Fig. 2.

command and control (C2) system.[1] A C2 system is a distributed modularized system. It achieves mission success by executing a set of generally accepted C2 functions in an asynchronous manner. These functions include [1] *Threat Detection, Threat Discrimination, Identification and Tracking, Threatening Assessment, Battle Planning, Weapon-to-Target Assignment, Engagement Control*, and *Damage Assessment*.

A simplified but typical structure of a tactic anti-air C2 system with two levels of command and control centers is shown in Fig. 6, which consists of one first-level command center (indicated as *C2 Center*) and two second-level command centers (indicated as *Sub-Centers*). A pair of (C2 Center, Sub-Center) may be (division, regiment) or (brigade, battalion). They are geographically dispersed due to environmental and survivability reasons, which contributes to the distributed architecture of C2 organization.

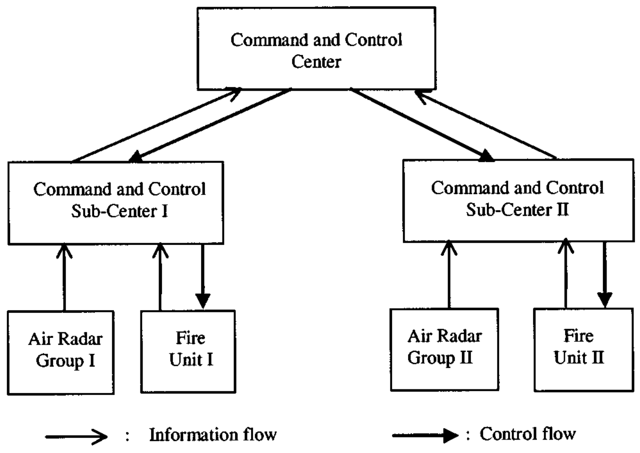

Command and Control Center
Command and Control Sub-Center I
Command and Control Sub-Center II
Air Radar Group I
Fire Unit I
Air Radar Group II
Fire Unit II
: Information flow
: Control flow

Fig. 6. Structure of a tactic anti-air C2 system.

A. Requirements of the C2 System

The operation of the system is described as follows.

1) Each radar group is composed of three air radars and a data processor. The specification is
   - — Each radar senses air targets every 30 time units (TU).
   - — The data from the three air radars are fused at the processor, which takes two to four TU.
   - — The fused data are coded (taking two to four TU) and sent to their corresponding sub-center (taking one to two TU).
2) The C2 Center is composed of three *seats*: two *intelligence seats* and one *decision-making seat*. The behavior specification of this component is as follows:
   - — The two intelligence seats communicate with the decision-making seat through a common memory.
   - — The messages from three sub-centers are first copied and dispatched to the two intelligence seats. The two actions together take one to two TU.
   - — The two intelligence seats then do situation assessment based on these messages to achieve a relatively more precise situation figure (taking three to five TU), and then make *threatening assessment* independently for each target and send the results to the decision-making seat. The two actions together take three to five TU.
   - — The decision-making seat works on a scheme of *battle planning* (taking five to six TU). The result is sent to the three sub-centers (taking one to two TU).
3) Each sub-center is composed of an intelligence seat and a decision-making seat. The behavior specification of this component is as follows.
   - — The intelligence seat receives message from its radar group and conducts *target discrimination, identification* and *tracking*, and further conducts *threatening assessment*, then sends the result to the C2 center. It takes two to three TU.
   - — After receiving the scheme of *battle planning* from C2 center, the sub-center fuses it with related data in its database (taking one to two TU) so as to form a detailed scheme of *weapon-to-target assignment* (taking five to seven TU). Then, the results are sent to fire units (taking one to two TU).
4) The specification of a Fire Unit is as follows.
   - — When the scheme of *weapon-to-target assignment* arrives from its sub-center, it conducts *engagement con-*

[1]Notice that the timing data in this case study is artificial and does not reflect actual data in a real system.

trol (taking four to six TU) and sends fire command to weapons (taking one to two TU).

— Then, it conducts *damage assessment* (taking five to seven TU), and feed backs the assessment results to its corresponding sub-center in time (taking one to two TU).

In this example, we focus on timing requirements including:

(R1) The system reaction time, i.e., the time delay from a message regarding enemy intelligence being received by any subcenter to a fire command against the enemy being issued by a corresponding fire unit, must be less than or equal to 45 time units.

(R2) Since a C2 system is a closed-loop system, a constraint reflecting the time limitation for the feedback of damage assessment results should be included. This is captured by the requirement that the time delay from a detailed firing assignment scheme made by a sub-center to the result of the damage assessment referred to the execution of this scheme received by the same sub-center must be less than or equal to 20 time units.

(R3) Since the bottleneck for information processing is often located in the C2 Center, the center is always asked to respond quickly. This is captured by the requirement that the whole processing time for a group of messages from the three sub-centers must be less than or equal to 22 time units.

(R4) A C2 system is a real-time information processing system. Obtaining targets' information timely and continuously is extremely important to win a war. This is captured by the requirement that each Radar Group outputs targets' information periodically at a period of 40 time units.

In order to simplify analysis, we also assume both sub-centers have identical topology and timing properties, and all the two fire units have identical topology and timing properties.

B. TPN Model of the C2 System

According to the structure and operation rules described above, we build the system's TPN model as shown in Fig. 7. Table I gives its description. The TPN model is composed of seven subnets, each of which corresponds to a component. Transitions $t_{13}, t_{31}, t_{16}, t_{61}, t_{23}, t_{34}, t_{56}, t_{67}$, and t_{67} serve as connectors among the components, and places RG1.MSG, RG2.MSG, C2C.R1, C2C.R2, C2C.S1, C2C.S2, SC1.SI, SC2.SI, SC1.RM, SC2.RM, SC1.RI, SC1.SM, SC2.SM, FU1.S, FU2.S, FU1.R, and FU2.R serve as communication ports of the components.

C. System Verification

The verification is driven by showing satisfaction of timing requirements, which is monitored during the construction of reachability tree, and terminated as soon as the goal is reached. This avoids the generation of a complete reachability tree and thus improves the efficiency.

In fact, we can use a submodel to analyze a timing requirement. The submodels used to analyze the four timing requirements listed in Section III-A are depicted in Figs. 8–11.

Let us take the analysis of requirement R3 as an example. The initial marking M_0 is shown at the bottom of the page. Define marking M_e such that (see the second equation at the bottom of the page). Applying reachability analysis described in Section II gives the following CS-classes:

$$\begin{aligned}
C_0 &= (M_0, D_0, ST_0)\\
M_0 &= \{\text{C2C.R1,C2C.R2}\},\\
D_0 &= \{D_0(t_{101}) = [1, 2]\},\\
ST_0 &= [0, 0];\\
C_1 &= (M_1, D_1, ST_1)\\
M_1 &= \{\text{p101, p102}\},\\
D_1 &= \{D_1(t_{102}) = [4, 7], D_1(t_{103}) = [4, 7]\},\\
ST_1 &= [1, 2];\\
C_2 &= (M_2, D_2, ST_2)\\
M_2 &= \{\text{p102, p103}\},\\
D_2 &= \{D_2(t_{103}) = [4, 7]\},\\
ST_2 &= [4, 7];\\
C_3 &= (M_3, D_3, ST_3)\\
M_3 &= \{\text{p101, p104}\},\\
D_3 &= \{D_3(t_{102}) = [4, 7]\},\\
ST_3 &= [4, 7];\\
C_4 &= (M_4, D_4, ST_4)\\
M_4 &= \{\text{p103, p104}\},\\
D_4 &= \{D_2(t_{104}) = [9, 13]\},\\
ST_4 &= [4, 7];\\
C_e &= (M_e, D_e, ST_e)
\end{aligned}$$

	C2C.R1	C2C.R2	p101	p102	p103	p104	C2C.S1	C2C.S2
M_0:	1	1	0	0	0	0	0	0

	C2C.R1	C2C.R2	p101	p102	p103	p104	C2C.S1	C2C.S2
M_e:	0	0	0	0	0	0	1	1

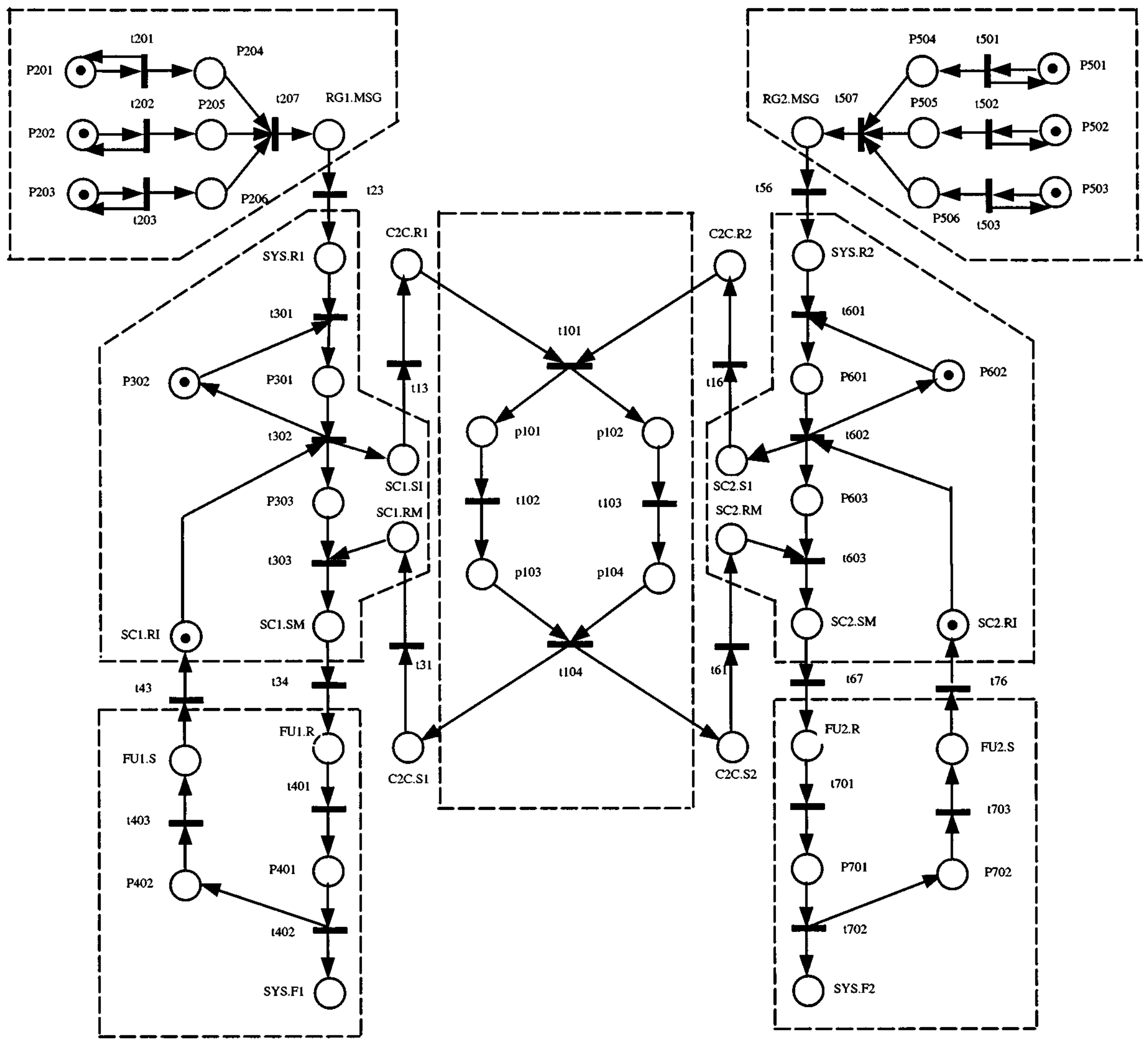


Fig. 7. TPN model of the C2 system.

$$M_e = \{\text{C2C.S1, C2C.S2}\},$$
$$D_e = \emptyset,$$
$$ST_e = [9, 13].$$

As we see, the time delay interval that the model takes to move from marking M_0 to M_e is $ST_e - ST_0 = [9, 13]$. It implies that the timing requirement R3 is proven satisfied.

Using the same technique, we can prove requirements R1 and R2 satisfied based on Figs. 8 and 9, respectively.

Now, we consider requirement R4. Reachability analysis shows that the TPN model dipicted in Fig. 11 has infinite markings (see equation at the bottom of the page). However, the reachability set has the following properties: 1) places p201–p207 are safe; 2) when each of transitions t_{201}, t_{202} and t_{203} fires one time, a token is deposited in place RG1.MSG; and 3) if each of transitions t_{201}, t_{202} and t_{203} fires n $(n \geq 1)$ times, n or $n-1$ tokens must have been deposited in place RG1.MSG. Based on these properties, it is easy to see that the net exhibits the same behavior from marking $(1\ 1\ 1\ 0\ 0\ 0\ 0\ i)$ to marking $(1\ 1\ 1\ 0\ 0\ 0\ 0\ i+1)$ as from marking $(1\ 1\ 1\ 0\ 0\ 0\ 0\ j)$ to marking $(1\ 1\ 1\ 0\ 0\ 0\ 0\ j+1)$, $i \geq 1$, $j \geq 1$, $i \neq j$. We can easily derive that the time for marking $(1\ 1\ 1\ 0\ 0\ 0\ 1)$ to be

	p201	p202	p203	p204	p205	p206	p207	RG1.MSG	
M_{4i}:	1	1	1	0	0	0	0	i	
M_{4i+1}:	1	1	1	1	1	1	0	i	$i = 0, 1, 2, \ldots$
M_{4i+2}:	1	1	1	0	0	0	1	i	
M_{4i+3}:	1	1	1	0	0	0	0	$i+1$	

TABLE I
LEGEND FOR FIG. 7

PLACE	DESCRIPTION	
p101, p102	Ready for situation assessment	
p103, p104	Ready for fusion and combat planning	
p201, p202, p203	Radar ready to sense air targets	
p204, p205, p206	Radar data for fusion	
p301	Waiting for evaluation of threat	
p302	Intelligence seat available	
p303	Waiting for decision-making of fire assignment	
p401	Ready to send fire command	
p402	Waiting for damage assessment	
RG1.MSG	Radar Group1 ready to send target message to Sub-Center I	
SYS.R1	A message from Air Radar Group I arrived	
SYS.F1	A combat command to Fire Unit I sent	
SC1.SI	Sub-Center I ready to send intelligence to Command and Control Center	
SC1.RM	Sub-Center I received command from Command and Control Center	
SC1.SM	Sub-Center I ready to send command to Fire Unit I	
SC1.RI	Sub-Center I received result of damage assessment from Fire Unit I	
FU1.R	Fire Unit I received command from Sub-Center I	
FU1.S	Fire Unit I ready to send result of damage assessment to Sub-Center I	
C2C.R1	Command and Control Center received message from Sub-Center I	
C2C.S1	Command and Control Center ready to send command to Sub-Center I	
TRANSITION	DESCRIPTION	TIMING INTERVAL
t101	Dispatch intelligence message to two staff seats	[1, 2]
t102, t103	Two staff seats conduct situation assessment	[3, 5]
t104	Top commander seat conducts information fusion and combat planning	[5, 6]
t201, t202, t203	Radar senses	[30, 30]
t204	Processor fuses data	[2, 4]
t205	Processor codes fused data	[1, 2]
t301	Sub-Center I conducts target discrimination, identification and tracking	[2, 3]
t302	Sub-Center I conducts threatening assessment	[1, 2]
t303	Sub-Center I conducts fire assignment	[4, 6]
t401	Fire Unit I conducts engagement control	[4, 6]
t402	Fire Unit I sends fire command	[1, 2]
t403	Fire Unit I conducts damage assessment	[5, 7]

reached from the initial marking $(1\ 1\ 1\ 0\ 0\ 0\ 0)$ is $[34, 36]$. So the requirement R4 is satisfied. Now, all requirements are verified.

IV. Conclusion

A new reachability-based analysis technique for TPN is presented. It is based on a concept called clock-stamped state class (CS-class). With the reachability tree generated based on CS-classes, we can straightforwardly compute the end-to-end time delay in task execution. Moreover, a CS-class can be uniquely mapped to a traditional state class based on which the conventional reachability tree is constructed. Therefore, the CS-class-based analysis technique is more general than the existing technique. A command and control system is used as an example to show the timing property verification procedure.

Currently, we are working on a compositional timing property verification technique, which helps to control the complexity of

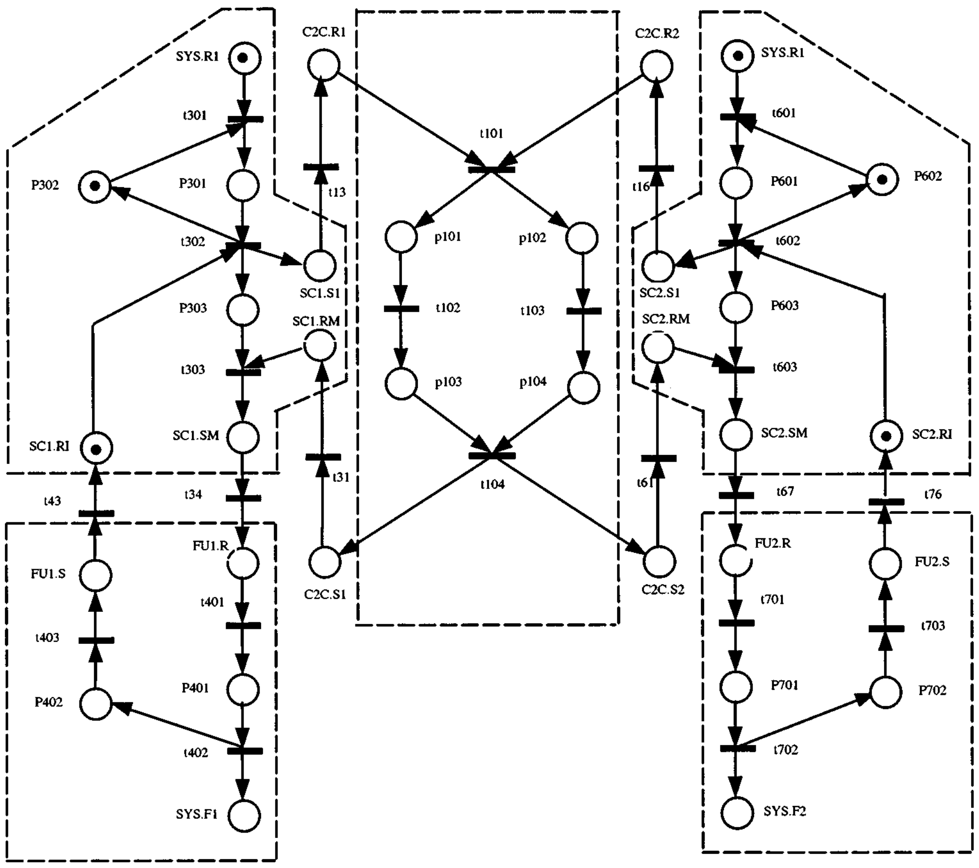


Fig. 8. Submodel for verifying requirement R1.

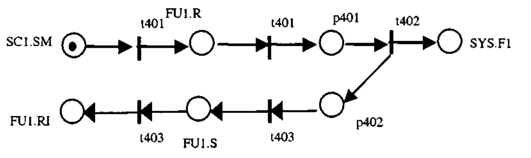


Fig. 9. Submodel for verifying requirement R2.

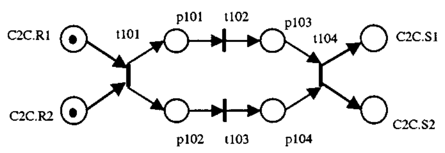


Fig. 10. Submodel for verifying requirement R3.

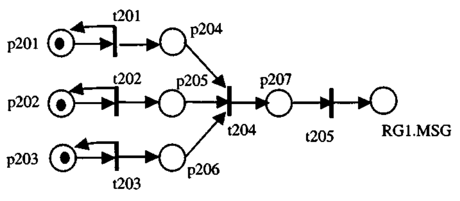


Fig. 11. Submodel for verifying requirement R4.

large-scale real-time system analysis. We have developed a set of component-level reduction rules for TPNs in [17].

REFERENCES

[1] M. Athans, “Command and control (C2) theory: A challenge to control science,” *IEEE Trans. Automat. Contr.*, vol. AC-32, no. 4, pp. 286–293, 1987.

[2] B. Berthomieu and M. Diaz, “Modeling and verification of time dependent systems using time Petri nets,” *IEEE Trans. Softw. Eng.*, vol. 17, no. 3, pp. 259–273, 1991.

[3] G. Bucci and E. Vivario, “Compositional validation of time-critical systems using communicating time Petri nets,” *IEEE Trans. Softw. Eng.*, vol. 21, no. 12, pp. 969–992, 1995.

[4] E. M. Clark, E. A. Emerson, and A. P. Sistla, “Automatic verification of finite concurrent systems using temporal logic specifications,” *ACM Trans. Program. Lang. Syst.*, vol. 8, no. 2, pp. 244–263, 1986.

[5] A. Danthine, “Protocol representation with finite state models,” *IEEE Trans. Commun.*, vol. 28, no. 4, 1984.

[6] Y. Deng, J. Wang, and R. Sinha, “Incremental architectural modeling and verification of real time concurrent systems,” in *Proc. 2nd IEEE Int. Conf. Formal Engineering Methods*, Brisbane, Australia, 1998, pp. 26–34.

[7] G. Florin, C. Fraize, and S. Natkin, “Stochastic Petri nets: Properties, applications and tools,” *Microelectron. Reliab.*, vol. 31, no. 4, pp. 669–697, 1991.

[8] C. Heitmeyer and D. Mandrioli, “Formal methods for real-time computing: An overview,” in *Formal Methods for Real-Time Computing*, C. Heitmeyer and D. Mandrioli, Eds. New York: Wiley, 1995, pp. 1–29.

[9] D. Haban and K. G. Shin, “Application of real-time monitoring to scheduling tasks with random execution times,” *IEEE Trans. Softw. Eng.*, vol. 16, pp. 1374–1389, 1990.

[10] K. M. Kavi, *Real-Time Systems: Abstractions, Languages, and Design Methodologies*. Los Alamitos, CA: IEEE Press, 1992.

[11] P. Merlin and D. Farber, “Recoverability of communication protocols—Implication of a theoretical study,” *IEEE Trans. Commun.*, vol. COM-24, pp. 1036–1043, Sept. 1976.

[12] T. Murata, “Petri nets: Properties, analysis and applications,” *Proc. IEEE*, vol. 77, pp. 541–584, Apr. 1989.

[13] C. Ramamoorthy and G. Ho, "Performance evaluation of asynchronous concurrent systems using Petri nets," *IEEE Trans. Softw. Eng.*, vol. SE-6, no. 5, pp. 440–449, 1980.

[14] A. D. Stoyemko, C. Hamacher, and R. C. Holt, "Analyzing hard-real-time programs for guaranteed schedulability," *IEEE Trans. Softw. Eng.*, vol. 12, no. 8, pp. 737–750, 1991.

[15] J. J. P. Tsai, S. J. Yang, and Y. H. Chang, "Timing constraint Petri nets and their application to schedulability analysis of real-time system specification," *IEEE Trans. Softw. Eng.*, vol. 21, no. 1, pp. 32–49, 1995.

[16] J. Wang, *Timed Petri Nets: Theory and Application*. Boston, MA: Kluwer, 1998.

[17] J. Wang and Y. Deng, "Incremental modeling and verification of flexible manufacturing systems," *Int. J. Intell. Manuf.*, vol. 10, no. 6, pp. 485–502, 1999.

[18] J. Wang, X. He, and Y. Deng, "Introducing software architecture specification and analysis in SAM through an example," *Info. Softw. Technol.*, vol. 41, no. 7, pp. 451–567, 1999.

[19] M. C. Zhou, *Petri Nets in Flexible and Agile Automation*. Boston, MA: Kluwer, 1995.

[20] W. M. Zuberek, "Timed Petri nets: Definitions, properties, and applications," *Microelectron. Reliab.*, vol. 31, no. 4, pp. 627–644, 1991.

Jiacun Wang received the M.S. and Ph.D. degrees in electrical and computer engineering from Nanjing University of Science and Technology (NUST), China, in 1989 and 1991, respectively.

He is currently a Research Associate in the School of Computer Science, Florida International University, Miami. Previously, he was an Associate Professor at NUST. He authored *Timed Petri Nets: Theory and Application* (Norwell, MA: Kluwer, 1998) and published more than 40 research papers in journals and conferences. His main research interests include software engineering, discrete event systems, formal methods, and distributed information systems.

Dr. Wang served on the Program Committees of the 1994 International Conference on Electronics and Information Technology, Beijing, China, the 1997 IEEE International Conference on Systems, Man, and Cybernetics, Orlando, FL, and the 1998 IEEE International Conference on Systems, Man and Cybernetics, San Diego, CA.

Yi Deng (M'99) received the Ph.D. degree in computer science from the University of Pittsburgh, Pittsburgh, PA, in 1992.

He is an Associate Professor of computer science and Managing Director of the Embedded Software Center (ESC), a joint university-industry R&D consortium, at the University of Texas at Dallas, Richardson. Previously, he was an Associate Professor and founding Director of the Center for Advanced Distributed System Engineering (CADSE), School of Computer Science, Florida International University, Miami. He has published more than 45 papers in various journals and refereed conferences in the above areas. He is an Associate Editor for the *International Journal of Software Engineering and Knowledge Engineering*, a referee, and a program committee member for several journals and conferences. His research has been supported by the NSF, ARO, AFOSR, NASA, and the U.S. Air Force Rome Laboratory. His research interests include software architecture, distributed object technology, secure enterprise systems, and formal methods for complex software systems.

Dr. Deng is a member of ACM. He is the Program Committee Co-Chair for the 10th International Conference on Software Engineering and Knowledge Engineering.

Gang Xu received the M.S. degree in computer science from Florida International University, Miami, in 1999.

He is currently an IP Engineer at AT&T Labs, Lincroft, NJ. His research interests include enterprise level Internet service architecture and infrastructure, Internet service integration, and IP security.