

Timing Constraint Petri Nets and Their Application to Schedulability Analysis of Real-Time System Specifications

Jeffrey J. P. Tsai, *Senior Member, IEEE*, Steve Jennhwa Yang, *Student Member, IEEE*, and Yao-Hsiung Chang

Abstract—In this paper, we present timing constraint Petri nets (or TCPN's for short) and describe how to use them to model a real-time system specification and determine whether the specification is schedulable with respect to imposed timing constraints. The strength of TCPN's over other time-related Petri nets is in the modeling and analysis of conflict structures. Schedulability analysis is conducted in three steps: *specification modeling, reachability simulation, and timing analysis*. First, we model a real-time system by transforming its system specification along with its imposed timing constraints into a TCPN; we call this net N_s . Then we *simulate* the reachability of N_s to verify whether a marking, M_n , is reachable from an initial marking, M_0 . It is important to note that a reachable marking in Petri nets is not necessarily reachable in TCPN's due to the imposed timing constraints. Therefore, in the *timing analysis* step, a reachable marking M_n found in the *reachability simulation* step is analyzed to verify whether M_n is reachable with the timing constraints. M_n is said to be reachable in the TCPN's if and only if we can find at least one firing sequence σ so that all transitions in σ are strongly schedulable with respect to M_0 under the timing constraints. If such M_n can be found, then we can assert that the specification is schedulable under the imposed timing constraints, otherwise the system specification needs to be modified or the timing constraints need to be relaxed. We also present a synthesis method for determining the best approximation of the earliest fire beginning time (EFBT) and the latest fire ending time (LFET) of each strongly schedulable transition.

Index Terms—Timing constraints, Petri nets, time Petri nets, timed Petri nets, specification and verification, timing analysis, real-time systems, synthesis.

I. INTRODUCTION

A REAL-TIME SYSTEM is characterized by its timely response to external stimuli [28]. In general, a response consists of a series of task executions, and a task execution is usually characterized by its *start time, execution time, and deadline* [20]. For obtaining timely responses in a real-time system, a schedule of task executions must be specified, designed, and executed in a timely manner with respect to imposed timing constraints. The process of verifying that a

schedule will meet the timing constraints is referred to as *schedulability analysis* [6], [21], [23]. Techniques proposed for schedulability analysis are usually considered from the design and run-time point of view. Stoyenko et al. present a set of language-independent schedulability analysis techniques [21] where they utilize the information about program organization and implementation and the hardware configuration to verify whether real-time software will meet its timing constraints. Haban and Shin [6] employ a real-time monitoring approach to analyze the monitored results, then feed the analyzed results back to the host operating system for dynamic scheduling and verification of the monitored tasks. Tokuda and Kotera in [23] use an interactive schedulability analyzer, *Scheduler 1-2-3*, for verifying at the design phase whether or not all tasks in a hard real-time system will be completed by their deadlines. Compared with the schedulability analysis presented in these three papers, which concentrated on the design and run-time phases, our focus is on the specification and verification point of view. Jahanian and Mok's real-time logic (RTL) is a typical example of real-time system specification and verification [8]. They are concerned with the consistency problem of safety assertions with respect to a timing specification expressed in RTL. In their approach, a formal logic RTL, employing an event-action model, is used to specify and verify the timing behavior of a real-time system. Once the specification of a system has been given in terms of the event-action model, it then can be transformed into a set of RTL formulas against which the safety assertions are analyzed. If the safety assertions are derivable from the formulas without any contradiction, then the system is considered to be safe. Timing constraint Petri nets (TCPN's) are more like the graphical representation of RTL described in [9] rather than RTL itself. However, TCPN's are more systematic and visualizable because they utilize existing concepts and techniques of Petri nets along with additional representation of timing constraints. The reasons Petri nets are used are given in Section II and a formal definition of TCPN's is presented in Section III. To facilitate the discussion, in this paper we concentrate the analysis on task level and assume resources are always available upon request. Nevertheless, the same analysis method can be applied to more detailed levels such as the function level.

In this paper, we present timing constraint Petri nets (TCPN's) and their use in schedulability analysis of a real-time system specification. TCPN's are derived from the concepts

Manuscript received January 1993; revised December 1993 and September 1994. This work was supported in part by National Science Foundation under Grants CCR-8809381 and CCR-9106540.

J. J. P. Tsai and S. J. Yang are with the Department of Electrical Engineering and Computer Science, University of Illinois at Chicago, Chicago, IL 60607 USA (e-mail: tsai@eecs.uic.edu, yang@kel.eecs.uic.edu).

Y. H. Chang is with LEADWELL CNC Machines Manufacturing Corporation, Taiwan.

IEEE Log Number 9407726.

of timing constraints and Petri nets. Usually, we model a real-time system as a set of tasks. Each task is characterized by two events, the *beginning* event at which a task begins its execution and the *ending* event at which a task completes its execution. A task remains in execution between the two events. In addition, tasks in real-time systems usually have timing constraints such as the ones defined in [3]. Dasarathy has classified the timing constraints; three of them are: 1) the *maximum timing constraint*, which is the maximum amount of time that may elapse between the occurrence of two events; 2) the *minimum timing constraint*, which is the minimum amount of time that must elapse between the occurrence of two events; 3) the *duration timing constraint*, which is the maximum amount of time that a task can last.

Petri nets are a particular kind of directed graph [14]. A typical Petri net has transitions, places, directed arcs, and tokens. As shown in Fig. 1(a), rectangles represent transitions and circles represent places. A transition, t_j is said to be enabled if each input place, p , of t_j has at least $w(p, t_j)$ tokens, where $w(p, t_j)$ is the weight of arc from p to t_j . A net is an ordinary net if all its arcs are weighted as one. For simplicity, in this paper, we refer to nets are ordinary unless noted otherwise. An enabled transition may fire any time after it is enabled. The firing of a transition is instantaneous and will remove one token from each of its input places and put one token into each of its output places [14].

TCPN's extend Petri nets by associating a *maximum timing constraint* and a *minimum timing constraint*, $(TC_{\min}(p_j), TC_{\max}(p_j))$, with each place p_j , or $(TC_{\min}(t_j), TC_{\max}(t_j))$ associated with each transition t_j , and a duration timing constraint, $[FIRE_{\text{dur}}(t_j)]$, associated with each transition t_j . For example, $(TC_{\min}(t_2), TC_{\max}(t_2))$ is denoted as (0, 5) and $[FIRE_{\text{dur}}(t_2)]$ is denoted as [6] as shown in Fig. 1(b).

In general, the analysis of a system specification can be done in two phases, the functional property analysis and the timing property analysis. The functional property analysis is performed first to assert the correctness of the functionality. It can be performed using existing Petri nets analysis methods without considering timing constraints. The timing property analysis is then performed to assert the correctness of the timing behavior. In general, valid timing behavior is assumed if the specification complies with the timing constraints. Since the functionality has been determined before the timing property analysis begins, the functionality relationship among transitions is preserved with the newly attached timing constraints. Due to the space limitations, we only cover the schedulability analysis, which is the timing property analysis performed in our research.

Schedulability analysis is conducted in three steps: *specification modeling*, *reachability simulation*, and *timing analysis*. First, we model a real-time system by transforming its system specification along with its imposed timing constraints into a TCPN; we call this net N_s . Then we simulate the reachability of N_s to verify whether a marking, M_n , is reachable from an initial marking, M_0 . It is important to note that a reachable marking in Petri nets is not necessarily reachable in TCPN's due to the imposed timing constraints. Therefore, in the *timing analysis* step, a reachable marking M_n found in the

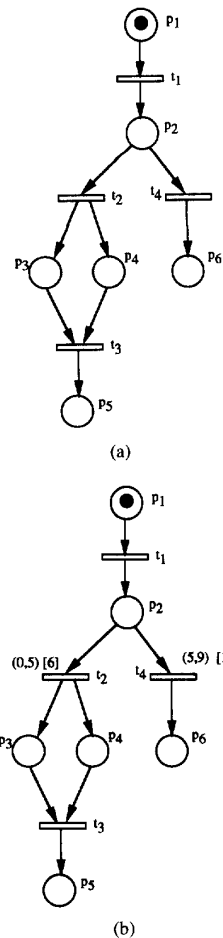


Fig. 1. (a) A fragment of a Petri net. (b) A fragment of a Petri net with timing constraints.

reachability simulation step is analyzed to verify whether M_n is reachable under the consideration of timing constraints. M_n is said to be reachable in the TCPN's if and only if we can find at least one firing sequence, σ , so that all transitions in σ are strongly schedulable with respect to M_0 under the timing constraints. If such M_n can be found, then we can assert that the specification is schedulable under the imposed timing constraints, otherwise the system specification needs to be modified or the timing constraints need to be relaxed.

We also present a synthesis method for determining the best approximation of the earliest fire beginning time (EFBT) and the latest fire ending time (LFET) of each strongly schedulable transition. Our schedulability analysis is performed during the specification phase, whose purpose is to verify a system specification against its imposed timing constraints rather than elaborating a scheduling policy for a particular schedule. Consequently, the analysis method presented in this paper can be used regardless of scheduling algorithms.

This paper is organized as follows. Section II reviews Petri nets and their time-related extensions. Section III defines TCPN's and discusses the usefulness of TCPN's. Section IV

presents schedulability analysis and synthesis using TCPN's. Finally, we conclude this paper by presenting our future research in Section V. An example which illustrates the usability of TCPN's is given in Appendix.

II. PETRI NETS AND THEIR TIME-RELATED EXTENSIONS

Petri nets (PN's) have gained popularity in recent years because of their ability to model and analyze concurrent systems. However, the concept of time is not explicitly provided in PN's, which limits their usefulness for real-time systems. Many efforts to extend PN's can be found in the areas of temporal behavior analysis [1], [2], [4], [5], [10]–[13], [15], [22] and performance evaluation [7], [16]–[19], and we would like to compare these extensions of PN's in this section. Most of the extensions have been achieved by imposing additional timing constraints onto the enabling and firing rules of the original PN's. Therefore, we conduct our comparison from the enabling and firing rules point of view, and from the timing constraints point of view.

We classify enabling rules as two types: *typeless enabling rules* and *typed enabling rules*. Typeless enabling rules treat all tokens as the same. Therefore, for the enabling of a transition, t_j , one only considers the presence of tokens in each input place of t_j . In contrast, *typed enabling rules* treat tokens individually so that each token may possess different attributes. Therefore, for the enabling of a transition, t_j , one not only considers the presence of tokens, but also the types of tokens, i.e., t_j is not enabled until each input place of t_j has the right combination of token types. Based on the classification of *typeless* and *typed enabling rules*, we classify firing rules as *typeless firing rules* and *typed firing rules* accordingly. *Typeless firing rules* are associated with typeless enabling rules which treat all tokens as the same. The result of a transition firing is implicit, which means tokens will be removed from each input place of t_j and added into each output place of t_j based on the arcs' weights. PN's use the *typeless enabling* and *firing rules*. In contrast, for those nets following *typed enabling rules*, the firing rules also have to be typed. The firing of a typed transition, t_j , will remove specific colored tokens from each input place of t_j and add specific colored tokens into each output place of t_j . As a result, a table is used to specify what combinations of input colored tokens can be used to enable transitions, and what combinations of colored tokens should be removed from input places and be added into output places after transition firing. Colored PN's are a typical example of nets that follow both typed enabling and firing rules. For facilitating the discussion, we also classify two firing modes based on how soon an enabled transition has to fire: *weak firing mode* and *strong firing mode*. The *weak firing mode* does not force any enabled transition to fire. In other words, an enabled transition may or may not fire. The *weak firing mode* is used in PN's modeling. The *strong firing mode* forces an enabled transition to fire immediately. In other words, a transition is forced to fire as soon as it is enabled. The *strong firing mode* is used in conflict-free firing PN's modeling [14]. The *strong firing mode* is not suitable for some nets with conflict structures because this mode will result in a contradiction, as explained

in the following example. As shown in Fig. 1(a), once t_1 fires, a token will be added into place p_2 . According to the definition of the *strong firing mode*, both t_2 and t_4 are enabled and begin to fire at the same time when the token arrives at p_2 , which results in a token being added into p_3 , p_4 and p_6 , respectively. This contradicts the definition of conflict structures, in which only one transition can fire.

The imposed timing constraints can be represented as constants or functions. The former includes timed Petri nets which treat a timing constraint as a single delay [7], [17], [18], and time Petri nets which treat a timing constraint as a time pair consisting of lower and upper bounds [10], [13], [19]. The latter includes stochastic PN's which treat a timing constraint as a probability function of transition firing rate [12], [16], and ER nets which treat a timing constraint as a function of colored tokens in input places [4], [5].

Timed Petri nets (timed PN's) were first proposed by Ramchandani [18] who examined the timing analysis of asynchronous concurrent systems. Ramamoorthy and Ho [17] extended the use of timed PN's to the area of performance evaluation. Timed PN's follow the *strong firing mode*, i.e., a transition, t_j , with a delayed time, T_{del} , will immediately fire at time when necessary tokens have arrived at time T_0 . Before T_0 , the tokens which have arrived are not preserved and can be used to enable other transitions if t_j is in a conflict structure. During the time period from T_0 to $(T_0 + T_{del})$, the tokens are preserved for t_j so that no other transitions can use these tokens. At time $(T_0 + T_{del})$, the tokens will and must be removed from t_j 's input places to output places.

Stochastic Petri nets (SPN's) are also mainly used for performance evaluation. In contrast to the constant delay used in timed PN's, SPN's use the average delay which is a probability function of a transition's firing rate. Peng and Shin use generalized stochastic Petri nets (GSPN's) to model the real-time control activities of a distributed system [16]. The modeled activities in GSPN's are then formed as a sequence of homogeneous continuous-time Markov chains (CTMC) in order to analyze the probability of missing deadlines. Timed and stochastic PN's are mostly used for performance evaluation, such as finding how fast a transition can initiate consecutive firing in a periodically operated timed PN's (SPN's). In other words, the performance is evaluated by finding a minimum cycle time for completing a firing sequence (each transition fires at least once) that leads back to the initial marking, i.e., finding the minimum cycle time for the execution of a periodical process.

Time Petri nets (time PN's) were introduced by Merlin and Farber [13] for analyzing the recoverability of communication protocols. Time PN's are similar to timed PN's except time PN's use a time pair instead of a single delay. A transition in time PN's is associated with (TC_{min}, TC_{max}) , where TC_{min} represents the minimum delay and the TC_{max} represents a time-out. If a transition, t_j , is enabled at time T_0 , then t_j can fire neither before $(T_0 + TC_{min})$ nor after $(T_0 + TC_{max})$. Since time PN's follow *strong firing mode*, if the firing does not take place during the time period from $(T_0 + TC_{min})$ to $(T_0 + TC_{max})$, then t_j must fire at $(T_0 + TC_{max})$. Leveson and Stolzy use time PN's to model a real-time system for safety

analysis [10]. To perform the safety analysis, a reachability graph of the time PN's which models the system behavior is first constructed to determine whether a high-risk state will be reached. The timing constraints imposed by the modeled system can then be derived to avoid such high risk states. Berthomieu and Diaz propose an enumerative method to analyze the temporal behavior of a concurrent system [1]. Through the technique of reachability analysis used in time PN's, Berthomieu and Diaz claim that their method can exhaustively validate the behavior of the modeled system.

We summarize the above extensions of PN's by distinguishing their difference in firing modes. Timed (stochastic) PN's use the *strong firing mode* in which a transition is forced to fire immediately after it is enabled. Firing of a transition will last for a period of time, and tokens will be preserved during the firing period. Time PN's use the *strong firing mode* in which a transition is forced to fire at time $T_0 + TC_{\max}$ if the transition has not fired and not been disabled by other transitions' firing. In time PN's modeling, tokens will not be preserved because the firing of a transition is instantaneously.

III. TIMING CONSTRAINT PETRI NETS

Time constraint Petri nets (TCPN's) are inspired by time and timed Petri nets. TCPN's extend Petri nets by adding *minimum timing constraint* and *maximum timing constraint* pairs to places or transitions, and by adding *durational timing constraint* to transitions or places. The major difference between TCPN's and time (timed) PN's is that TCPN's follow the *weak firing mode* and the analysis method is based on either the relative or absolute time mode, whereas time (timed) PN's follow the *strong firing mode* and the analysis method is based on the relative time mode only. We will illustrate the advantage of using the *weak firing mode* and the *absolute time mode* at the end of this section. TCPN's are most suitable for systems with conflict structures (priority, decision, and choice structures). The formal definition of TCPN's is:

A. Definition of TCPN's

A timing constraint Petri net is a 6-tuple $\langle P, T, F, C, D, M \rangle$ where:

P is a set of places, i.e., $P = \{p_1, p_2, \dots, p_m\}$;

T is a set of transitions, i.e., $T = \{t_1, t_2, \dots, t_n\}$;

F is a set of arcs which connects places and transitions;

C is a set of integer pairs, $(TC_{\min}(pt_j), TC_{\max}(pt_j))$, where pt_j is either a place or a transition;

D is a set of firing durations, $\{FIRE_{\text{dur}}(pt_j)\}$,

where pt_j is either a place or a transition, and

M is a set of marking with m -vector,

$\{M_{(p_1)}, \dots, M_{(p_j)}, \dots, M_{(p_m)}\}$, where $M_{(p_j)}$ denotes the numbers of token in place p_j . M_0 denotes the initial marking.

A transition t_j with a time pair, $(TC_{\min}(t_j), TC_{\max}(t_j))$, is said to be enabled if each of its input places has at least one token. A transition t_j , which is enabled at time T_0 , is said to be *irable* during the time period from $T_0 + TC_{\min}(t_j)$ to $T_0 + TC_{\max}(t_j)$ in which $TC_{\max}(t_j) \geq TC_{\min}(t_j)$. A firable transition can fire but there is no guarantee that the firing will complete successfully because the firing of a transition takes a

period of time $FIRE_{\text{dur}}(t_j)$. All the tokens (denoted as TK's) used for enabling a transition t_j will be preserved during the t_j 's firing, and TK's can be used to enable other transitions if t_j fails to complete the firing. If all the transitions enabled by TK's fail to complete their firing, TK's will be trapped in their corresponding places. A transition is said to be schedulable if it is firable and can complete its firing successfully. i.e., $(TC_{\max}(t_j) - TC_{\min}(t_j)) \geq FIRE_{\text{dur}}(t_j)$. A marking M_n is said to be reachable in PN's modeling if there is a firing sequence, $\sigma = (M_0 t_1 M_1 \dots M_j t_j \dots t_n M_n)$ that transforms M_0 to M_n . The set of all possible markings reachable from M_0 is denoted by $R(M_0)$, and the set of all possible firing sequences from M_0 is denoted by $L(M_0)$ [14]. With the consideration of timing constraints, in TCPN's modeling, a marking M_n is said to be *reachable* if and only if all transitions in σ are proved to be *schedulable* with respect to the timing constraints, i.e., $(TC_{\max}(t_j) - TC_{\min}(t_j)) \geq FIRE_{\text{dur}}(t_j)$.

In general, the time pairs associated with transitions are referred to as *transition time pairs* and the time pairs associated with places are referred to as *place time pairs*. For the duration times which are associated with transitions like timed Petri nets do, we say they are *transition durations*. If duration times are associated with places like Coolahan's approach [2], we say they are *place durations*. For those places and transitions do not have explicit timing constraints associated with them, the default values of a *place time pair* (*transition time pairs*) are (zero, infinity) and the default value of a *transition duration* is zero.

PN's and TCPN's have different behavior in reachability problems. A marking M_n which is reachable in PN's modeling is not necessarily reachable in TCPN's modeling because of the imposed timing constraints. In PN's modeling, a marking M_n is reachable if there exist at least one firing sequence, σ , which transforms M_0 to M_n after firing all transitions in σ at least once. However, due to the imposed timing constraints in TCPN's, such as $TC_{\max}(t_j) \geq TC_{\min}(t_j)$ and $(TC_{\max}(t_j) - TC_{\min}(t_j)) \geq FIRE_{\text{dur}}(t_j)$, transitions in σ may not be firable or schedulable according to the definitions of TCPN's modeling. If at least one transition in σ is nonfirable or nonschedulable, then M_n is nonreachable. For example in the PN's in Fig. 1(a), let $M_0 = (p_1)$, $M_3 = (p_5)$, and $M_4 = (p_6)$. Both M_3 and M_4 are reachable after firing sequence $\sigma_1 = (t_1 t_2 t_3)$ and $\sigma_2 = (t_1 t_4)$, respectively. Assuming M_0, M_3 , and M_4 remain the same, we can impose timing constraints onto t_2 and t_4 and form a TCPN as shown in Fig. 1(b). In this case, based on the definition of a schedulable transition, t_2 in σ_1 is nonschedulable because $(5-0) < 6$ whereas t_4 in σ_2 is schedulable because $(9-5) > 1$. That is M_3 is nonreachable whereas M_4 is still reachable in TCPN's modeling.

To deal with such differences of reachability problems in PN's and TCPN's modeling, a systematic method is required to determine whether M_n is reachable in TCPN's. In case M_n is nonreachable, then the method can be used to find the cause as well, i.e., find the transitions that caused M_n to be nonreachable. In this paper, our schedulability analysis is used to pinpoint which transitions in σ are not schedulable when a marking in the TCPN's is nonreachable.

B. Significance of TCPN's

We conclude this section by distinguishing the advantage of TCPN's over the other time-related extensions of PN's. Time (timed) PN's conduct their analysis based on relative time mode, which is only suitable for transition time pairs. In addition, both time and timed PN's follow the *strong firing mode*, which not only violates the PN's firing mode, but also prohibits them from representing conflict structures. TCPN's allow either *transition* or *place time pairs* and are able to do analysis based on absolute or relative time modes. TCPN's follow the *weak firing mode*, which not only preserve the same firing mode used by PN's, but also is capable of modeling and analyzing conflict structures.

Most of the time-related PN's are based on relative time. This will not cause any problem as long as the timing constraints can be modeled purely by *transition time pairs*. However, purely *transition time pairs* are not always sufficient. For example, [10] notes that two nets with either *transition time pairs* or *place time pairs* are isomorphic because they can be converted into each other without losing timing information. Unfortunately, this is true only when the token arrival times are not considered. We will use an example to demonstrate the need to know the token arrival time in order to perform certain timing analysis. In addition, both time and timed PN's use the *strong firing mode* which will cause problems when modeling conflict structures.

Fig. 2(a) shows a mutual exclusion example which models the use of an exclusive resource between two tasks *A* and *B*. Places and transitions are discussed as follows:

- If place p_1 has a token, then a resource is available.
- If place $p_2(p_4)$ has a token, then task *A*(*B*) is ready to acquire a resource.
- If place $p_3(p_5)$ has a token, then task *A*(*B*) is using a resource.
- If transition $t_1(t_4)$ fires, then task *A*(*B*) is requesting a resource.
- If transition $t_2(t_5)$ fires, then task *A*(*B*) is acquiring a resource.
- If transition $t_3(t_6)$ fires, then task *A*(*B*) is releasing a resource.
- If transition t_7 fires, then task *B* is destroying a resource.

Assume a resource can be acquired no earlier than 5 time units and no later than 15 time units after the resource is available. We attach a time pair (5, 15) onto place p_1 as shown in Fig. 2(a) to represent the time interval a resource is available. This example also shows not every timing constraint can be modeled by *transition time pairs*. To demonstrate *transition* and *place time pairs* are not isomorphic unless the token arrival times are considered, we convert the net with *place time pairs* as shown in Fig. 2(a) into a net with *transition time pairs* as shown in Fig. 2(b) without considering token arrival times. Unfortunately, the meaning of timing constraints specified in Fig. 2(a) is distorted because token arrival times in p_1 , p_2 and p_4 are ignored. The timing constraint in Fig. 2(a) specifies that the token in place p_1 may be used to enable t_2 and t_5 only between time 5 to 15 after the token arrived at p_1 .

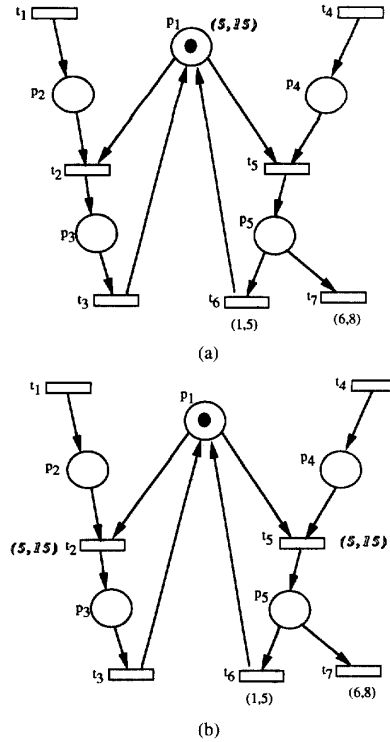


Fig. 2. (a) A mutual exclusion example with *place time pairs*. (b) A mutual exclusion example with *transition time pairs*.

However, the conversion shown in Fig. 2(b) implies that token in p_1 can be used to enable t_2 and t_5 indefinitely if neither t_1 nor t_4 fire. As a result, the conversion from *place time pairs* to *transition time pairs* is isomorphic only if one takes the token arrival time into consideration.

The conversion from *transition time pairs* to *place time pairs* with the consideration of token arrival times is done by computing the maximum of each input place's TC_{\min} and the minimum of each input place's TC_{\max} . In Fig. 2(a), let t_1 fire at time $T1$, t_4 fire at time $T4$, and a token arrive at p_1 at time T . Since a transition is enabled if and only if each of its input place has at least one token. Therefore, in this example, we have

$$t_2 \text{ is enabled during } (\text{Max}\{T1 + 0, T + 5\}, \text{Min}\{T1 + \infty, T + 15\}),$$

and

$$t_5 \text{ is enabled during } (\text{Max}\{T4 + 0, T + 5\}, \text{Min}\{T4 + \infty, T + 15\}).$$

Since ∞ is always greater than $T + 15$, we have

$$t_2 \text{ is enabled during } (\text{Max}\{T1 + 0, T + 5\}, T + 15), \text{ and}$$

$$t_5 \text{ is enabled during } (\text{Max}\{T4 + 0, T + 5\}, T + 15).$$

As we addressed in the introduction, for a transition to be fireable, the timing constraint must be $TC_{\min} \leq TC_{\max}$. Therefore, we have

$$\text{Max}\{T1 + 0, T + 5\} \leq T + 15$$

and

$$\text{Max}\{T4 + 0, T + 5\} \leq T + 15.$$

Thus, we conclude that transitions t_2 and t_5 are fireable during time intervals, $(\text{Max}\{T1, T + 5\}, T + 15)$ and $(\text{Max}\{T4, T + 5\}, T + 15)$, respectively. From this conclusion, we can demonstrate another problem due to the *strong firing mode*. In Fig. 2(b), if both t_2 and t_5 do not fire by time $(T + 15)$, according to the definition of the strong firing mode, then both t_2 and t_5 must fire at time $(T + 15)$. This contradicts the definition of conflict structures. Beside causing two conflict transitions to fire simultaneously, the *strong firing mode* may cause a transition not to be able to be fired at all. In other words, the *strong firing mode* is prone to cause dead transition. For example, in Fig. 2(b) transition t_7 may never have a chance to fire if t_6 is with a time pair $(1, 5)$ and t_7 is with a time pair $(6, 10)$, because t_6 is forced to fire at time $T5 + 5$ (assume t_5 fires at $T5$) if t_6 has not fired until $T5 + 5$. Consequently, t_7 will never fire and become a dead transition.

Another advantage the *weak firing mode* is in the modeling of lost tokens. In [10], [13], a failure transition and a fault condition are needed explicitly in order to model the loss of tokens because of the use of the *strong firing mode*. This results in a number of shortcomings: 1) it violates the assumption of PN's that tokens never accidentally disappear. 2) designers have to explicitly model every possible location where tokens may be lost. 3) Extra transitions and places increase the complexity of both net and reachability graphs. By using TCPN's, i.e. use the *weak firing mode* for modeling such failure-fault correlation (failures causing faults), we do not need those extra failure transitions and fault conditions. In addition, TCPN's modeling provides an intuitive representation of lost tokens without violating PN's modeling by utilizing the existence of trapped tokens to simulate the existence of lost tokens. If a transition, t_j , does not fire because of lost token (faults), then one simply looks at all places prior to t_j in a firing sequence for any trapped tokens. If any trapped token can be found in place, p , we detect that the faults are caused because the output transition of p fails to fire during the timing constraints (failures). For example, in Fig. 2(a), if t_6 does not fire in time (a fault), then the cause of fault (a failure) may be that a token has lost in place p_4 because transition t_5 fails to fire in time. This shows how timing constraint violations could result in the lose of tokens.

IV. SCHEDULABILITY ANALYSIS USING TCPN'S

In this section, we present our schedulability analysis with an illustrated example featuring a real-time controlling system. We assume that the state in which the system samples data from outside as the initial marking M_0 , and assume the state in which the system provides timely response as the final marking M_n . For a system specification to be valid, M_n must be reachable from M_0 in TCPN's modeling. For M_n to be reachable, all transitions in the firing sequences from M_0 to M_n must be strongly schedulable. We will address how to determine if a transition is strongly schedulable in TCPN's modeling in this section.

A. Basic Concepts

As defined in [14], a marking, M_n , is said to be *reachable* if there is a firing sequence, $\sigma = (M_0 t_1 M_1 \cdots M_i t_i \cdots t_n M_n)$ or simply $(t_1 \cdots t_i \cdots t_n)$ that transforms M_0 to M_n . Due to the timing constraints, to prove M_n is reachable in TCPN's modeling, we have to prove that all the transitions in σ are strongly schedulable with respect to M_0 . In other words, let t_n be the final transition of σ from M_0 to M_n . M_n is reachable if and only if t_n and all the transitions that occurred prior to t_n are strongly schedulable. In this section, we present a method to determine whether a transition is strongly schedulable under this definition. The firing sequence, δ , used in TCPN's is different from the firing sequence, σ , used in PN's. We define δ as follows.

$\delta_k(M_n)$ is the collection of places and transitions except the first transition in the k -th firing sequence from M_0 to M_n , where n is the number of the final transition in $\delta_k(M_n)$.

For example, comparing δ and σ in Fig. 1(b), $\delta_1(M_3) = (p_2 t_2 p_3 t_3)$, $\delta_2(M_3) = (p_2 t_2 p_4 t_3)$ and $\delta_1(M_4) = (p_2 t_4)$, whereas $\sigma(M_3) = (t_1 M_1 t_2 M_2 t_3) = (t_1 p_2 t_2 p_3 p_4 t_3)$ and $\sigma(M_4) = (t_1 M_1 t_4) = (t_1 p_2 t_4)$.

Note that $\delta_k(M_n)$ is a by-product of finding σ in PN's, i.e., $\delta_k(M_n)$ can be obtained from $L(M_0)$ (see Section III.A). Therefore, no additional mechanism is needed to find $\delta_k(M_n)$, and the complexity of our schedulability analysis is the same as the reachability problem in PN's. The complexity is $O(mn)$ where m is the number of conflicting transitions (conditional flows) and n is the number of concurrent transitions. Terms are defined here to facilitate the following discussion.

Definition 1: $\text{FIRE}_{\text{enabled}}(t_j)$ is the time at which a transition t_j is enabled; $\text{FIRE}_{\text{begin}}(t_j)$ is the time at which a transition t_j begins the firing; $\text{FIRE}_{\text{end}}(t_j)$ is the time at which a transition t_j ends the firing; $\text{FIRE}_{\text{dur}}(t_j)$ is a period of time during which a transition t_j keeps firing; From Definition 1, we have

$$\text{FIRE}_{\text{dur}}(t_j) = \text{FIRE}_{\text{end}}(t_j) - \text{FIRE}_{\text{begin}}(t_j) \quad (4.1)$$

$$\text{FIRE}_{\text{enabled}}(t_j) \leq \text{FIRE}_{\text{begin}}(t_j). \quad (4.2)$$

Definition 2: $\text{IT}(p_j)$ is a set of input transitions of p_j ; $\text{OT}(p_j)$ is a set of output transitions of p_j ; $\text{IP}(t_j)$ is a set of input places of t_j ; $\text{OP}(t_j)$ is a set of output places of t_j .

Definition 3: $\text{TC}_{\text{min}}(p_j)/\text{TC}_{\text{max}}(p_j)$ are the minimum/maximum elapsed time intervals between the token arrival time of p_j and the beginning/ending firing times of p_j 's output transition. $\text{TOKEN}_{\text{arr}}(p_j)$ is the time at which a token arrives at a place p_j ; $\text{TOKEN}_{\text{rem}}(p_j)$ is the time at which a token is removed from a place p_j ; From Definition 3, we have

$$\text{TC}_{\text{min}}(p_j) \leq \text{FIRE}_{\text{begin}}(t_i) - \text{TOKEN}_{\text{arr}}(p_j),$$

$$\text{TC}_{\text{max}}(p_j) \geq \text{FIRE}_{\text{end}}(t_i) - \text{TOKEN}_{\text{arr}}(p_j),$$

where $t_i \in \text{OT}(p_j)$. The movement of tokens from each input place to each output place of t_j uses no time. As a result, the token removal time of place p_j is the same as the ending firing time of p_j 's output transition. Similarly, the token arrival time of p_j is the same as the ending firing time of p_j 's input

transition. i.e.,

$$\text{TOKEN}_{\text{rem}}(p_j) = \text{FIRE}_{\text{end}}(t_i), \text{ where } t_i \in OT(p_j) \quad (4.3)$$

$$\text{TOKEN}_{\text{arr}}(p_j) = \text{FIRE}_{\text{end}}(t_i), \text{ where } t_i \in IT(p_j) \quad (4.4)$$

$$\text{TOKEN}_{\text{rem}}(p_j) \leq \text{TOKEN}_{\text{arr}}(p_j) + \text{TC}_{\text{max}}(p_j). \quad (4.5)$$

A place needs at least $\text{TC}_{\text{min}}(p_j)$ seconds of delay to enable a transition and can hold a token only for at most $\text{TC}_{\text{max}}(p_j)$ after a token's arrival. Therefore, a place can only enable its output transition during the time from $(\text{TOKEN}_{\text{arr}}(p_j) + \text{TC}_{\text{min}}(p_j))$ to $(\text{TOKEN}_{\text{arr}}(p_j) + \text{TC}_{\text{max}}(p_j))$. As a result, the maximum enabling time needed by p_j to enable its output transition, $OT(p_j)$, is $(\text{TC}_{\text{max}}(p_j) - \text{TC}_{\text{min}}(p_j))$. Since the firing of $OT(p_j)$ needs $\text{FIRE}_{\text{dur}}(OT(p_j))$ and it has to be enabled before it can be FIRED, therefore, we have

$$\text{FIRE}_{\text{dur}}(t_i) \leq \text{TC}_{\text{max}}(p_j) - \text{TC}_{\text{min}}(p_j), \text{ where } t_i \in OT(p_j). \quad (4.6)$$

otherwise, t_i cannot be Fired successfully.

B. Specification of a Real-Time Control System

For a typical real-time system, the system continuously samples and processes the data in a timely fashion. Fig. 3(a) is a TCPN transformed from the specifications list below. The time pair of minimum and maximum timing constraints is denoted as $(\text{TC}_{\text{min}}, \text{TC}_{\text{max}})$ such as $(0, 45)$, and the firing duration is denoted as $[\text{FIRE}_{\text{dur}}]$ such as $[25]$ in the net. The system samples data (denoted as transition t_1) every 50 time units (TU) from the external environment and provides either a normal response (denoted as transition t_6) or an emergency response (denoted as transition t_9) within 45 TU. If the system cannot complete before its deadline, then it tries the emergency response. Therefore, we model the normal data input (denoted as transition t_2) with a higher priority than the emergency data input (denoted as transition t_7) by specifying t_2 with an earlier deadline. As we can see in Fig. 3(a), a conflict structure $(p_2, t_2, \text{ and } t_7)$ can be easily modeled with TCPN's. In contrast, time (timed) PN's do not have such capabilities. In TCPN's modeling, assume a token arrives at p_2 at time T , then t_7 may fire during $T + 10$ to $T + 30$ if t_2 does not fire at $T + 6$. In contrast, in time (timed) PN's modeling, t_7 will never fire because t_2 must fire at time $T + 6$. Therefore, let the initial marking, M_0 , be (p_1) . We would like to use schedulability analysis to determine whether marking $M_6, (p_1 p_{12})$, which denotes the normal response, and marking $M_9, (p_1 p_{13})$, which denotes the emergency response are reachable. The following specifications are for a real-time control system.

S1. The system samples data from the external environment every 50 TU and the system must respond with either a normal or an emergency response but not both within 45 TU after the completion of sampling. The task of *sampling* takes 5 TU.

S2. Once the data is obtained, the system goes to the normal data input procedure (NDI) first, if the NDI fails, then it goes to the emergency data input procedure (EDI).

S3. NDI must wait for 2 TU and complete within 6 TU after the completion of the sampling. The task of NDI takes 8 TU.

S4. EDI must wait for 10 TU and complete within 30 TU after the completion of the sampling. The task of EDI takes 2 TU.

S5. If the system executes the normal data input procedure, then the system needs to display the sampled data on the screen before giving its normal response. The task of the normal response takes 10 TU.

S6. Displaying data is done by displaying the analyzed data on the background. (i.e., data has to wait until both the background display and the data analysis are completed). The task of displaying data takes 15 TU.

S7. Displaying background can begin as soon as NDI is complete. The task of displaying background takes 5 TU.

S8. Due to the data transfer delay, analyzing data must wait for 5 TU after the completion of NDI. The task of analyzing data takes 25 TU.

S9. If the system goes to the emergency data input procedure, then the system only needs to analyze data and gives the emergency responses, (no data displaying is necessary). The task of emergency response takes 10 TU.

S10. Due to the data transfer delay, analyzing data must wait for 5 TU after the completion of EDI. The task of analyzing data takes 20 TU.

C. Analyzing Strongly Schedulable Transitions

For a transition t_j to be fireable, it must be enabled by all its input places. Therefore, we have Definition 4.

Definition 4: A transition t_j is said to be a *weakly fireable transition* and denoted as $WFT(t_j)$ if each of the input places of t_j currently has at least one token without considering the arrival times of tokens in each input place (or assuming that the arrival times of tokens are the same). That is, given the time pair, $(\text{EFBT}(t_j), \text{LFET}(t_j))$, which denotes the earliest fire beginning time and the latest fire ending time of t_j after t_j is enabled, then t_j is a *WFT* if and only if

$$\text{EFBT}(t_j) = \text{Max}\{\text{TC}_{\text{min}}(p_j)\} + \text{TC}_{\text{min}}(t_j),$$

$$\text{LFET}(t_j) = \text{Min}\{\text{TC}_{\text{max}}(p_j), \text{TC}_{\text{max}}(t_j)\},$$

and

$$\text{LFET}(t_j) - \text{EFBT}(t_j) \geq 0, \text{ where } p_j \in IP(t_j).$$

Since the firing needs a duration time, it is not guaranteed that a fireable transition will complete its firing successfully. Therefore, we have.

Definition 5: A transition t_j is a *weakly schedulable transition* and denoted as $WST(t_j)$ if t_j is a *WFT* and can be fired successfully without considering the arrival times of tokens in each input place. That is, t_j 's duration of firing time cannot be less than its fireable time, so that

$$\text{LFET}(t_j) - \text{EFBT}(t_j) \geq \text{FIRE}_{\text{dur}}(t_j).$$

For example in Fig. 3(a), t_2 is not a *WST* because

$$\begin{aligned} \text{LFET}(t_2) - \text{EFBT}(t_2) &= \text{Min}\{\text{TC}_{\text{max}}(p_2), \\ &\quad \text{TC}_{\text{max}}(t_2)\} - (\text{Max}\{\text{TC}_{\text{min}}(p_2)\} + \text{TC}_{\text{min}}(t_2)) \\ &= 6 - 2 \leq 8 = \text{FIRE}_{\text{dur}}(t_2). \end{aligned}$$

To make t_2 a *WST*, we relax the timing constraints by increasing $\text{TC}_{\max}(t_2)$ from 6 to 10, so that

$$\text{LFET}(t_2) - \text{EFBT}(t_2) = 10 - 2 = 8 = \text{FIRE}_{\text{dur}}(t_2).$$

Theorem 1: A transition t_j is said to be a *strongly firable transition* with respect to Mo and denoted as *SFT* (t_j) if t_j is a WFT and each of the input places of t_j has at least one token at the same time with the consideration of the actual arrival time of a token in each input place (or take into account the actual arrival time of tokens). That is, given the time pair, $(\text{EFBT}(t_j), \text{LFET}(t_j))$, which denotes the earliest fire beginning time and the latest fire ending time of t_j after t_j is enabled, then t_j is a *SFT* if and only if

$$\text{EFBT}(t_j) = \text{Max}\{\text{Min}\{\text{TOKEN}_{\text{arr}}(p_j)\} + \text{TC}_{\min}(p_j)\} + \text{TC}_{\min}(t_j) \quad (4.a)$$

$$\text{LFET}(t_j) = \text{Min}\{\text{Max}\{\text{TOKEN}_{\text{arr}}(p_j)\} + \text{Min}\{\text{TC}_{\max}(p_j), \text{TC}_{\max}(t_j)\}\} \quad (4.b)$$

$$\text{LFET}(t_j) - \text{EFBT}(t_j) \geq 0, \quad \text{where } p_j \in \text{IP}(t_j) \quad (4.c)$$

$$\begin{aligned} \text{EFBT}(t_j) &= \text{Max}\{\text{FIRE}_{\text{end}}(t_1) + \sum \text{TC}_{\min}(pt_{mk}) \\ &\quad + \sum \text{FIRE}_{\text{dur}}(t_{nk})\} + \text{TC}_{\min}(t_j), \\ &\quad \text{where } pt_{mk}, t_{nk} \in \delta_k(M_j) \end{aligned}$$

$$\begin{aligned} \text{LFET}(t_j) &= \text{Min}\{\text{Min}\{\text{TC}_{\max}(P_j), \text{TC}_{\max}(t_j)\} \\ &\quad + \text{FIRE}_{\text{end}}(t_1) + \sum \text{TC}_{\max}(pt_{mk})\}, \\ &\quad \text{where } pt_{mk} \in \delta_k(M_j), \end{aligned}$$

$$\text{LFET}(t_j) - \text{EFBT}(t_j) \geq 0.$$

Proof: t_j is firable only if it is enabled by every input place of t_j simultaneously, and will stop firing once one of its input places stops enabling it. To determine if t_j is firable, we need to take into account the arrival time of all input places of t_j . Since $\text{TC}_{\min}(p_j)$ and $\text{TC}_{\max}(p_j)$ are fixed, we need only to find the lower bound and upper bound of $\text{TOKEN}_{\text{arr}}(p_j)$ in order to find $\text{EFBT}(t_j)$ and $\text{LFET}(p_j)$. From (4.4), we know that $\text{TOKEN}_{\text{arr}}(p_j)$ is equal to $\text{FIRE}_{\text{end}}(\text{IT}(p_j))$. Therefore, we can find the bounds of $\text{TOKEN}_{\text{arr}}(p_j)$ by finding the bounds of $\text{FIRE}_{\text{end}}(\text{IT}(p_j))$ derived from $\text{IP}(p_j)$ back to the initial transition t_1 .

From Definition 3, we have

$$\text{TOKEN}_{\text{arr}}(p_j) + \text{TC}_{\min}(p_j) + \text{TC}_{\min}(t_j) \leq \text{FIRE}_{\text{begin}}(t_i),$$

where

$$t_i \in \text{OT}(p_j) \quad (4.7)$$

$$\text{FIRE}_{\text{end}}(t_i) \leq \text{TOKEN}_{\text{arr}}(p_j) + \text{TC}_{\max}(p_j),$$

where

$$t_i \in \text{OT}(p_j). \quad (4.8)$$

From (4.1), we have

$$\text{FIRE}_{\text{begin}}(t_i) = \text{FIRE}_{\text{end}}(t_i) - \text{FIRE}_{\text{dur}}(t_i),$$

where

$$t_i \in \text{OT}(p_j). \quad (4.9)$$

Applying (4.9) into (4.7), we have

$$\begin{aligned} \text{TOKEN}_{\text{arr}}(p_j) + \text{TC}_{\min}(p_j) + \text{TC}_{\min}(t_j) \\ \leq \text{FIRE}_{\text{end}}(t_i) - \text{FIRE}_{\text{dur}}(t_i), \quad \text{where } t_i \in \text{OT}(p_j) \end{aligned}$$

or

$$\begin{aligned} \text{TOKEN}_{\text{arr}}(p_j) + \text{TC}_{\min}(p_j) + \text{TC}_{\min}(t_j) + \text{FIRE}_{\text{dur}}(t_i) \\ \leq \text{FIRE}_{\text{end}}(t_i) \quad \text{where } t_i \in \text{OT}(p_j). \end{aligned} \quad (4.10)$$

From (4.8) and (4.10), we have

$$\begin{aligned} \text{TOKEN}_{\text{arr}}(p_j) + \text{TC}_{\min}(p_j) + \text{TC}_{\min}(t_j) + \text{FIRE}_{\text{dur}}(t_i) \\ \leq \text{FIRE}_{\text{end}}(t_i) \leq \text{TOKEN}_{\text{arr}}(p_j) + \text{TC}_{\max}(p_j), \end{aligned}$$

where

$$t_i \in \text{OT}(p_j) \quad (4.11)$$

or

$$\begin{aligned} \text{FIRE}_{\text{end}}(t_i) + \text{TC}_{\min}(p_j) + \text{TC}_{\min}(t_j) + \text{FIRE}_{\text{dur}}(t_i) \\ \leq \text{FIRE}_{\text{end}}(t_i) \leq \text{FIRE}_{\text{end}}(t_i) \text{TC}_{\max}(p_j), \end{aligned}$$

where

$$t_i \in \text{OT}(p_j). \quad (4.12)$$

From $\delta_k(M_j)$ and (4.4), we know that $\text{TOKEN}_{\text{arr}}(p_j) = \text{FIRE}_{\text{end}}(t_{j-1})$, which means we can find $\text{TOKEN}_{\text{arr}}(p_j)$ by finding $\text{FIRE}_{\text{end}}(t_{j-1})$. Thus we have

$$\text{TOKEN}_{\text{arr}}(p_j) = \text{FIRE}_{\text{end}}(t_{j-1}). \quad (4.13)$$

We can determine the bounds of $\text{TOKEN}_{\text{arr}}(p_j)$ by deriving the bound of $\text{FIRE}_{\text{end}}(t_{j-1})$ as follows:

From (4.12), we have

$$\begin{aligned} \text{FIRE}_{\text{end}}(t_{j-2}) + \text{TC}_{\min}(p_{j-1}) + \text{FIRE}_{\text{dur}}(t_{j-1}) \\ \leq \text{FIRE}_{\text{end}}(t_{j-1}) \leq \text{FIRE}_{\text{end}}(t_{j-2}) + \text{TC}_{\max}(p_{j-1}) \end{aligned} \quad (4.14)$$

extending $\text{FIRE}_{\text{end}}(t_{j-2})$ by using (4.12), we have

$$\begin{aligned} \text{FIRE}_{\text{end}}(t_{j-3}) + \text{TC}_{\min}(p_{j-2}) + \text{FIRE}_{\text{dur}}(t_{j-2}) \\ \leq \text{FIRE}_{\text{end}}(t_{j-2}) \leq \text{FIRE}_{\text{end}}(t_{j-3}) \\ + \text{TC}_{\max}(p_{j-2}). \end{aligned} \quad (4.15)$$

Combining (4.14) and (4.15), we have

$$\begin{aligned} \text{FIRE}_{\text{end}}(t_{j-3}) + \text{TC}_{\min}(p_{j-2}) + \text{TC}_{\min}(p_{j-1}) \\ + \text{FIRE}_{\text{dur}}(t_{j-2}) + \text{FIRE}_{\text{dur}}(t_{j-1}) \leq \text{FIRE}_{\text{end}}(t_{j-1}) \\ \leq \text{FIRE}_{\text{end}}(t_{j-3}) + \text{TC}_{\max}(p_{j-2}) + \text{TC}_{\max}(p_{j-1}). \end{aligned} \quad (4.16)$$

We can continue deriving in the same way until $\text{FIRE}_{\text{end}}(t_1)$ is encountered, at which point we have

$$\begin{aligned} \text{FIRE}_{\text{end}}(t_1) + \text{TC}_{\min}(p_2) + \text{TC}_{\min}(p_3) \cdots + \text{TC}_{\min}(p_{j-1}) \\ + \text{TC}_{\min}(t_j) + \text{FIRE}_{\text{dur}}(t_2) + \text{FIRE}_{\text{dur}}(p_3) \\ + \cdots + \text{FIRE}_{\text{dur}}(t_{j-1}) \\ \leq \text{FIRE}_{\text{end}}(t_{j-1}) \leq \text{FIRE}_{\text{end}}(t_1) \\ + \text{TC}_{\max}(p_2) + \text{TC}_{\max}(p_3) + \cdots + \text{TC}_{\max}(p_{j-1}) \end{aligned} \quad (4.17)$$

or

$$\begin{aligned} & \text{FIRE}_{\text{end}}(t_1) + \Sigma \text{TC}_{\text{min}}(pt_{\text{mk}}) + \text{TC}_{\text{min}}(t_j) \\ & + \Sigma \text{FIRE}_{\text{dur}}(t_{\text{nk}}) \leq \text{FIRE}_{\text{end}}(t_{j-1}) \\ & \leq \text{FIRE}_{\text{end}}(t_1) + \Sigma \text{TC}_{\text{max}}(pt_{\text{mk}}), \end{aligned}$$

where

$$pt_{\text{mk}}, t_{\text{nk}} \in \delta_k(M_j). \quad (4.18)$$

Replacing $\text{FIRE}_{\text{end}}(t_{j-1})$ by $\text{TOKEN}_{\text{arr}}(p_j)$, we can determine the lower bound and upper bound of $\text{TOKEN}_{\text{arr}}(p_j)$ and get:

$$\begin{aligned} & \text{FIRE}_{\text{end}}(t_1) + \Sigma \text{TC}_{\text{min}}(pt_{\text{mk}}) + \text{TC}_{\text{min}}(t_j) \\ & + \Sigma \text{FIRE}_{\text{dur}}(t_{\text{nk}}) \leq \text{TOKEN}_{\text{arr}}(p_j) \\ & \leq \text{FIRE}_{\text{end}}(t_1) + \Sigma \text{TC}_{\text{max}}(pt_{\text{mk}}) \end{aligned}$$

where

$$pt_{\text{mk}}, t_{\text{nk}} \in \delta_k(M_j). \quad (4.19)$$

From (4.19), we get the lower bound and upper bound of $\text{TOKEN}_{\text{arr}}(p_j)$ and apply them to (4.a) and (4.b) to get

$$\begin{aligned} \text{EFBT}(t_j) &= \text{Max}\{\text{FIRE}_{\text{end}}(t_1) + \Sigma \text{TC}_{\text{min}}(pt_{\text{mk}}) \\ & + \Sigma \text{FIRE}_{\text{dur}}(t_{\text{nk}})\}, \end{aligned}$$

where

$$\begin{aligned} & pt_{\text{mk}} \in \delta_k(M_j) \text{ and } t_{\text{nk}} \in \delta_k(M_j) \\ \text{LFET}(t_j) &= \text{Min}\{(\text{FIRE}_{\text{end}}(t_1) + \Sigma \text{TC}_{\text{max}}(pt_{\text{mk}}))\}, \end{aligned}$$

where

$$pt_{\text{mk}} \in \delta_k(M_j)$$

i.e.,

- $\text{EFBT}(t_j)$ = The lower bound of t_j 's enabled time of all paths from t_1 to t_j .
- $\text{LFET}(t_j)$ = The upper bound of t_j 's end firing time of all paths from t_1 to t_j .

For example, in Fig. 3(a), t_6 is not a SFT because:

$$\begin{aligned} \delta_1(M_6) &= (p_2 t_2 p_3 t_3 p_5 t_5 p_7 t_6) \\ \delta_2(M_6) &= (p_2 t_2 p_4 t_4 p_6 t_5 p_7 t_6) \\ \delta_3(M_6) &= (p_{10} t_6). \end{aligned}$$

Let t_1 end its firing at T_0 .

$$\begin{aligned} & \text{EFBT}(t_6) \\ &= \text{Max}\{[T_0 + (\text{TC}_{\text{min}}(p_2) + \text{TC}_{\text{min}}(t_2) \\ & + \text{TC}_{\text{min}}(p_3) + \text{TC}_{\text{min}}(t_3) + \text{TC}_{\text{min}}(p_5) \\ & + \text{TC}_{\text{min}}(t_5) + \text{TC}_{\text{min}}(p_7)) + (\text{FIRE}_{\text{dur}}(t_2) \\ & + \text{FIRE}_{\text{dur}}(t_3) + \text{FIRE}_{\text{dur}}(t_5))], \\ & [T_0 + (\text{TC}_{\text{min}}(p_2) + \text{TC}_{\text{min}}(t_2) + \text{TC}_{\text{min}}(p_4) \\ & + \text{TC}_{\text{min}}(t_4) + \text{TC}_{\text{min}}(p_6) + \text{TC}_{\text{min}}(t_5) \\ & + \text{TC}_{\text{min}}(p_7)) + (\text{FIRE}_{\text{dur}}(t_2) \\ & + \text{FIRE}_{\text{dur}}(t_4) + \text{FIRE}_{\text{dur}}(t_5))], \\ & [T_0 + \text{TC}_{\text{min}}(p_{10})] + \text{TC}_{\text{min}}(t_6)\} \\ &= \text{Max}\{[T_0 + (0 + 2 + 0 + 0 + 0 + 0 + 0) \\ & + (8 + 5 + 15)], [T_0 + (0 + 2 + 5 + 0 \\ & + 0 + 0 + 0) + (8 + 25 + 15)], [T_0 + 0]\} + 0 \\ &= \text{Max}\{(T_0 + 2 + 28), (T_0 + 7 + 48), (T_0 + 0)\} + 0 \\ &= T_0 + 53 \end{aligned}$$

$\text{LFET}(t_6)$

$$\begin{aligned} &= \text{Min}\{\{\text{Min}\{\text{TC}_{\text{max}}(p_7), \text{TC}_{\text{max}}(t_6)\} \\ & + T_0 + (\text{TC}_{\text{max}}(p_2) + \text{TC}_{\text{max}}(t_2) + \text{TC}_{\text{max}}(p_3) \\ & + \text{TC}_{\text{max}}(t_3) + \text{TC}_{\text{max}}(p_5) + \text{TC}_{\text{max}}(t_5)], \\ & [\text{Min}\{\text{TC}_{\text{max}}(p_7), \text{TC}_{\text{max}}(t_6)\} + T_0 \\ & + (\text{TC}_{\text{max}}(p_2) + \text{TC}_{\text{max}}(t_2) + \text{TC}_{\text{max}}(p_4) \\ & + \text{TC}_{\text{max}}(t_4) + \text{TC}_{\text{max}}(p_6) + \text{TC}_{\text{max}}(t_5)], \\ & [\text{Min}\{\text{TC}_{\text{max}}(p_{10}), \text{TC}_{\text{max}}(t_6)\} + T_0]\} \\ &= \text{Min}\{[\infty + T_0 + (\infty + 6 + \infty + \infty \\ & + \infty + \infty)], [\infty + T_0 + (\infty + 6 + \infty + \infty \\ & + \infty + \infty)], [45 + T_0]\} \\ &= T_0 + 45 \end{aligned}$$

$$\text{LFET}(t_6) - \text{EFBT}(t_6)$$

$$= (T_0 + 45) - (T_0 + 53) = 45 - 53 < 0$$

which contradicts Theorem 1.

Therefore, t_6 is not a SFT.

To make t_6 a SFT, we relax the timing constraints by increasing the $\text{TC}_{\text{max}}(p_{10})$ from 45 to 65, as shown in Fig. 3(b), so that

$$\text{LFET}(t_6) = T_0 + 65 \text{ and}$$

$$\text{LFET}(t_6) - \text{EFBT}(t_6) = (T_0 + 65) - (T_0 + 53) = 12 > 0.$$

Theorem 2: A transition t_j is said to be a *strongly schedulable transition* with respect to M_0 and denoted as $\text{SST}(t_j)$ if t_j can be fired successfully under the consideration of the arrival time of a token in each input place. That is, given the time pair, $(\text{EFBT}(t_j), \text{LFET}(t_j))$, which denotes the earliest fire beginning time and the latest fire ending time of t_j after t_j is enabled, then t_j is a SST if and only if

$$\begin{aligned} \text{EFBT}(t_j) &= \text{Max}\{\text{Min}\{\text{TOKEN}_{\text{arr}}(p_j)\} \\ & + \text{TC}_{\text{min}}(p_j)\} + \text{TC}_{\text{min}}(t_j) \end{aligned} \quad (4.2)$$

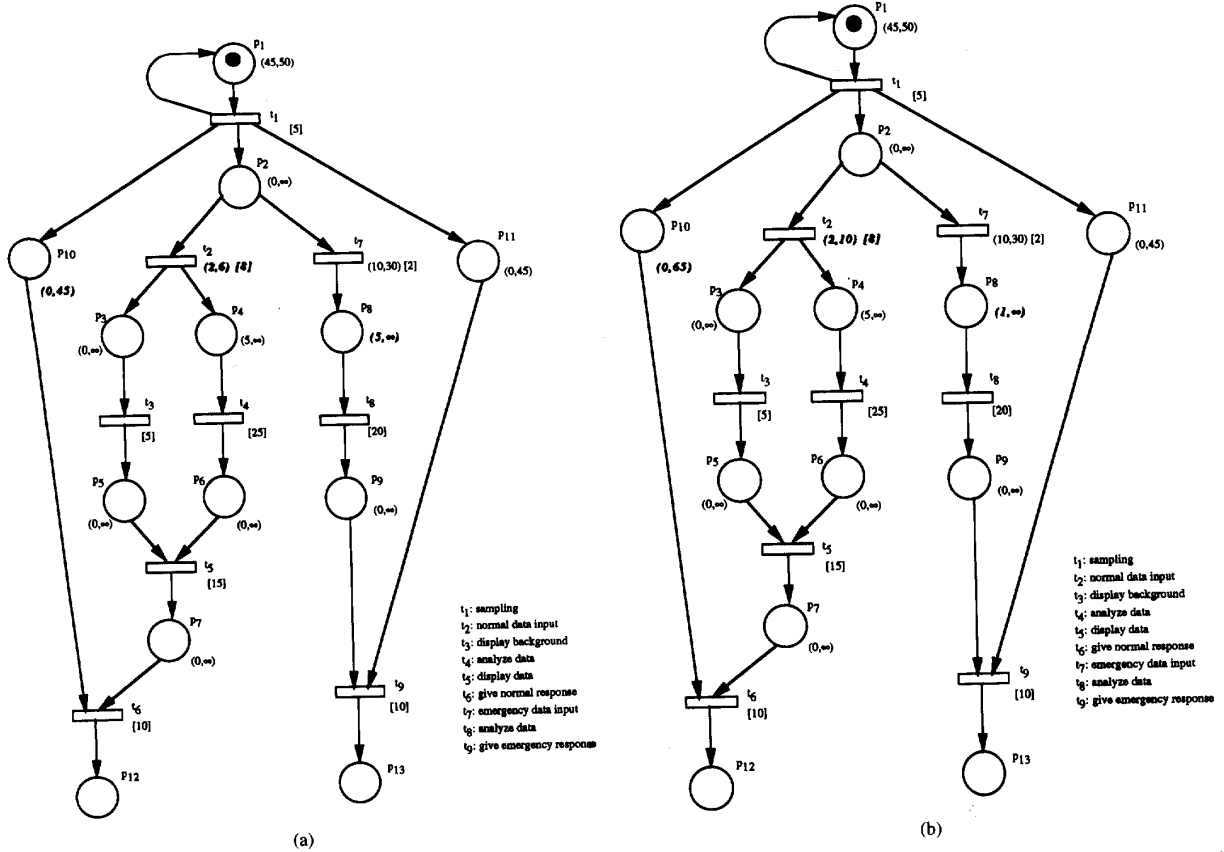


Fig. 3. (a) A fragment of a TCPN modeling a real-time control system specification. (b) A fragment of a TCPN with strongly schedulable TC_{\min} and TC_{\max} on each transition. (Continued on next page.)

$$LFET(t_j) = \text{Min}\{\text{Max}\{\text{TOKEN}_{\text{arr}}(p_j)\} + \text{Min}\{TC_{\max}(p_j), TC_{\max}(t_j)\}\} \quad (4.b)$$

$$LFET(t_j) - EFBT(t_j) \geq \text{FIRE}_{\text{dur}}(t_j),$$

where

$$p_j \in \text{IP}(t_j) \quad (4.d)$$

or

$$EFBT(t_j) = \text{Max}\{\text{FIRE}_{\text{end}}(t_1) + \sum TC_{\min}(pt_{\text{mk}}) + \sum \text{FIRE}_{\text{dur}}(t_{\text{nk}})\} + TC_{\min}(t_j),$$

where

$$LFET(t_j) = \text{Min}\{(\text{Min}\{TC_{\max}(P_j), TC_{\max}(t_j)\} + \text{FIRE}_{\text{end}}(t_1) + \sum TC_{\max}(pt_{\text{mk}}))\}$$

where

$$LFET(t_j) - EFBT(t_j) \geq \text{FIRE}_{\text{dur}}(t_j), \text{ where } p_j \in \text{IP}(M_j).$$

Proof: The proof of this theorem is the same as the proof of Theorem 1 except that 4.d is replaced by (4.c) in all equations.

For example, in Fig. 3(a), t_9 is not a SST because:

$$\delta_1(M_9) = (p_2 t_7 p_8 t_8 p_9 t_9)$$

$$\delta_2(M_9) = (p_{11} t_9).$$

Let t_1 end its firing at T_0 .

$$\begin{aligned} EFBT(t_9) &= \text{Max}\{[T_0 + (TC_{\min}(p_2) + TC_{\min}(t_7) \\ &\quad + TC_{\min}(p_8) + TC_{\min}(t_8) + TC_{\min}(p_9)) \\ &\quad + (\text{FIRE}_{\text{dur}}(t_7) + \text{FIRE}_{\text{dur}}(t_8))], \\ &\quad [T_0 + TC_{\min}(p_{11})]\} + TC_{\min}(t_9) \\ &= \text{Max}\{[T_0 + (0 + 10 + 5 + 0 + 0) + (2 + 20)], \\ &\quad [T_0 + 0]\} + 0 \\ &= T_0 + 37 \end{aligned}$$

$$\begin{aligned} LFET(t_9) &= \text{Min}\{[\text{Min}\{TC_{\max}(p_9), TC_{\max}(t_9)\} \\ &\quad + T_0 + TC_{\max}(p_2) + TC_{\max}(t_7) \\ &\quad + TC_{\max}(p_8) + TC_{\max}(t_8)], \end{aligned}$$

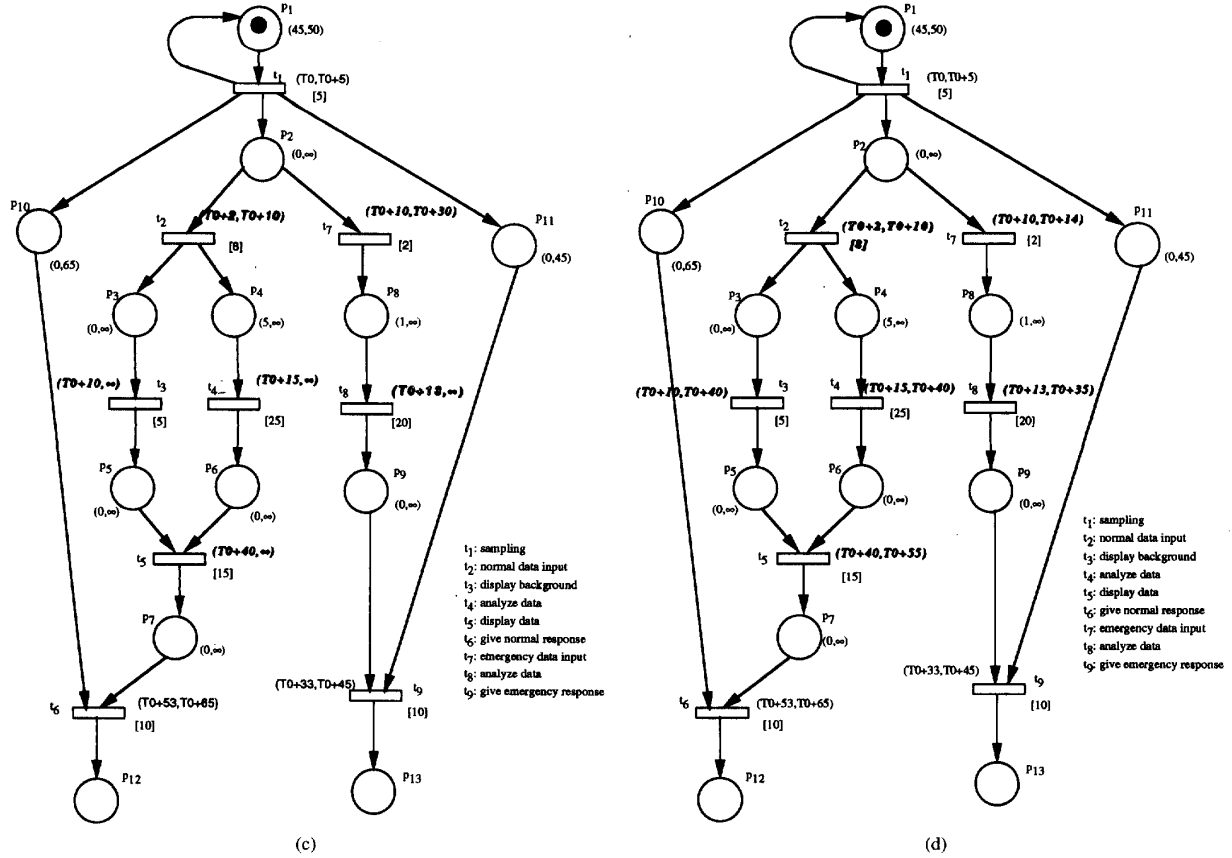


Fig. 3. (Continued.) (c) A fragment of a TCPN with synthesized EFBT and LFET on each transition after forward computation. (d) A fragment of a TCPN with synthesized EFBT and LFET on each transition after backward computation.

$$\begin{aligned} & \{\text{Min}\{\text{TC}_{\max}(p_{11}), \text{TC}_{\max}(t_9)\} + T_0\} \\ &= \text{Min}\{[\infty + T_0 + \infty + 30 + \infty + \infty], [45 + T_0]\} \\ &= T_0 + 45. \end{aligned}$$

$$\begin{aligned} \text{LFET}(t_9) - \text{EFBT}(t_9) &= (T_0 + 45) - (T_0 + 37) \\ &= 8 < \text{FIRE}_{\text{dur}}(t_9) = 10 \end{aligned}$$

which contradicts Theorem 2. Therefore, t_9 is not a SST.

To make t_9 a SST, we relax the timing constraints by reducing the $\text{TC}_{\min}(p_8)$ from 5 to 1, as shown in Fig. 3(b), so that we have

$$\begin{aligned} \text{EFBT}(t_9) &= \text{Max}\{[T_0 + (0 + 10 + 1 + 0 + 0) \\ &\quad + (2 + 20)], [T_0 + 0] + 0\} \\ &= T_0 + 33 \\ \text{LFET}(t_9) &= T_0 + 45. \\ \text{LFET}(t_9) - \text{EFBT}(t_9) &= (T_0 + 45) - (T_0 + 33) \\ &= 12 > \text{FIRE}_{\text{dur}}(t_9) = 10. \end{aligned}$$

Theorem 3: If at least one of the transitions in a TCPN is not strongly schedulable with respect to the initial marking M_0 , then this net is not a schedulable net.

Proof: For a transition t_j in a TCPN, it is firable for only a limited period of time. In addition, the firing of a transition in a TCPN lasts for a duration of time. If the firable time of t_j is less than its firing duration, then t_j cannot complete its firing successfully. In other words, t_j is not schedulable under the imposed timing constraints.

D. Synthesis

Real-time system specification and verification tries to find the most timely valid responses for the system. Most of the verification methods proposed nowadays are able to verify whether a timely response is valid with respect to the timing requirements, but not many of them can synthesize the approximate range of such timely responses. Besides proving those timely responses are schedulable with respect to the specified timing requirements, the other objective of our schedulability analysis is to synthesize the best approximation of time range for the timely response. In this paper, our synthesis is conducted by computing the best approximation of the earliest fire beginning time (EFBT) and the latest fire ending time (LFET) for each schedulable transition. The computation of synthesis consists of forward and backward computations. Forward computation is used to construct the

best approximation of EFBT and backward computation is used to construct the best approximation of LFET.

1) *Forward Computation*: In general, the forward computation follows the schedulability analysis method (especially the Theorem 2) presented in previous subsection. That is, the forward computation converts the specified TC_{\min} and TC_{\max} time pairs into the EFBT and LFET time pairs, then analyzes whether the EFBT and LFET time pairs are strongly schedulable or not. If they are not, then the EFBT and LFET are refined by relaxing TC_{\min} and TC_{\max} until all transitions are strongly schedulable. Note that the time mode used in TC_{\min} and TC_{\max} is referred to as the relative time between the two transitions' firing, whereas the time mode used in EFBT and LFET is absolute time with respect to the time instant of initial marking. In the following, we use forward computation to compute the EFBT and LFET of each transition based on the refined TC_{\min} and TC_{\max} specification in Fig. 3(b). The result of forward computation is shown in Fig. 3(c). The EFBT and LFET of t_6 and t_9 as shown in Fig. 3(c) have been obtained in a previous subsection and they are both strongly schedulable transitions with respect to the time instant of initial marking, i.e., the time instant at which transition t_0 completes its firing, we assume the time is T_0 .

$$\begin{aligned} \text{LFET}(t_6) - \text{EFBT}(t_6) &= (T_0 + 65) - (T_0 + 53) \\ &= 12 > \text{FIRE}_{\text{dur}}(t_6) = 10. \end{aligned}$$

$$\begin{aligned} \text{LFET}(t_9) - \text{EFBT}(t_9) &= (T_0 + 45) - (T_0 + 33) \\ &= 12 > \text{FIRE}_{\text{dur}}(t_9) = 10. \end{aligned}$$

Based on Theorem 2, we compute and analyze the remaining transitions in the net in Fig. 3(b) by finding their firing sequence, δ , and EFBT and LFET as follows:

For transition, t_2 , whose $\delta(M_2) = (p_2t_2)$, and we have

$$\begin{aligned} \text{EFBT}(t_2) &= T_0 + TC_{\min}(p_2) + TC_{\min}(t_2) \\ &= T_0 + 0 + 2 \\ &= T_0 + 2 \end{aligned}$$

$$\begin{aligned} \text{LFET}(t_2) &= \text{Min}\{TC_{\max}(p_2), TC_{\max}(t_2)\} + T_0 \\ &= 10 + T_0 \end{aligned}$$

$$\begin{aligned} \text{LFET}(t_2) - \text{EFBT}(t_2) &= T_0 + 10 - (T_0 + 2) = 8 \\ &= \text{FIRE}_{\text{dur}}(t_2) = 8. \end{aligned}$$

Therefore, t_2 is a SST.

For transition, t_3 , whose $\delta(M_3) = (p_2t_2p_3t_3)$, we have

$$\begin{aligned} \text{EFBT}(t_3) &= T_0 + TC_{\min}(p_2) + TC_{\min}(t_2) \\ &\quad + TC_{\min}(p_3) + \text{FIRE}_{\text{dur}}(t_2) + TC_{\min}(t_3) \\ &= T_0 + 0 + 2 + 0 + 8 + 0 \\ &= T_0 + 10 \end{aligned}$$

$$\begin{aligned} \text{LFET}(t_3) &= \text{Min}\{TC_{\max}(p_3), TC_{\max}(t_3)\} \\ &\quad + T_0 + TC_{\max}(p_2) + TC_{\max}(t_2) \\ &= \infty + T_0 + \infty + 10 \\ &= \infty \end{aligned}$$

$$\begin{aligned} \text{LFET}(t_3) - \text{EFBT}(t_3) &= \infty - (T_0 + 10) = \infty \\ &> \text{FIRE}_{\text{dur}}(t_3) = 5. \end{aligned}$$

Therefore, t_3 is a SST.

For transition, t_4 , whose $\delta(M_4) = (p_2t_2p_4t_4)$, we have

$$\begin{aligned} \text{EFBT}(t_4) &= T_0 + TC_{\min}(p_2) + TC_{\min}(t_2) + TC_{\min}(p_4) \\ &\quad + \text{FIRE}_{\text{dur}}(t_2) + TC_{\min}(t_4) \\ &= T_0 + 0 + 2 + 5 + 8 + 0 \\ &= T_0 + 15 \end{aligned}$$

$$\begin{aligned} \text{LFET}(t_4) &= \text{min}\{TC_{\max}(p_4), TC_{\max}(t_4)\} + T_0 \\ &\quad + TC_{\max}(p_2) + TC_{\max}(t_2) \\ &= \infty + T_0 + \infty + 10 \\ &= \infty \end{aligned}$$

$$\begin{aligned} \text{LFET}(t_4) - \text{EFBT}(t_4) &= \infty - (T_0 + 15) = \infty \\ &> \text{FIRE}_{\text{dur}}(t_4) = 25. \end{aligned}$$

Therefore, t_4 is a SST.

For transition, t_5 , whose $\delta_1(M_5) = (p_2t_2p_3t_3p_5t_5)$ and $\delta_2(M_5) = (p_2t_2p_4t_4p_6t_5)$, we have

$$\begin{aligned} \text{EFBT}(t_5) &= \text{Max}\{[T_0 + TC_{\min}(p_2) + TC_{\min}(t_2) \\ &\quad + TC_{\min}(p_3) + TC_{\min}(t_3) + TC_{\min}(p_5) \\ &\quad + (\text{FIRE}_{\text{dur}}(t_2) + \text{FIRE}_{\text{dur}}(t_3))], \\ &\quad [T_0 + TC_{\min}(p_2) + TC_{\min}(t_2) + TC_{\min}(p_4) \\ &\quad + TC_{\min}(t_4) + TC_{\min}(p_6) + (\text{FIRE}_{\text{dur}}(t_2) \\ &\quad + \text{FIRE}_{\text{dur}}(t_4))]\} + TC_{\min}(t_5) \\ &= \text{Max}\{[T_0 + (0 + 2 + 0 + 0 + 0) + (8 + 5)], \\ &\quad [T_0 + (0 + 2 + 5 + 0 + 0) + (8 + 25)]\} + 0 \\ &= \text{Max}\{(T_0 + 2 + 13), (T_0 + 7 + 33)\} \\ &= T_0 + 40 \end{aligned}$$

$$\begin{aligned} \text{LFET}(t_5) &= \text{Min}\{[\text{Min}\{TC_{\max}(p_5), TC_{\max}(t_5)\} \\ &\quad + T_0 + TC_{\max}(p_2) + TC_{\max}(t_2) + TC_{\max}(p_3) \\ &\quad + TC_{\max}(t_3)], \\ &\quad [\text{Min}\{TC_{\max}(p_6), TC_{\max}(t_5)\} \\ &\quad + T_0 + TC_{\max}(p_2) + TC_{\max}(t_2) + TC_{\max}(p_4) \\ &\quad + TC_{\max}(t_4)]\} \\ &= \text{Min}\{[\infty + T_0 + \infty + 10 + \infty + \infty], \\ &\quad [15 + T_0 + \infty + 10 + \infty + \infty]\} \\ &= \infty \end{aligned}$$

$$\begin{aligned} \text{LFET}(t_5) - \text{EFBT}(t_5) &= \infty - (T_0 + 40) = \infty \\ &> \text{FIRE}_{\text{dur}}(t_5) = 15. \end{aligned}$$

Therefore, t_5 is a SST.

For transition, t_7 , whose $\delta(M_7) = (p_2t_7)$, we have

$$\begin{aligned} \text{EFBT}(t_7) &= T0 + \text{TC}_{\min}(p_2) + \text{TC}_{\min}(t_7) \\ &= T0 + 0 + 10 \\ &= T0 + 10 \end{aligned}$$

$$\begin{aligned} \text{LFET}(t_7) &= \text{Min}\{\text{TC}_{\max}(p_2), \text{TC}_{\max}(t_7)\} + T0 \\ &= 30 + T0 \end{aligned}$$

$$\begin{aligned} \text{LFET}(t_7) - \text{EFBT}(t_7) &= T0 + 30 - (T0 + 10) = 20 \\ &> \text{FIRE}_{\text{dur}}(t_7) = 2. \end{aligned}$$

Therefore, t_7 is a SST.

For transition, t_8 , whose $\delta(M_8) = (p_2t_7p_8t_8)$, we have

$$\begin{aligned} \text{EFBT}(t_8) &= T0 + \text{TC}_{\min}(p_2) + \text{TC}_{\min}(t_7) \\ &\quad + \text{TC}_{\min}(p_8) + \text{FIRE}_{\text{dur}}(t_7) + \text{TC}_{\min}(t_8) \\ &= T0 + 0 + 10 + 1 + 2 + 0 \\ &= T0 + 13 \end{aligned}$$

$$\begin{aligned} \text{LFET}(t_8) &= \text{Min}\{\text{TC}_{\max}(p_8), \text{TC}_{\max}(t_8)\} \\ &\quad + T0 + \text{TC}_{\max}(p_2) + \text{TC}_{\max}(t_7) \\ &= \infty + T0 + \infty + 30 \\ &= \infty \end{aligned}$$

$$\begin{aligned} \text{LFET}(t_8) - \text{EFBT}(t_8) &= \infty - (T0 + 13) = \infty \\ &> \text{FIRE}_{\text{dur}}(t_8) = 20. \end{aligned}$$

Therefore, t_8 is a SST.

2) *Backward Computation:* As shown in Fig. 3(c), we highlight those transitions whose $\text{LFET}(t_i)$ are denoted as infinity because the default value of TC_{\max} is infinity. In this subsection, we use backward computation to refine such infinity to a meaningful upper bound for each transition. The backward computation is based on the temporal order of a transition's firing within a firing sequence. If we can find a transition, t_j , whose $\text{LFET}(t_j)$ is bounded, then we can find an upper bound of LFET for all the transitions that fired before t_j . The backward computation is conducted by computing the $\text{LFET}(t_j)$ for each transition, t_j , using the following equation:

$$\begin{aligned} \text{LFET}(t_j) &= \text{Min}\{\text{LFET}(t_k) - \text{FIRE}_{\text{dur}}(t_k) - \text{TC}_{\min}(pk)\} \\ &\quad \text{where } p_k \in \text{OP}(t_j) \text{ and } p_k \in \text{IP}(t_k). \end{aligned}$$

The computation of $\text{LFET}(t_j)$ for each transition, t_j , with infinity shown in Fig. 3(c) is as follows, and the result is shown and highlighted in Fig. 3(d)

$$\begin{aligned} \text{LFET}(t_5) &= \text{LFET}(t_6) - \text{FIRE}_{\text{dur}}(t_6) - \text{TC}_{\min}(P_6) \\ &= (T0 + 65) - 10 - 0 \\ &= T0 + 55 \end{aligned}$$

$$\begin{aligned} \text{LFET}(t_3) &= \text{LFET}(t_5) - \text{FIRE}_{\text{dur}}(t_5) - \text{TC}_{\min}(P_4) \\ &= (T0 + 55) - 15 - 0 \\ &= T0 + 40 \end{aligned}$$

$\text{LFET}(t_4)$

$$\begin{aligned} &= \text{LFET}(t_5) - \text{FIRE}_{\text{dur}}(t_5) - \text{TC}_{\min}(P_5) \\ &= (T0 + 55) - 15 - 0 \\ &= T0 + 40 \end{aligned}$$

$$\begin{aligned} \text{LFET}(t_2) &= \text{Min}\{[\text{LFET}(t_3) - \text{FIRE}_{\text{dur}}(t_3) - \text{TC}_{\min}(P_2)], \\ &\quad [\text{LFET}(t_4) - \text{FIRE}_{\text{dur}}(t_4) - \text{TC}_{\min}(P_3)]\} \\ &= \text{Min}\{(T0 + 40 - 5 - 0), (T0 + 40 - 25 - 5)\} \\ &= \text{Min}\{T0 + 35, T0 + 10\} \\ &= T0 + 10 \end{aligned}$$

$$\begin{aligned} \text{LFET}(t_8) &= \text{LFET}(t_9) - \text{FIRE}_{\text{dur}}(t_9) - \text{TC}_{\min}(P_8) \\ &= (T0 + 45) - 10 - 0 \\ &= T0 + 35 \end{aligned}$$

$$\begin{aligned} \text{LFET}(t_7) &= \text{LFET}(t_8) - \text{FIRE}_{\text{dur}}(t_8) - \text{TC}_{\min}(P_7) \\ &= (T0 + 35) - 20 - 1 \\ &= T0 + 14. \end{aligned}$$

Combining forward and backward computation provides designers of a real-time system with a more meaningful timing specification.

E. Run-Time Analysis of the Schedulability Problem

The analysis presented so far is completely based on the specification in TCPN's before the real execution. For a safe and schedulable TCPN, we say that it is an asserted TCPN with respect to the given requirements specification. A similar analysis can be performed after the execution for the run-time testing and debugging, since the post-mortem analysis is based on the collected execution histories [24], [25], [28]. If any unsafe place or nonschedulable transition is detected, then the detected places and the transitions are the most likely causes of the timing constraint violations.

Theorem 4: A transition which is live in PN's is not necessarily a *weakly* or *strongly schedulable transition* in TCPN's. However, a transition which is dead in PN's is definitely a nonschedulable transition in TCPN's.

Proof: Due to the stringent firing rules of TCPN's, the firing of a transition can only be started and must be completed within the time pair $(\text{TC}_{\min}, \text{TC}_{\max})$, or the transition is disabled. Therefore, a live transition in PN's could become dead because of the imposed timing constraints.

Theorem 5: A place which is safe in PN's and time (timed) PN's is not necessary safe in TCPN's. However, a place which is safe in TCPN's is definitely safe in both PN's and time (timed) PN's.

Proof: According to the strong firing mode of time PN's, a transition must fire and will consume one token from each of its input places at the maximum timing constraint. In contrast, in TCPN's, a nonschedulable transition will not be fired in this case, and will not consume any token which causes its input places to accumulate tokens and makes the net unsafe.

Theorem 6: For a nonsafe place p_j in TCPN's, the firing sequence which includes p_j is the bottleneck of performance, or the output transitions of p_j is too slow to meet the timing constraints.

Proof: For a nonsafe place p_j , a token is trapped because the firing sequence which includes p_j takes too much time to move a token into p_j . Thus, p_j 's output transitions do not have enough time to consume tokens. A bottleneck in the system performance has then been found.

For example, in PN's or time (timed) PN's modeling, all places shown in the net fragment in Fig. 3(b) are safe. However, it is not the case in TCPN's modeling and analysis, because any place may be nonsafe after a few runs of the specification if some of the transitions are too slow to react. Assuming place p_0 is nonsafe after a few runs of the specification, then we can detect that transition, t_1 (sampling) is too slow to collect enough data. To solve this, we can modify the specification by relaxing the timing constraints in certain places.

V. CONCLUSION AND FUTURE RESEARCH

TCPN's and their application to the **schedulability analysis** of a real-time system specification are presented in this paper. Our **schedulability analysis** always computes the worst case of the scheduling of a system specification with respect to the imposed timing constraints. Once all transitions in a firing sequence of a final marking M_n in a TCPN are verified to be strongly schedulable with respect to the initial marking, we say that such a specification is asserted with respect to its timing requirements. In addition, throughout the example of modeling and analyzing a real-time control system, we successfully demonstrated that it is possible to perform the specification and the scheduling analysis using the same representation, that is using TCPN's.

The analysis presented in this paper is based on the system specification in a TCPN before its execution. However, we believe a similar analysis can be applied as well to the run-time testing and debugging. In our future research, we will integrate our former research results [24]–[27] with the **schedulability analysis** method presented in this paper. The target code of an asserted specification will be executed on a realistic or simulated target system to collect its run-time information. Once the run-time information is collected, along with our visualization and debugging method, the **schedulability analysis** is able to detect the run-time errors with respect to the timing requirements from the specification phase.

APPENDIX

In this appendix, we use a patient monitoring system to illustrate the usability of our approach. We show how TCPN's can easily model concurrence, and how TCPN's can clearly represent timing requirements. TCPN's are also useful in denoting conflicting transitions or case conditions based on their timing priorities. It will also be shown in this example that additional timing assertions can be easily verified against the timing constraints.

A patient monitoring system is used to monitor a patient's pulse rate and oxygen saturation based on the signals obtained from the sensors attached on patient's skin. In this example, we divide a monitoring system into three subsystems: a **sampling subsystem**, an analysis subsystem, and an **alarm subsystem**. Their functional and timing specification as well as the imposed timing assertions (timing constraints) are described as follows.

1) Functional Specifications:

FS1: The three subsystems, sampling, analysis, and alarm subsystems run concurrently when the power is on.

FS2: The sampling subsystem samples the pulse signal periodically, the analysis subsystem is activated each time a sampled signal is received from the sampling system. The alarm subsystem is activated only when an abnormal signal is detected.

FS3: Once the signals are sampled, the signals are processed by one of the three processors, then sent to the analysis subsystem for the analysis of pulse rate and oxygen saturation. The sampling and analysis subsystems communicate asynchronously.

FS4: After receiving sampled signals from the sampling subsystem, the analysis subsystem calculates and displays the analyzed signals on a screen, and compares them against the preset index figures to detect any abnormal situations.

FS5: The analysis subsystem always looks for an abnormal situation first. If the analyzed figures are greater or less than the index Figs., the analysis subsystem sends a message to the alarm subsystem and displays the abnormal signals. If no abnormal is detected, it simply displays the normal signals. The analysis and alarming subsystems communicate synchronously.

FS6: The alarm subsystem continues to signal an alarm until someone turns off the alarm.

2) Timing Specifications:

TS1: Assume the sampling subsystem samples the pulse signal every 30 milliseconds (ms). The time for pulse searching is 20 ms.

TS2: A signal will be processed by an available processor which has the highest priority and is ready for processing. Processor 1 ($P1$) has the highest priority for processing a signal. If $P1$ is still busy for 1 ms after the signal has arrived, then $P2$ will process the signal. If both $P1$ and $P2$ are busy for 2 ms after the signal has arrived, then $P3$ will process the signal. It takes 5 ms for each processor to process a signal and another 5 ms to prepare the processor for the next signal.

TS3: A signal will be ready after 2 ms and must be ready for sending within 5 ms after the signal is processed. Sending a signal takes 6 ms.

TS4: In the analysis subsystem, receiving a signal from the sampling subsystem takes 5 ms, calculating the figures for signals takes 20 ms, displaying the background on screen takes 10 ms, and displaying the calculated figures on screen takes 10 ms. Normal signals are ready for display 6 ms later if no abnormal signals are detected. It takes 2 ms for the normal signal to be displayed, and it must be ready within

8 ms. Detecting any abnormal situation takes 5 ms. The analyzed signals are always displayed regardless of whether they are normal or not, and both takes 10 ms.

TS5: In the analysis subsystem, sending takes 5 ms and because of the communication delay, the message is not available until 5 ms later and must be received within 20 ms after it has been sent.

TS6: In the alarm subsystem, receiving messages takes 5 ms, while returning acknowledgments from a receiver takes another 5 ms. Due to communication delays, the acknowledgment is not available until 5 ms later and must be received within 20 ms after it has been sent. It takes 5 ms to initialize first alarm. The sampling subsystem also takes 5 ms to receive the acknowledgments.

3) Timing Assertions:

TA1: Signals must be sent from the sampling subsystem to the analysis subsystem within 15 ms once the signals are sampled.

TA2: Once the analysis subsystem receives the sampled signals, it must display the analyzed signals within 50 ms if they are normal.

TA3: Once the analysis subsystem receives the sampled signals, it must display the analyzed signals within 50 ms if they are abnormal.

TA4: The alarming subsystem must issue an alarm within 25 ms once any abnormal signal has been found in the analysis subsystem.

Without a graphical representation, understanding the relationship among the specifications is very time-consuming and error prone, not to mention verifying their schedulability against the timing requirements and timing assertions. In the following, we can see how TCPN's are useful in visualizing and analyzing such specifications. Using TCPN's to verify a schedulable transition is very straightforward; it simply follows the theorems derived in this paper. What follows is how we verify the four timing assertions. We first model a Petri net N_p based on the functional specifications, then we simulate a reachable marking using the reachability analysis of PN's. Second, we impose the timing specifications into N_p to form a timing constraint Petri net N_s . The timing assertions are also added to N_s to verify the timing assertions. The verification is performed by determining if the desired final markings of N_s are reachable in TCPN's. That is, to analyze whether the last transition whose firing leads to the final marking is schedulable with respect to the initial marking.

Since we are interested in the timing behavior between two transitions, the reachability of N_s can be reduced to the places and transitions between the two transitions. We also set the default values of **place time pairs or transition time pairs** as $(0, \infty)$ and the default value of a **transition duration** as (0) . Abbreviations used in Fig. 4 are summarized in Table I and II.

For TA1, we let M_{10} be the marking after the firing of transition **sam** which is (ISR, P1R, P2R, P3R, TA1) and M_{1n} be the final marking after transition **ssg** which is (SIG). There are four firing sequences for a reachable M_{1n} from M_{10} in

TABLE I
ABBREVIATIONS OF PLACES DENOTED IN FIG. 4

Places	Places
ADD: Abnormal Display is Done	RDI: Ready to Display
AKR: Acknowledgement is Ready	RDS: Ready to Display Signal
AMR: Alarm subsystem Ready	RNA: Ready for Next Alarm
ASR: Analysis subsystem Ready	RRM: Ready to Receive Message
FAN: Found AbNormal	RTS: Ready to detect calculated Signal
ISR: Incoming Signal is Ready for processing	SGR: sampled Signal Ready to be sent
MIR: Message Is Received	SIG: sampled incoming Signal
MGR: Message is Ready	SMD: Sending Msg to alarm subsystem is Done
NDD: Normal Display is Done	SMP: Sampling subsystem's sampling period
NSR: Normal Signal display is Ready	SMR: Sampling subsystem Ready
P1D: processing in Processor 1 is Done	TA1: Timing Assertion 1
P1R: Processor 1 Ready	TA2: Timing Assertion 2
P2D: processing in Processor 2 is Done	TA3: Timing Assertion 3
P2R: Processor 2 Ready	TA4: Timing Assertion 4
P3D: processing in Processor 3 is Done	W4A: Waiting for Acknowledgement
P3R: Processor 3 Ready	
P4R: Processor 4 Ready	
RAM: Ready to Alarm	

TABLE II
ABBREVIATIONS OF TRANSITIONS DENOTED IN FIG. 4

Transitions
<i>1am</i> : 1st alarm
<i>cal</i> : calculating incoming signals
<i>cam</i> : continue to alarm
<i>dad</i> : set display abnormal data done
<i>dan</i> : detecting abnormal
<i>das</i> : displaying abnormal signal
<i>dib</i> : displaying background
<i>dnd</i> : set display normal data done
<i>dns</i> : displaying normal signal
<i>p1p</i> : processor 1 is processing
<i>p2p</i> : processor 2 is processing
<i>p3p</i> : processor 3 is processing
<i>pns</i> : prepare normal signal display
<i>rak</i> : sender receiving acknowledgment from receiver
<i>rec</i> : receiving message
<i>rtr</i> : set ready to receive
<i>rts</i> : set ready to sample
<i>rss</i> : receiving signals
<i>s1r</i> : set processor 1 ready
<i>s2r</i> : set processor 2 ready
<i>s3r</i> : set processor 3 ready
<i>sak</i> : receiver sending acknowledgment to sender
<i>sam</i> : sampling subsystem is sampling signals
<i>sen</i> : sending message
<i>ssg</i> : sending signal to analysis subsystem
<i>syn</i> : synchronizing three subsystems
<i>toa</i> : turn off alarm

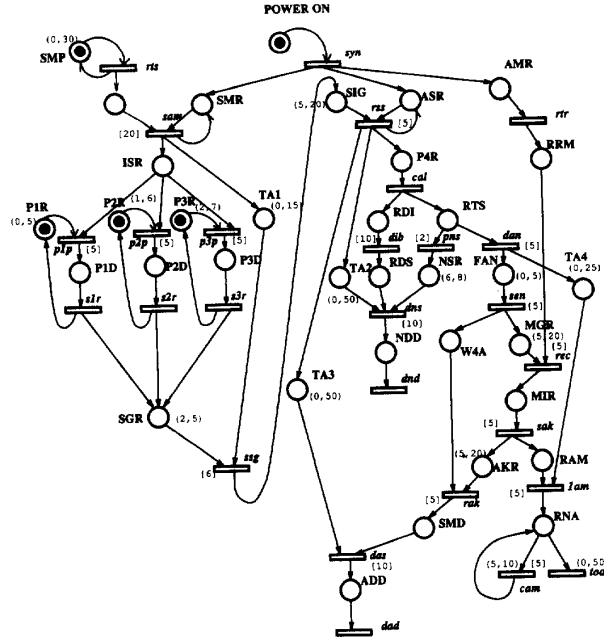


Fig. 4. A TCPN of a patient monitoring system.

this case:

$$\begin{aligned} \delta_1(M_{1n}) & \text{ is } (ISRp1pP1Ds1rSGRssg) \\ \delta_2(M_{1n}) & \text{ is } (ISRp2pP2Ds2rSGRssg) \\ \delta_3(M_{1n}) & \text{ is } (ISRp3pP3Ds3rSGRssg) \\ \delta_4(M_{1n}) & \text{ is } (TA1ssg) \end{aligned}$$

From Theorem 2, we have

$$\begin{aligned} \text{EFBT}(ssg) &= \text{Max}\{[\text{FIRE}_{\text{end}}(sam) \\ &+ (\text{TC}_{\text{min}}(ISR) + \text{TC}_{\text{min}}(p1p) \\ &+ \text{TC}_{\text{min}}(P1D) + \text{TC}_{\text{min}}(s1r) \\ &+ \text{TC}_{\text{min}}(SGR) + \text{TC}_{\text{min}}(ssg) \\ &+ \text{FIRE}_{\text{dur}}(p1p) + \text{FIRE}_{\text{dur}}(s1r)], \\ &[\text{FIRE}_{\text{end}}(sam) \\ &+ (\text{TC}_{\text{min}}(ISR) + \text{TC}_{\text{min}}(p2p) \\ &+ \text{TC}_{\text{min}}(P2D) + \text{TC}_{\text{min}}(s2r) \\ &+ \text{TC}_{\text{min}}(SGR) + \text{TC}_{\text{min}}(ssg) \\ &+ \text{FIRE}_{\text{dur}}(p2p) + \text{FIRE}_{\text{dur}}(s2r)], \\ &[\text{FIRE}_{\text{end}}(sam) \\ &+ (\text{TC}_{\text{min}}(ISR) + \text{TC}_{\text{min}}(p3p) + \text{TC}_{\text{min}}(P3D) \\ &+ \text{TC}_{\text{min}}(s3r) + \text{TC}_{\text{min}}(SGR) + \text{TC}_{\text{min}}(ssg) \\ &+ \text{FIRE}_{\text{dur}}(p3p) + \text{FIRE}_{\text{dur}}(s3r)], \\ &[\text{FIRE}_{\text{end}}(sam) + \text{TC}_{\text{min}}(ssg) + \text{TC}_{\text{min}}(TA1)]] \\ &= \text{Max}\{[\text{FIRE}_{\text{end}}(sam) + 0 + 0 + 0 + 0 \\ &+ 2 + 0 + 5 + 5], \\ &[\text{FIRE}_{\text{end}}(sam) + 0 + 1 + 0 + 0 + 2 + 0 + 5 + 5], \\ &[\text{FIRE}_{\text{end}}(sam) + 0 + 2 + 0 + 0 + 2 + 0 + 5 + 5], \end{aligned}$$

$$\begin{aligned} &[\text{FIRE}_{\text{end}}(sam) + 0 + 0]] \\ &= \text{Max}\{\text{FIRE}_{\text{end}}(sam) + 13, \text{FIRE}_{\text{end}}(sam) + 14, \\ &\quad \text{FIRE}_{\text{end}}(sam) + 15, \text{FIRE}_{\text{end}}(sam)\} \\ &\text{FIRE}_{\text{end}}(sam) + 15 \end{aligned}$$

$$\begin{aligned} \text{LFET}(ssg) &= \text{Min}\{[\text{FIRE}_{\text{end}}(sam) \\ &+ (\text{TC}_{\text{max}}(ISR) + \text{TC}_{\text{max}}(p1p) + \text{TC}_{\text{max}}(P1D) \\ &+ \text{TC}_{\text{max}}(s1r) + \text{TC}_{\text{max}}(SGR))] \\ &[\text{FIRE}_{\text{end}}(sam) + (\text{TC}_{\text{max}}(ISR) + \text{TC}_{\text{max}}(p2p) \\ &+ \text{TC}_{\text{max}}(P2D) + \text{TC}_{\text{max}}(s2r) + \text{TC}_{\text{max}}(SGR)], \\ &[\text{FIRE}_{\text{end}}(sam) + (\text{TC}_{\text{max}}(ISR) + \text{TC}_{\text{max}}(p3p) \\ &+ \text{TC}_{\text{max}}(P3D) + \text{TC}_{\text{max}}(s3r) + \text{TC}_{\text{max}}(SGR)], \\ &[\text{FIRE}_{\text{end}}(sam) + (\text{TC}_{\text{max}}(TA1))] \\ &= \text{Min}\{[\text{FIRE}_{\text{end}}(sam) + \infty + 5 + \infty + \infty + 5], \\ &[\text{FIRE}_{\text{end}}(sam) + \infty + 6 + \infty + \infty + 5], \\ &[\text{FIRE}_{\text{end}}(sam) + \infty + 7 + \infty + \infty + 5], \\ &[\text{FIRE}_{\text{end}}(sam) + 20]\} \\ &= \text{Min}\{\text{FIRE}_{\text{end}}(sam) + \infty, \text{FIRE}_{\text{end}}(sam) + \infty, \\ &\quad \text{FIRE}_{\text{end}}(sam) + \infty, \text{FIRE}_{\text{end}}(sam) + 20\} \\ &= \text{FIRE}_{\text{end}}(sam) + 20 \end{aligned}$$

$\text{FIRE}_{\text{dur}}(ssg) = 6$, From Theorem 2, we have,

$$\begin{aligned} \text{LFET}(ssg) - \text{EFBT}(ssg) &= [\text{FIRE}_{\text{end}}(sam) + 20] \\ &- [\text{FIRE}_{\text{end}}(sam) + 15] = 5 < \text{FIRE}_{\text{dur}}(ssg). \end{aligned}$$

Therefore, M_{1n} is not reachable from M_{10} in TCPN's even though it is reachable in PN's, and transition ssg is not schedulable with respect to M_{10} . From Theorem 6, we can find that the bottleneck is due to the firing sequence, $\delta_3(M_{1n})$, (ISR p3p P3D s3r SGR).

A similar method is used to verify the reachability of M_{2n} , M_{3n} , and M_{4n} . We only show the firing sequence of each marking with respect to their own initial marking and the final analysis results. Detailed computations are similar to the one used to verify TA1.

For TA2, we let M_{20} be the marking after the firing of transition rss which is (P4R TA2 TA3) and M_{2n} be the final marking after transition dns which is (NDD). There are three firing sequences for a reachable M_{2n} from M_{20} in this case:

$$\begin{aligned} \delta_1(M_{2n}) & \text{ is } (P4RcalRDIdibRDSdns) \\ \delta_2(M_{2n}) & \text{ is } (P4RcalRTS PN's NSRdns) \\ \delta_3(M_{2n}) & \text{ is } (TA2dns) \end{aligned}$$

$$\begin{aligned} \text{EFBT}(dns) &= \text{FIRE}_{\text{end}}(rss) + 32 \\ \text{LFET}(dns) &= \text{FIRE}_{\text{end}}(rss) + 50 \end{aligned}$$

$\text{FIRE}_{\text{dur}}(dns) = 10$. From Theorem 2 we have,

$$\begin{aligned} \text{LFET}(dns) - \text{EFBT}(dns) &= [\text{FIRE}_{\text{end}}(rss) + 50] \\ &- [\text{FIRE}_{\text{end}}(rss) + 32] = 18 > \text{FIRE}_{\text{dur}}(dns). \end{aligned}$$

Therefore, M_{2n} is reachable in TCPN's from M_{20} , and transition dns is schedulable with respect to M_{20} .

For TA3, we let M_{30} be the marking after the firing of transition rss which is (P4R TA2 TA3) and M_{3n} be the final marking after transition das which is (ADD). There are four firing sequences for a reachable M_{3n} from M_{30} in this case:

$$\begin{aligned}\delta_1(M_{3n}) & \text{ is (P4Rca} \overline{\text{RD}} \overline{\text{dib}} \\ & \quad \text{RDS} \overline{\text{das}}) \\ \delta_2(M_{3n}) & \text{ is (P4Rca} \overline{\text{RTS}} \overline{\text{dan}} \\ & \quad \text{FAN} \overline{\text{sen}} \overline{\text{W4ArakSMD}} \overline{\text{dan}}) \\ \delta_3(M_{3n}) & \text{ is (P4Rca} \overline{\text{RTS}} \overline{\text{dan}} \\ & \quad \text{FAN} \overline{\text{sen}} \overline{\text{MGR}} \overline{\text{rec}} \overline{\text{MIR}} \overline{\text{sak}} \\ & \quad \overline{\text{AKR}} \overline{\text{raksMD}} \overline{\text{das}}) \\ \delta_4(M_{3n}) & \text{ is (TA3} \overline{\text{das}})\end{aligned}$$

$$\text{EFBT}(\overline{\text{das}}) = \text{FIRE}_{\text{end}}(\overline{\text{rss}}) + 55$$

$$\text{LFET}(\overline{\text{das}}) = \text{FIRE}_{\text{end}}(\overline{\text{rss}}) + 50$$

$\text{FIRE}_{\text{dur}}(\overline{\text{das}}) = 10$. From Theorem 1, we have,

$$\begin{aligned}\text{LFET}(\overline{\text{das}}) - \text{EFBT}(\overline{\text{das}}) &= [\text{FIRE}_{\text{end}}(\overline{\text{rss}}) + 50] \\ &- [\text{FIRE}_{\text{end}}(\overline{\text{rss}}) + 55] = -5 < 0.\end{aligned}$$

Therefore, M_{3n} is not reachable from M_{30} in TCPN's even though it is reachable in PN's, and transition das is not schedulable with respect to M_{30} .

For TA4, we let M_{40} be the marking after the firing of transition dan which is (FAN TA4) and M_{4n} be the final marking after transition $1am$ which is (RNA). There are two firing sequences for a reachable M_{4n} from M_{40} in this case:

$$\begin{aligned}\delta_1(M_{4n}) & \text{ is (FAN} \overline{\text{sen}} \overline{\text{MGR}} \overline{\text{rec}} \overline{\text{MIR}} \overline{\text{sak}} \overline{\text{RAM}} \overline{\text{1am}}) \\ \delta_2(M_{4n}) & \text{ is (TA4} \overline{\text{1am}})\end{aligned}$$

$$\text{EFBT}(\overline{\text{1am}}) = \text{FIRE}_{\text{end}}(\overline{\text{dan}}) + 20$$

$$\text{LFET}(\overline{\text{1am}}) = \text{FIRE}_{\text{end}}(\overline{\text{dan}}) + 25$$

$\text{FIRE}_{\text{dur}}(\overline{\text{1am}}) = 5$. From Theorem 2, we have,

$$\begin{aligned}\text{LFET}(\overline{\text{1am}}) - \text{EFBT}(\overline{\text{1am}}) &= [\text{FIRE}_{\text{end}}(\overline{\text{dan}}) + 25] \\ &- [\text{FIRE}_{\text{end}}(\overline{\text{dan}}) + 20] = 5.\end{aligned}$$

Therefore, M_{4n} is reachable in TCPN's from M_{40} , and transition $1am$ is schedulable with respect to M_{40} .

ACKNOWLEDGMENT

The authors are grateful to R. Smith, A. Liu, and T. Murata for their constructive comments on earlier versions of this manuscript. The authors also appreciate the anonymous reviewers' comments which have helped us to clarify many technical points and greatly enhanced the readability of this manuscript.

REFERENCES

- [1] B. Berthomieu and M. Diaz, "Modeling and verification of time dependent systems using time Petri nets," *IEEE Trans. Software Eng.*, vol. SE-17, pp. 259-273, Mar. 1991.
- [2] J. E. Coolahan, Jr. and N. Roussopoulos, "Timing requirements for time driven systems using augmented Petri nets," *IEEE Trans. Software Eng.*, vol. SE-9, pp. 603-616, Sept. 1983.
- [3] B. Dasarathy, "Timing constraints of real-time systems: constructs for expressing them, methods of validating them," *IEEE Trans. Software Eng.*, vol. SE-11, pp. 80-86, Jan. 1985.
- [4] M. Felder, D. Mandrioli, and A. Morzenti, "Proving properties of real-time systems through logical specifications and Petri net models," *IEEE Trans. Software Eng.*, vol. SE-20, pp. 127-141, Feb. 1994.
- [5] C. Ghezzi, D. Mandrioli, S. Morasca, and M. Pezze, "A unified high-level Petri net formalism for time-critical systems," *IEEE Trans. Software Eng.*, vol. SE-17, pp. 160-171, Feb. 1991.
- [6] D. Haban and K. G. Shin, "Application of real-time monitoring to scheduling tasks with random execution times," *IEEE Trans. Software Eng.*, vol. SE-16, pp. 1374-1389, Dec. 1990.
- [7] M. A. Holliday and M. K. Vernon, "A generalized timed Petri net model for performance analysis," *IEEE Trans. Software Eng.*, vol. SE-13, pp. 1297-1310, Dec. 1987.
- [8] F. Jahanian and A. K.-L. Mok, "Safety analysis of timing properties in real-time systems," *IEEE Trans. Software Eng.*, vol. SE-12, pp. 890-904, Sept. 1986.
- [9] ———, "A graph-theoretic approach for timing analysis and its implementation," *IEEE Trans. Comput.*, vol. C-36, pp. 961-975, Aug. 1987.
- [10] N. G. Leveson and J. L. Stolzy, "Safety analysis using Petri nets," *IEEE Trans. Software Eng.*, vol. SE-13, no. 3, pp. 386-397, Mar. 1987.
- [11] J. Li, I. Suzuki and M. Yamashita, "A new structural induction theorem for rings of temporal Petri nets," *IEEE Trans. on Software Engin.*, vol. SE-20, no. 2, pp. 115-126, Feb. 1994.
- [12] N. Lopez-Benitez, "Dependability modeling and analysis of distributed programs," *IEEE Trans. Software Eng.*, vol. SE-20, pp. 345-352, May 1994.
- [13] P. M. Merlin and D. J. Farber, "Recoverability of communication protocols implications of a theoretical study," *IEEE Trans. Commun.*, vol. COM-24, pp. 1036-1043, Sept. 1976.
- [14] T. Murata, "Petri nets: properties, analysis and application," in *Proc. IEEE*, vol. 77, 1989, pp. 541-580.
- [15] M. Notomi and T. Murata, "Hierarchical reachability graph of bounded Petri nets for concurrent-software analysis," *IEEE Trans. Software Eng.*, vol. SE-20, pp. 325-336, May 1994.
- [16] D. Peng and K. G. Shin, "Modeling of concurrent task execution in a distributed system for real-time control," *IEEE Trans. Comput.*, vol. C-36, pp. 500-516, Apr. 1987.
- [17] C. V. Ramamoorthy and G. S. Ho, "Performance evaluation of asynchronous concurrent systems using Petri nets," *IEEE Trans. Software Eng.*, vol. SE-6, pp. 440-449, Sept. 1980.
- [18] C. Ramchandani, "Analysis of asynchronous concurrent systems by Petri nets." Cambridge, MA: MIT, Project MAC, TR-120, Feb. 1974.
- [19] R. R. Razouk and C. V. Phelps, "Performance analysis using time Petri nets," in *Proc. 4th IFIP Protocol Specification, Testing and Verification*, Y. Yemini et al., Eds. Amsterdam, The Netherlands: North-Holland, 1985.
- [20] J. A. Stankovic, K. Ramamritham, and S. Cheng, "Evaluation of a flexible task scheduling algorithm for distributed hard real-time systems," *IEEE Trans. Comput.*, vol. 34, pp. 1130-1143, Dec. 1985.
- [21] A. D. Stoyenko, C. Hamacher, and R. C. Holt, "Analyzing hard-real-time programs for guaranteed schedulability," *IEEE Trans. Software Eng.*, vol. SE-17, pp. 737-750, Aug. 1991.
- [22] I. Suzuki and H. Lu, "Temporal Petri nets and their application to modeling and analysis of a handshake daisy chain arbiter," *IEEE Trans. Comput.*, vol. C-38, pp. 696-704, May 1989.
- [23] H. Tokuda and M. Kotera, "Scheduler 1-2-3: an interactive schedulability analyzer for real-time systems," in *Proc. of IEEE 12th Int. Comput. Software and Applic. Conf.*, Chicago, IL, Oct. 1988, pp. 211-219.
- [24] J. J. P. Tsai, K. Y. Fang, H. Y. Chen, and Y. D. Bi, "A non-interference monitoring and replay mechanism for real-time software testing and debugging," *IEEE Trans. Software Eng.*, vol. SE-16, pp. 897-916, Aug. 1990.
- [25] J. J. P. Tsai, K. Y. Fang, and H. Y. Chen, "A non-invasive architecture to monitoring real-time distributed systems," *IEEE Comput.*, vol. 23, pp. 11-23, Mar. 1990.
- [26] J. J. P. Tsai and T. J. Weigert, *Knowledge-Based Software Development For Real-Time Distributed Systems*. Singapore: World Scientific Publication, 1993.

- [27] J. J. P. Tsai, T. Weigert, and H.C. Jang "A hybrid knowledge representation as a basis of requirement specification and specification analysis," *IEEE Trans. Software Eng.*, vol. SE-18, pp. 1076-1100, Dec. 1992.
- [28] J. J. P. Tsai and S. J. H. Yang, *Monitoring and Debugging of Distributed Real-Time Systems*. Washington, DC: IEEE Computer Society Press, 1995.

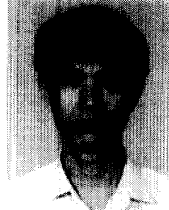


Jeffrey J. P. Tsai (S'82-M'85-SM'91) received the Ph.D. degree in computer science from Northwestern University, Evanston, Illinois.

He is an Associate Professor in the Department of Electrical Engineering and Computer Science at the University of Illinois at Chicago where he is also directing the Distributed Real-Time Intelligent Systems Laboratory. His major current research interests include knowledge-based software engineering, distributed real-time systems, safety-critical systems, multiagents architecture and CSCW. He

co-authored the book, *Knowledge-Based Software Development for Real-Time Distributed Systems* (World Scientific, 1993), co-edited the book, *Monitoring and Debugging of Distributed Real-Time Systems* (IEEE Computer Society Press, 1995), and has published over 100 articles in refereed journals and conference proceedings.

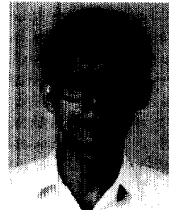
Dr. Tsai received an Engineering Foundation Society Research Award from the IEEE and the Engineering Foundation Society in 1988, a University Scholar Award from the President of the University of Illinois in 1994, an IEEE Meritorious Service Award from the IEEE Computer Society in 1994, and is currently an IEEE Distinguished Visitor. He also serves as the Co-Editor-in-Chief of the *International Journal of Artificial Intelligence Tools*, a Guest Editor of the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, special issue on Dependability of A.I. Systems, and an Editor of the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, the *International Journal of Software Engineering and Knowledge Engineering*, and the *International Journal of Systems Integration*. He was a Program Co-Chair of the 13th IEEE Computer Software and Applications Conference, the Program Chair of the 5th IEEE Tools with Artificial Intelligence Conference, the Tutorial Chair of the 1st IEEE International Conference on Requirements Engineering, and the General Chair of the 6th IEEE International Conference on Tools with Artificial Intelligence. He is a member of the Association for Computing Machinery, the American Association of Artificial Intelligence, and Upsilon Pi Upsilon.



Steve Jennhwa Yang (S'94) received the B.E. degree in computer science from the Tamkang University, Taiwan in 1985, and the M.S. degree in electrical engineering and computer science from the University of Illinois at Chicago in 1993.

He is now working as a research assistant in the Distributed Real-Time Intelligent Systems Laboratory, and expects to receive the Ph.D. degree in electrical engineering and computer science from the University of Illinois at Chicago in 1995. He is the co-editor of the book, *Monitoring and Debugging of Distributed Real-Time Systems* (IEEE Computer Society Press, 1995). His current research interests include intelligent systems, Petri nets modeling and analysis of safety-critical systems, real-time instrumentation and debugging, and formal methods for verifying and synthesizing distributed real-time systems.

Mr. Yang is a student member of the Association for Computing Machinery.



Yao-Hsiung Chang received the B.S. degree in mechanical engineering from the National Taiwan University, Taiwan in 1988, and the M.S. degree in mechanical engineering from The University of Michigan, Ann Arbor in 1991, and the M.S. degree in electrical engineering and computer science from the University of Illinois at Chicago in 1993.

He is currently a software engineer at LEADWELL CNC machines manufacturing corporation, Taiwan. His research interests include real-time flexible manufacturing systems and PC-based CNC controllers.