Information Technology Journal, 2013
ISSN 1812-5638 / DOI: 10.3923/itj.2013.

Modeling and Analysis of IoT Real-Time System Using TCPN

Huaizhou Yang and Shaowei Pan
College of Computer Science, Xi'an Shiyou University, Xi'an 710065, China

Abstract: The time-sensibility is an important feature of Internet of Things (IoT). For the purpose of accurately modeling the IoT real-time systems with complex timing constraints and eliminating timing constraint conflicts, in this study, Timing Constraint Petri Net (TCPN) is chosen as an unitized modeling method of timing constraints and system orchestration process. A method of temporal behavior analysis and verification is presented to find and eliminate timing constraint conflicts from local and global viewpoints, respectively. Some important analysis and verification formulas are presented and proved. Via model analysis and verification, the period of validity of each system activity can be obtained, which facilitates the system monitoring and management. By an example, the effectiveness of the proposed method is illustrated. The proposed method can help to guarantee the timely and successful response of each system activity and furthermore, it can help to guarantee the stability and reliability of IoT real-time system.

Key words: Internet of things, real-time system, timing constraint Petri net, timing analysis

INTRODUCTION

In recent years, the technology of Internet of Things (IoT) has attracted highly attention of academia. IoT promotes the further developments of the existing technologies of sensing, computing and communication. Especially, IoT focuses on the real-time and intelligent interactions among humans, computers and things (Guo *et al*., 2012). Unlike the traditional Internet and sensor networks, most IoT systems are real-time or time-sensitive systems (Shen *et al*., 2010). For example, in a hospital IoT system, it is required that the monitor data of a patient in intensive care must be analyzed and processed on time and without any error. High stability and reliability become the basic characteristics of IoT. However, in an IoT real-time system, it is a difficult work to guarantee the system's stability and reliability, mostly due to the complex time restrictions on target events and system activities. These time restrictions are called system timing constraints. For example, an event is valid only if it happens within or after a time limit and a system activity is valid only if it starts and finished within or after a time limit. Except the whole response time of system needs to be satisfied, every process time of events and system activities must satisfy their timing constraints, respectively. Any violation of timing constraints will destroy the stability and reliability of IoT real-time system. Because there exist the time sequences and logic dependent relationships between system events and activities, timing constraint conflicts often happen after imposing a serials of timing constraints to an IoT real-time system. Initial timing constraints imposed to an IoT real-time system usually are referred to as relative time limits, which only consider the local and individual timing requirements. When checking and verifying the correctness of the whole system timing constraints from a global viewpoint, timing constraint conflicts could be found. We need to consider the actual starting time and duration of every system event and activity according to their time sequences and logic dependent relationships, as well as transform the relative time limits to the absolute time limits. Therefore, timing constraint modeling and analysis of IoT real-time system become very important, which can facilitate the verification of timing constraints and the temporal management of large-scale IoT applications.

How to express the timing constraints is an important problem in the research of IoT real-time system modeling. In the work of Li *et al*. (2009), the period of time that an RFID (Radio Frequency Identification) event can legally live in a system is divided into different categories for controlling the time restrictions. Furthermore, they present a special data structure, which is called double level sequence list, to record the intermediate stages of events and to process the unordered event streams. In the work of Li *et al*. (2011), an IoT services modeling approach is proposed based on timed automata. The timing constraints of system activities are reflected as some clock variables. An activity can be executed only when an expression of clock variables is satisfied. These works have made a good job in the expression of timing constraints. However, these solutions remain hard to be

Corresponding Author: Huaizhou Yang, College of Computer Science, Xi'an Shiyou University, Xi'an 710065, China

used to reflect the complex timing constraints of the IoT real-time system. A main reason is the timing constraints of the IoT real-time system include not only the lifecycles of system events and the executable periods of system activities but also the communication delay time, the system decision-making time and the actual duration of activities. Any missing of timing constraints will produce the untruthfulness of system model. In addition, multiple timing constraints could be imposed to a single activity. How to express timing constraints in this situation is not be well solved in the works above mentioned.

The QoS support of IoT system is still an open issue that needs to be researched to meet the requirements of IoT scenarios (Atzori *et al*., 2010). The analysis and verification of temporal characteristics are important steps to guarantee the QoS of IoT system. In the work of Funk *et al*. (2009), an analysis framework of IoT system is presented. In this framework, the data collection mechanisms can be dynamically refined and the collected data can be analyzed. Unfortunately, their work stop at the phase of conceptual design and the concrete system analysis methods still need to be researched. Some works concentrate on using the available time information of data to construct intelligent behaviors of system (Kortuem *et al*., 2010; Munoz-Organero *et al*., 2010). However, only simple temporal relations are analyzed in these works. How to verify the validity of intelligent behaviors with timing constraints is still not clear. Many technologies and methods, such as smart workflow (Giner *et al*., 2010) and contextual reasoning (Garcia-Macias *et al*., 2011), are researched to construct a self-adaptive IoT system. However, due to the lack of effectual temporal modeling method, the universality of these research results is restricted.

Being different from the related works, we focus on the modeling and analysis of IoT real-time system with complex timing constraints. The various timing constraints and the possible system orchestration processes are synthetically reflected by Timing Constraint Petri Net (TCPN) in this study. Based on the presented modeling method, we can conveniently transform the temporal context in real world to the corresponding timing constraints and impose them to the IoT real-time system. In order to guarantee the stability and reliability of system, the analysis methods of timing constraint information are described in detail in this study. In contrast with classical TCPN, more strict verification conditions of timing constraints are defined. Through modeling and analysis of timing constraint information, we can find the possible timing constraint conflicts and then adjust and eliminate them. In addition, the concrete activities which produce timing constraint conflicts can be identified and the effectiveness of system can be predicted. The method presented in this study has high universality, which can help construct a robust IoT real-time system.

The main contribution of this study is twofold: (1) An unitized modeling method of complex timing constraints of IoT real-time system and system orchestration process and (2) An analysis and verification method of the timing constraints.

THE TIMING CONSTRAINT SPECIFICATION AND MODELING OF IOT REAL-TIME SYSTEM BASED ON TCPN

Here, first, the categories and basic relationships of timing constraints in IoT real-time system are discussed in detail. Then a modeling method of IoT real-time system is presented based on TCPN. Finally, a modeling example is described.

The categories and basic relationships of timing constraints: In a typical IoT real-time system, most system processes exist in temporal context and have timing constraints. After carefully analyzing various IoT real-time system, we divide timing constraints into three categories, which are defined as below.

Definition 1: Timing constraints to enable activities (tC_{enable}): A tC_{enable} is a time interval, during which a condition to enable an activity can be satisfied and sustained. A tC_{enable} is a time pair denoted as (tc_{enable}^{min} , tc_{enable}^{max}). A condition becomes true and may be used to enable an activity only between tc_{enable}^{min} to tc_{enable}^{max} after a task starts up.

The precondition that an activity can be enabled usually is the required resources have been acquired. In here, the resources have extensive contents, which include data, communication link, system memory and so on. After a task starts up, it needs some delay time to acquire these resources. For example, in a RFID application system, in order to complete a task, the RFID readers need to read the RFID tags and transfer the RFID data to the system for further processing. These actions will produce a delay time before the activity of data processing is enabled. Therefore, in a tC_{enable}, if a task starts up at time t_0, then tc_{enable}^{min} means the activity can be enabled only after a period of delay time from t_0 to $t_0 + tc_{enable}^{min}$.

In addition, because every activity has a duration of execution, an activity can be successfully finished only when its enabling condition is satisfied and sustained no change in its execution duration. In another word, the acquired resources of the activity must be held on and

keep valid in the execution duration. The IoT real-time systems are often constructed in a dynamic and distributed environment. Most resources have their own temporal features. For example, the blood-pressure information of a patient in intensive care has a period of validity. If this information are not processed in time, they become invalid and must be acquired again for further intelligent analysis. Therefore, in a tC_{enable}, if a task starts up at time t_0, then tc_{enable}^{max} means the activity must be finished before the time $t_0 + tc_{enable}^{max}$.

Definition 2: Timing constraints of executable activities ($tC_{executable}$): The $tC_{executable}$ is the period of time that an activity can be executed after it is enabled. A $tC_{executable}$ is a time pair denoted as ($tc_{executable}^{min}$, $tc_{executable}^{max}$). An activity can be executed only between $tc_{executable}^{min}$ to $tc_{executable}^{max}$ after it is enabled.

When all enabling conditions are satisfied, an activity is said to be enabled. An activity, which is enabled at time t_0, is said to be executable during the time period from $t_0 + tc_{executable}^{min}$ to $t_0 + tc_{executable}^{max}$. After an activity is enabled, the system needs some time to verify the data or accomplish the data transformation. Therefore, the delay time from t_0 to $t_0 + tc_{executable}^{min}$ can not be ignored in an IoT real-time system. In addition, in order to guarantee the execution result can be acquired on time, $tc_{executable}^{max}$, as a execution deadline, is usually imposed to an activity.

Definition 3: Timing constraints on execution duration of activity ($tC_{duration}$): The $tC_{duration}$ is the execution time limit of an activity. It represents the amount of time an activity takes to perform its target task.

A $tC_{duration}$ can be denoted as [$tC_{duration}$]. For example, [5] means an activity needs to spend 5 time units from its beginning to end. In contrast with tC_{enable} and $tC_{excutable}$, a $tC_{duration}$ only includes a single time variable. The actual beginning time of an activity is dynamically determined at run-time.

The categories and basic relationships of timing constraints in IoT real-time system are illustrated by Fig. 1. In three categories of timing constraints, tC_{enable} needs to be emphasized. tC_{enable} is associated with enabling conditions of activities. It is important to note that multiple enabling conditions are maybe imposed to a single activity. Therefore, there maybe exist multiple tC_{enable} to an activity. In this situation, the enabling period is determined by all tC_{enable} imposed to the activity. In Fig. 1, as an example, two tC_{enable} are imposed to an activity. The enabling period of this activity can be determined by the expression:

$$\mathrm{Max}(t_{01} + tc1_{enable}^{min}, t_{02} + tc2_{enable}^{min}), \mathrm{Min}(t_{01} + tc1_{enable}^{max}, t_{02} + tc2_{enable}^{max})$$

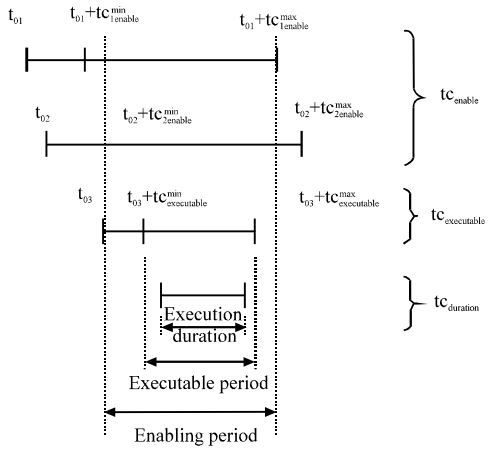

Fig. 1: The categories and basic relationships of timing constraints

in which t_{01} is the task startup time of $tC_{1_{enable}}$ and t_{02} is the task startup time of $tc_{2_{enable}}$. In Fig. 1, we suppose $t_{01} + tc1_{enable}^{min} < t_{02} + tc2_{enable}^{min}$ and $t_{01} + tc1_{enable}^{max} < t_{02} + tc2_{enable}^{max}$, then the enabling period of this activity is $(t_{02} + tc2_{enable}^{min}, t_{01} + tc1_{enable}^{max})$. Being different from the tc_{enable}, there is only one $tc_{excutable}$ imposed to an activity. In Fig. 1, t_{03} is the enabling startup time. In here, $t_{03} = t_{02} + tc2_{enable}^{min}$. In order to guarantee the activity is executed successfully, the execution duration must less than or equal to the executable period, i.e.:

$$tc_{duration} \le tc_{executable}^{max} - tc_{executable}^{min}$$

From above discussion, we can know the time variables in tc_{enable} and $tc_{excutable}$ are relative time variables without considering the actual startup time. There exist many activities and their timing constraints in an IoT real-time system. The execution sequence of activities will affect the actual startup time of the timing constraints. Therefore, we need to consider the initial startup time of the system, transform the relative time variables to absolute time variables and then verify the timing constraints from a global viewpoint. The detailed global analysis of timing constraints will be presented in subsequent content of this study.

The modeling of timing constraints and system processes based on TCPN: Petri nets are mainly used for modeling and analyzing the discrete systems with asynchronous, concurrent and non-deterministic features. The Timing Constraint Petri Net (TCPN) extends the basic Petri nets by associating places and transitions with timing constraints. TCPN has been proved suitable for modeling the systems with conflict structures. Once an IoT real-time system is modeled by a TCPN, we can detect the system

defects by formal verification and eliminate these problems. Especially, via verification of timing constraints, we can guarantee the IoT real-time systems meet the desired temporal demands.

Definition 4: Timing constraint Petri net (Tsai *et al*., 1995) A TCPN is a 6-tuple (P, T, F, C, D, M) where:

- P is a finite set of places, i.e., P = $\{p_1, p_2, \ldots, p_m\}$
- T is a finite set of transitions, i.e., T = $\{t_1, t_2, \ldots, t_n\}$
- $F \subseteq (P \times T) \cup (T \times P)$ is finite set of arcs which connect 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, $[FIRE_{dur}(pt_j)]$, where pt_j is either a place or a transition
- M is a set of marking with m-vector, $(M_{(p1)}, \ldots, M_{(pj)}, \ldots, M_{(pm)})$, where $M_{(pj)}$ denotes the numbers of token in place p_j. M_0 denotes the initial marking

A TCPN is an unitized model of timing constraints and system orchestration process. According to the above discussed timing constraint categories of IoT real-time system, based on TCPN, an IoT real-time system with complex timing constraints can be accurately described. The enabling conditions of system activities are represented by places. The tasks and resources of system are represented by tokens in places. An IoT real-time system timing constraint tC_{enable} with a time pair (tc^{min}_{enable} , tc^{max}_{enable}) is represented by the time pair $(TC_{min}(p_i), TC_{max}(p_i))$ on a corresponding place p_i in TCPN. The system activities are represented by transitions. An IoT real-time system timing constraint $tC_{executable}$ with a time pair ($tc^{min}_{executable}$, $tc^{max}_{executable}$) on an activity j is represented by the time pair $(TC_{min}(t_j), TC_{max}(t_j))$ on transition t_j in TCPN. An IoT real-time system timing constraint $tC_{duration}$ on an activity j is represented by $[FIRE_{dur}(t_j)]$ on transition t_j in TCPN. The system states are represented by marking set M.

Some important modeling issues need to be explained as follow:

- The firing mode of transitions in TCPN is different from that in basic Petri net. In TCPN, a transition is said to be enabled if there is at least one token in each of its input places as well as each timing constraint on its input places is satisfied. After a transition is enabled, the transition is said to be firable if the timing constraint on it is satisfied. The firing of a transition needs to spend a period of time, during which the enabling tokens are preserved. After a transition completes its firing successfully, the movement of tokens from each input place to each output place of it spends no time
- TCPN follows the weak firing mode. Any enabled transition is not forced to fire immediately. When a transition in a conflict structure has a higher priority than others, this transition is often specified with an earlier deadline. Therefore, TCPN is suitable for modeling and analyzing the conflict structures
- For those places and transitions without explicit timing constraints associated with them, the default values of $(TC_{min}(pt_j), TC_{max}(pt_j))$ is $(0, \infty)$ and the default value of $[FIRE_{dur}(t_j)]$ is [0]

A modeling example of IoT real-time system: The patient monitoring systems are a class of typical IoT real-time systems. As a modeling example, first, the specifications of a patient monitoring system are listed as follow:

- **S1:** The physical information of a patient is sampled every 60 time units (TU). The activity of sampling takes 10 TU. After acquiring monitoring data, the system must response to emergent situations within 50 TU and response to normal situations within 50 TU
- **S2:** The verification and transformation of initial sampling data takes 5 TU. However, this activity must wait for 6 TU after the completion of sampling data due to the data transfer delay time
- **S3:** After the verification and transformation of initial sampling data, the system begins to analyze pulse, blood pressure and cardiogram data respectively in a parallel manner. The analysis of pulse takes 5 TU. The analysis of blood pressure takes 8 TU. The analysis of cardiogram takes 15 TU
- **S4:** After three kinds of individual data analysis, the results of analysis are associated together and a further synthesis analysis is performed. The activity of synthesis analysis takes 6 TU
- **S5:** The synthesis analysis result only can be used within 23 TU in order to avoid data invalidation.
- **S6:** Through contrasting between the result of synthesis analysis and the margin of safety preset for a patient, the system can determine whether an emergent situation occurs or not. Due to the communication delay of querying safety information from a external database, the activity of safety checking must wait for 4 TU and complete within 15 TU after the completion of synthesis analysis. The activity of safety checking takes 3 TU
- **S7:** If safety checking is successful, then the patient log file is updated. The activity of log update takes 5 TU

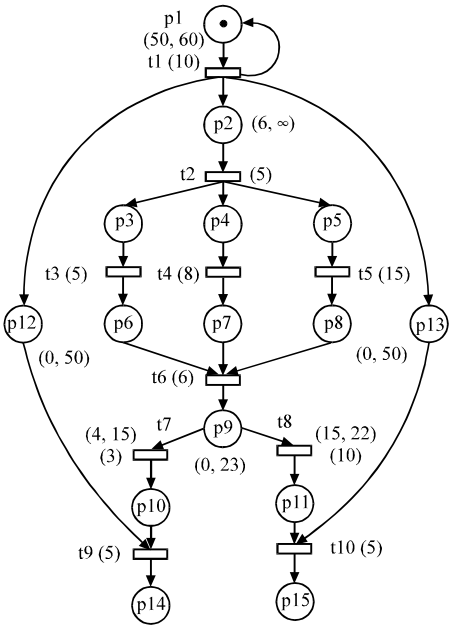

Fig. 2: The TCPN example of a patient monitoring system.

- **S8:** If safety checking is fail, i.e., an emergent situation occurs, then the system produces an alarm. The activity of emergency response must wait for 15 TU and complete within 22 TU after the completion of synthesis analysis. The activity of emergency response takes 10 TU
- **S9:** After the emergency response, it takes 10 TU to record the alarm information

According to the system specifications above described, we can transform them into a TCPN shown in Fig. 2. t_1 represents the activity of data sampling. t_2 represents the activity of data verification and transformation. t_3, t_4 and t_5 represent three parallel activities, i.e., pulse analysis, blood pressure analysis and cardiogram analysis, respectively. t_6 represents the activity of synthesis analysis. t_7 represents the activity of safety checking. If t_7 is fired and completed successfully, then the activity of log update, denoted as t_9, is executed. If t_7 is not completed successfully, then the activity of emergency response, denoted as t_8, is executed. Therefore, t_7 has a higher priority than t_8 by specifying t_7 with an earlier deadline. t_{10} represents the activity of recording the alarm information. The timing constraints in system specifications are imposed on corresponding places and transitions, respectively.

THE TIMING CONSTRAINT ANALYSIS AND VERIFICATION OF IOT REAL-TIME SYSTEM BASED ON TCPN

After modeling an IoT real-time system by TCPN, the timing constraints in the model need to be verified to avoid the timing constraint conflicts. In this section, first, we present a local verification method to find possible simple timing constraint conflicts. Then we present a global verification method to find complex and potential timing constraint conflicts. Different from classical TCPN (Tsai *et al*., 1995) and some related works (Juan *et al*., 2001; Dai *et al*., 2008), more strict and practical verification conditions of timing constraints are analyzed and given.

In order to facilitate the expression and discussion, some abbreviative notations are defined as follow:

Definition 5: $IP(t_j)$ is a set of input places of t_j. $OP(t_j)$ is a set of output places of t_j. $IT(p_j)$ is a set of input transitions of p_j. $OT(p_j)$ is a set of output transitions of p_j. $EEBT(t_j)$ is the earliest enabling beginning time of t_j. $LFET(t_j)$ is the latest enabling ending time of t_j. $EFBT(t_j)$ is the earliest fire beginning time of t_j. $LFET(t_j)$) is the latest fire ending time of t_j. ($EFBT(t_j)$, $LFET(t_j)$)) is the period of validity of t_j. $Tk_{arr}(p_i)$ is the time at which a token arrives at place p_j.

Local timing constraint analysis and verification: As we can see in Fig. 2, the initial timing constraints imposed on a TCPN only include local temporal information, i.e., a timing constraint ($TC_{min}(t_j)$), ($TC_{max}(t_j)$) or [$FIRE_{dur}(t_j)$] only affects transition t_j and a timing constraint ($TC_{min}(p_j)$), ($TC_{max}(p_j)$) only affects the output transition of p_j. The token arrival time of place and the enabling startup time of transition are not considered. However, we can verify these initial timing constraints to find some possible timing constraint conflicts. Two basic verification conditions are listed as follow:

$$TC_{max}(pt_j) \geq TC_{min}(pt_j) \quad (1)$$

$$TC_{max}(t_j)\text{-}TC_{min}(t_j) \geq FIRE_{dur}(t_j) \quad (2)$$

The condition (Eq. 1) assures to form a correct time interval ($TC_{min}(p_j)$), ($TC_{max}(p_j)$). The condition (Eq. 2) assures the executable period of t_j is greater than or equal to the actual execution period of t_j. In addition, we need to guarantee a transition's enabling period is long enough to support its execution, so another verification condition is listed as follow:

$$\forall p_i \in IP(t_j), \exists TC_{max}(p_i) - TC_{min}(p_i) - TC_{min}(t_j) \geq FIRE_{dur}(t_j) \quad (3)$$

We need to prove the necessity and correctness of condition (Eq. 3) by following theorem.

Theorem 1: If $TC_{max}(p_i)\text{-}TC_{min}(p_i)\text{-}TC_{min}(t_j) < FIRE_{dur}(t_j)$ for any input place p_i of transition t_j, then t_j can not complete its firing process successfully.

Proof: A transition t_j can be fired 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. Therefore, it is obvious that the expression $TC_{max}(p_i)$-$TC_{min}(p_i) \geq FIRE_{dur}(t_j)$ must be satisfied for any input place p_i of t_j. After being enabled, t_j must wait a delay time $TC_{min}(t_j)$ before it is fired, so the actual enabling period to guarantee the firing process of t_j is $TC_{max}(p_i)$-$TC_{min}(p_i)$-$TC_{min}(t_j)$ for any input place p_i of t_j. If $TC_{max}(p_i)$-$TC_{min}(p_i)$-$TC_{min}(t_j) < FIRE_{dur}(t_j)$, then the enabling period will be over before the end of firing process of t_j, so the firing process will fail.

Some important problems related to local timing constraint verification need to be explained as follow:

- Even if all three conditions from Eq. 1 to 3 are satisfied, there is no guarantee that the firing process of a transition t_j will complete successfully. For example, in Fig. 3a, local verification condition 1-3 are all satisfied because 10>2, 15>5, 9>1,9-1 = 8>[7], 10-2-1 = 7 = [7] and 15-5-1 = 9>[7]. However, when the token arrival time of p_1 is $T_{01} = 6$ and the token arrival time of p_2 is $T_{02} = 0$, then the absolute timing constraints imposed on p_1 and p_2 are (8,16) and (5,15), respectively. Therefore, the absolute enabling period is (8,15) and the absolute executable period is (9,17). Under the combined constraints coming from the absolute enabling period and the absolute executable period, the period of validity to guarantee the firing process of t_1 is (9,15). Because 15-9 = 6<[7], t_1 can not complete its firing process successfully
- Some classical TCPN analysis methods are not strict enough to be used to find timing constraint conflicts in local verification process. For example, in the research of Tsai *et al.* (1995), when the arrival times of tokens in each input place are not considered, a transition t_j is called a Weakly Schedulable Transition (WST) if and only if:

$$EFBT(t_j) = Max\{TC_{min}(p_j)\} + Tc_{min}(t_j)$$
$$LFET(t_j) = Min\{TC_{max}(p_j)\} + TC_{max}(t_j)$$
$$LFET(t_j) \geq EFBT(t_j) \text{ and } LFET(t_j) - EFBT(t_j) \geq FIRE_{dur}(t_j)$$

where, $p_j \in IP(t_j)$.

However, in fact, even if a transition is not a WST, it can still complete its firing process successfully in some situation. For example, in Fig. 3a, $EFBT(t_1) = Max\{2, 5\}+1 = 6$, $LFET(t_1) = Min\{10, 15, 9\} = 9$ and $LFET(t_1)-EFBT(t_1) = 9-6 = 3<[7]$, so t_1 is not a WST. However, in run time, if the token arrival time of p_1 is $T'_{01} = 4$ and the token arrival time of p_2 is $T_{02} = 0$, then the enabling intervals of p_1 and p_2 are (6,14) and (5,15), respectively. Therefore, the absolute enabling period is (6, 14) and the absolute executable period is (7, 15). Under the combined constraints coming from the absolute enabling period and the absolute executable period, the period of validity to guarantee the firing process of t_1 is (7,14). Because 14-7 = 7 = [7], t_1 can complete its firing process successfully.

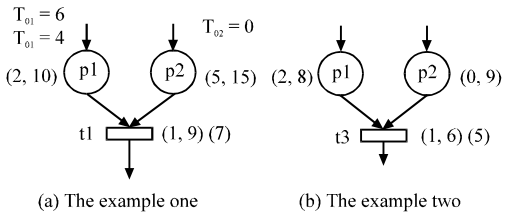

Fig. 3(a-b): The examples of timing constraint analysis and verification

Global timing constraint analysis and verification: Via local verification, only some simple timing constraint conflicts can be found. In order to guarantee all transitions can successfully complete their firing processes under the system timing constraints, we need to make global timing constraint analysis and verification. The initial marking M_0 and the arrival times of tokens in each place must be taken into account. The functional analysis and verification to a TCPN model can be performed by existing Petri net analysis technology, such as reachability analysis, without considering timing constraints. Therefore, we only focus on timing constraint analysis and verification to guarantee the correctness of system timing behaviors.

Theorem 2: When each of the input places of a transition t_j has a certain token arrival time, t_j can successfully complete its firing process if and only if:

$$EFBT(t_j) = Max\{TK_{arr}(p_i) + TC_{min}(p_i)\} + TC_{min}(t_j)$$
$$LFET(t_j) = Min\{Max\{TK_{arr}(p_i) + TC_{min}(p_i)\} + TC_{max}(t_j),$$
$$Min\{TK_{arr}(p_i) + TC_{max}(p_i)\}\} \text{ and } LFET(t_j) - EFBT(t_j) \geq FIRE_{dur}(t_j)$$

where, $p_i \in IP(t_j)$.

Proof: The enabling period $(EEBT(t_j), LEET(t_j))$ of t_j is determined by every enabling interval on the input places of it, so:

$$EEBT(t_j) = Max\{TK_{arr}(p_i) + TC_{min}(p_i)\} \text{ and}$$
$$LEET(t_j) = Min\{TK_{arr}(p_i) + TC_{max}(p_i)\}$$

where $p_i \in IP(t_j)$.

When considering the enabling startup time, i.e., $EEBT(t_j)$, the absolute executable period of t_j is $(EEBT(t_j)+TC_{min}(t_j), EEBT(t_j)+TC_{max}(t_j),)$. Therefore, under

the combined constraints coming from the enabling period and the executable period, the period of validity (EEBT(t_j), LFET(t_j)) is:

$$\begin{aligned}&EFBT(t_j) = EEBT(t_j) + TC_{min}(t_j) = \\ &\quad Max\{TK_{arr}(p_i) + TC_{min}(p_i)\} + TC_{min}(t_j) \text{ and}\\ &LFET(t_j) = Min\{EEBT(t_j) + TC_{max}(t_j), LEET(t_j)\}\\ &\quad = Min\{Max\{TK_{arr}(p_i) + TC_{min}(p_i)\} + TC_{max}(t_j),\\ &\quad Min\{TK_{arr}(p_i) + TC_{max}(p_i)\}\}\end{aligned}$$

where, $p_i \in IP(t_j)$. If $LFET(t_j)-EFBT(t_j)>0$, then t_j can be fired successfully. However, only when $LFET(t_j)-EFBT(t_j) \geq FIRE_{dur}(t_j)$, t_j can complete its firing process successfully.

For example, in Fig. 3a, when $Tk_{arr}(p_1) = 6$ and $Tk_{arr}(p_2) = 0$, according to Theorem 2, $EFBT(t_1) = Max\{6+2, 0+5\}+1 = 9$, $LFET(t_1) = Min\{Max\{6+2, 0+5\}+9, Min\{6+10, 0+15\}\} = 15$ and $LFET(t_1)-EFBT(t_1) = 6<FIRE_{dur}(t_j) = 7$, so t_j can not complete its firing process successfully. However, when $Tk_{arr}(p_1) = 4$ and $Tk_{arr}(p_2) = 0$, according to Theorem 2:

$$\begin{aligned}EFBT(t_1) &= Max\{4+2, 0+5\}+1 = 7\\ LFET(t_1) &= Min\{Max\{4+2, 0+5\}+9\\ &\quad Min\{4+10, 0+15\} = 14 \text{ and}\\ LFET(t_1)-EFBT(t_1) &= 7 = FIRE_{dur}(t_1) = 7\end{aligned}$$

so t_j can complete its firing process successfully.

In contrast with classical TCPN analysis methods, Theorem 2 is more strict and exact. For example, in the research of Tsai *et al.* (1995), when the arrival times of tokens are considered, a transition t_j is called a Strongly Schedulable Transition (SST) if and only if:

$$\begin{aligned}EFBT(t_j) &= Max\{Min\{TK_{arr}(p_j)\}+TC_{min}(p_j)\}+TC_{min}(t_j)\\ LFET(t_j) &= Min\{Max\{TK_{arr}(p_j)\}+Min\{TC_{max}(p_j)\}, Tc_{max}(t_j)\\ &\quad LFET(t_j)-EFBT(t_j) \geq FIRE_{dur}(t_j)\end{aligned}$$

where, $p_j \in IP(t_j)$.

However, in fact, even if a transition is not a SST, it can still complete its firing process successfully in some situation. For example, in Fig. 3b, we suppose $Tk_{arr}(p_1) = 6$ and $Tk_{arr}(p_2) = 7$. According to the definition of SST:

$$\begin{aligned}EFBT(t_3) &= Max\{6+2, 6+0\}+1 = 9\\ LFET(t_3) &= Min\{7+Min\{8, 6\}, 7+Min\{9,6\} = 13 \text{ and}\\ LFET(t_3)-EFBT(t_3) &= 13-9 = 4<[5]\end{aligned}$$

so t_3 is not a SST. However, through analysis, we can know the absolute enabling period of t_3 is $(Max\{6+2, 7+0\}, Min\{6+8, 7+9\}) = (8,14)$ and the absolute executable period is $(8+1, 8+6) = (9, 14)$. Therefore, the period of validity of t_3 is (9,14). Because $14-9 = 5 = [5]$, t_3 can complete its firing process successfully. According to Theorem 2, we also can get the period of validity of t_3 is (9,14).

It is important to note TCPN follows weak firing mode, which does not force any enabled transition to fire immediately. Therefore, in most situations, the token arrival time of a place p_j has a time interval denoted as $(ETK_{arr}(p_j), LTK_{arr}(p_j))$. $ETK_{arr}(p_j)$ is the earliest token arrival time of p_j and $LTK_{arr}(p_j)$ is the latest token arrival times of p_j.

Theorem 3: The token arrival time interval of a place p_j can be got by following formulas:

$$\begin{aligned}&ETK_{arr}(p_j)=Min\{EFBT(t_i) + FIRE_{dur}(t_i)\} \text{ and}\\ &LTK_{arr}(p_j) = Min\{LFET(t_i)\}\end{aligned}$$

where, $t_i \in IT(p_j)$.

Proof: A transition t_i ends its firing process earliest at $EFBT(t_i)+FIRE_{dur}(t_i)$ and latest at $LFET(t_i)$. After t_i completes its firing process, the movement of tokens from each input place to each output place of t_i uses no time. Therefore, when multiple input transitions are considered:

$$\begin{aligned}&ETK_{arr}(p_j)=Min\{EFBT(t_i) + FIRE_{dur}(t_i)\} \text{ and}\\ &LTK_{arr}(p_j) = Min\{LFET(t_i)\}\end{aligned}$$

When the token arrival time interval needs to be considered, we calculate EFBT and LFET by forward and backward calculation respectively. In forward calculation, we suppose every transition is fired immediately once it can be fired, so the token arrival time is ETK_{arr} for every output place of these transitions, i.e., $Tk_{arr}(p_j) = ETK_{arr}(p_j)$. Therefore, after the initial token arrival time of a system is ascertained, we can calculation EFBT of every transition by Theorem 2 and 3 step by step. In backward calculation, if a transition t_j with bounded value of $LFET(t_j)$ can be found, then we can calculate the LFET values for all the transitions which are fired before t_j based on the firing sequence. The backward calculation formula is presented in following Theorem 4.

Theorem 4: The latest fire ending time of t_j is:

$$LFET(t_i) = Min\{LFET(t_j) - FIRE_{dur}(t_j) - TC_{min}(p_j)\}$$

where $p_j \in IP(t_j)$ and $p_j \in OP(t_i)$.

Proof: $LFET(t_j) = FIRE_{dur}(t_j)$ is the latest fire beginning time of t_j, so $LTK_{arr}(p_j)$ is $LFET(t_j)-FIRE_{dur}(t_j)-TC_{min}(p_j)$. When t_i has multiple output places, in order to guarantee each $LTK_{arr}(p_j)$ is satisfied, $LFET(t_i) = Min\{LTK_{arr}(p_j)\}$.

For example, from system specification S1 in previous modeling example of a patient monitoring system, we know the system must response to emergent situations within 50 TU and response to normal situations within 50 TU after acquiring monitoring data. Therefore, in Fig. 2, if

we assume t_1 ends its firing at T_0, then we can set $LFET(t_9) = LFET(t_{10}) = T_0+50$ to guarantee the system makes response in time. According to Theorem 4, we can get:

$$LFET(t_7) = LFET(t_9) - FIRE_{dur}(t_9) - TC_{min}(p_{10}) = T_0+50-5-0=T_0+45$$

Through forward and backward calculation, we can get EFBT and LFET of each transition. If $LFET(t_j)$-$EFBT(t_j)<FIRE_{dur}(t_j)$, then a global timing constraint conflict is found.

AN EXAMPLE OF TIMING CONSTRAINT ANALYSIS AND VERIFICATION OF IOT REAL-TIME SYSTEM

We still use above mentioned patient monitoring system to make timing constraint analysis and verification. The initial TCPN model of the system has been shown in Fig. 2. First, we make local timing constraint verification via the local verification conditions (1) to (3). Two timing constraint conflicts can be found in the initial model. They are:

$$Tc_{max}(t_8)-TC_{min}(t_8) = 22-15 = 7<FIRE_{dur}(t_8) = (10)$$
$$\text{and } Tc_{max}(p_9)-TC_{min}(p_9)-TC_{min}(t_8) = 23-0-15 = 8<FIRE_{dur}(t_8) = (10)$$

To eliminate this two timing constraint conflicts, we relax the timing constraints by increasing $Tc_{max}(t_8)$ from 22 to 25 and $TC_{max}(p_9)$ from 23 to 25, so that:

$$TC_{max}(t_8)-TC_{min}(t_8) = 25-10 = 7<FIRE_{dur}(t_8) = (10) \text{ and}$$
$$TC_{max}(p_9)-TC_{min}(p_9)-TC_{min}(t_8) = 22-0-15 = 10 = FIRE_{dur}(t_8) = (10)$$

After eliminating local timing constraint conflicts, the TCPN model is shown in Fig. 4a.

Next, we make global timing constraint verification. In Fig. 4(a), t_1 is a special transition with responsibility to sample the physical information of a patient. In fact, in the continual system running process, t_1 runs in parallel with other transitions. Therefore, we assume t_1 ends its firing at T_0 according to the analysis tradition of TCPN.

We make forward calculation first. Based on Theorem 2, we can get:

$$EFBT(t_3) = Tk_{arr}(p_2)+TC_{min}(t_2) = T_0+6+0 = T_0+6$$

And then, based on Theorem 3, we can get $ETK_{arr}(p_3) = EFBT(t_2)+FIRE_{dur}(t_2) = (T_0+6)+5 = T_0+11$. According to the analysis in previous subsection, we know $Tk_{arr}(p_3) = ETK_{arr}(p_3)$ in forward calculation. Therefore, based on Theorem 2, we can get:

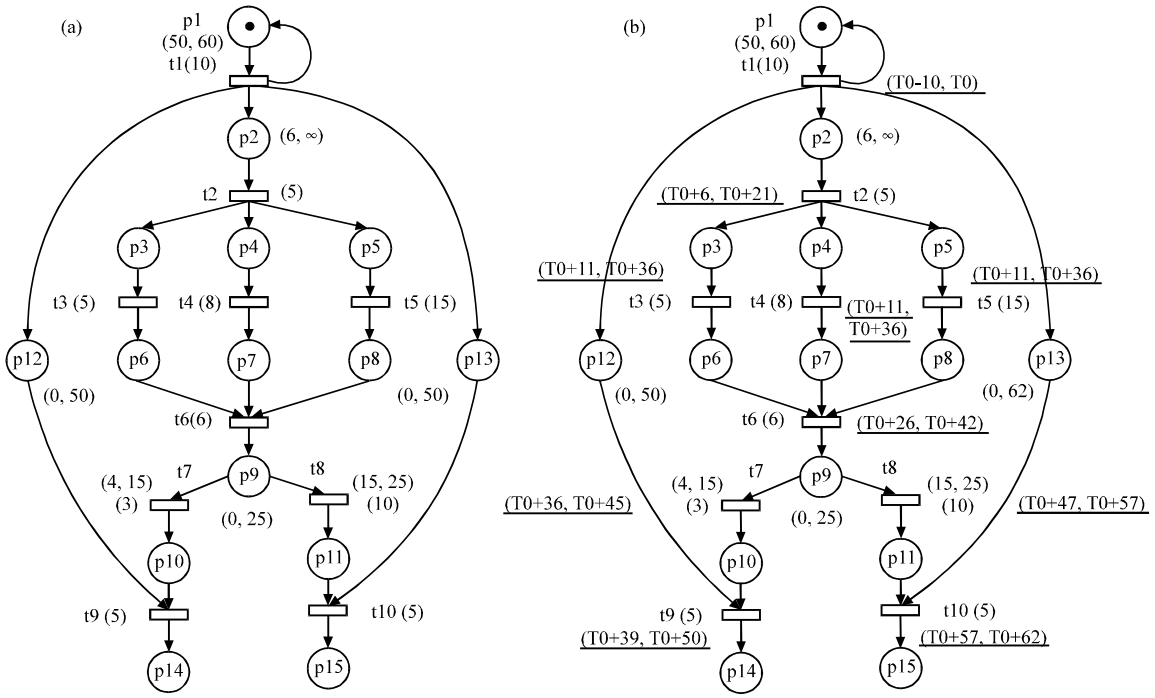

Fig. 4(a-b): The timing constraints analysis and verification of the patient monitoring system

Table 1: Results of global timing constraint analysis

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
$FIRE_{dur}$	10	5	5	8	15	6	3	10	5	5
EFBT	T_0-10	T_0+6	T_0+11	T_0+11	T_0+11	T_0+26	T_0+36	T_0+47	T_0+39	T_0+57
LFET	T_0	T_0+14	T_0+29	T_0+29	T_0+29	T_0+35	T_0+45	T_0+45	T_0+50	T_0+50

Table 2: Periods of validity of transitions in the final model

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9	t_{10}
$FIRE_{dur}$	10	5	5	8	15	6	3	10	5	5
EFBT	T_0-10	T_0+6	T_0+11	T_0+11	T_0+11	T_0+26	T_0+36	T_0+47	T_0+39	T_0+57
LFET	T_0	T_0+21	T_0+36	T_0+36	T_0+36	T_0+42	T_0+45	T_0+57	T_0+50	T_0+62

$$EFBT(t_3) = Tk_{arr}(p_3)+TC_{min}(t_3) = (T_0+11)+0+0 = T_0+11$$

According to the same method, we can get the EFBT values of other transitions. The calculation results of EFBT are shown in Table 1.

After finishing forward calculation, we make backward calculation. In previous subsection, we have known $LFET(t_9) = LFET(t_{10}) = T_0+50$ and $LFET(t_7)$-0-0 = T_0+45. According to Theorem 4, we can get $LFET(t_8) = LFET(t_{10})-TC_{min}(p_{11}) = (T_0+50)-5-0 = T_0+45$. Therefore:

$$LFET(t_6) = Min\{LFET(t_7)-FIRE_{dur}(t_{10})-TC_{min}(p_9), LFET(t_8)-FIRE_{dur}(t_8)-TC_{min}(p_9)\} = Min\{T_0+45-3-0, T_0+45-10-0\} = T_0+35$$

According to the same method, we can get the LFET values of other transitions. The calculation results of LFET are shown in Table 1.

From Table 1, we can find two global timing constraint conflicts. They are:

$$LFET(t_8)-EFBT(t_8) = (T_0+45)-(T_0+47) = -2<FIRE_{dur}(t_8) = (10) \text{ and } LFET(t_{10})-EFBT(t_{10}) = (T_0+50)-(T_0+57) = -7<FIRE_{dur}(t_{10}) = (5)$$

To eliminate this two timing constraint conflicts, we relax the timing constraint by increasing $TC_{max}(p_{13})$ from 50 to 62, so that $LFET(t_{10}) = T_0+62$. Via recalculation, we get the periods of validity of transitions in the final model, in which there are no timing constraint conflicts left. The results of recalculation are shown in Table 2 and the final model is shown in Fig. 4b. The periods of validity of transitions are underlined in the final model.

CONCLUSION AND FUTURE WORK

By using TCPN, an IoT real-time system with various categories of timing constraints can be modeled accurately. By taking advantage of the timing constraint analysis and verification methods presented in this study, the local and global timing constraint conflicts can be found and eliminated and the activities producing conflicts can be ascertained. In addition, via analysis and calculation, the period of validity of each system activity, i.e., the absolute time range for a activity to execute, can be obtained. This facilitates the monitoring and management of the system activities. Through timely and successful response of each system activity, the high stability and reliability of IoT real-time system are guaranteed.

Some problems are left and still need to be resolved in the future work. When global timing constraint conflicts are found, we most probably eliminate them by adjusting multiple existing timing constraints. How to find a suitable adjustment scheme to reset new timing constraint values from a global viewpoint is a difficult and important problem. In addition, the algorithms of automatic model analysis and verification are needed for large-scale IoT applications. These problems will be researched in our future work.

ACKNOWLEDGMENT

This research was sponsored by Shaanxi Province Research and Development Program in Science and Technology of China under the grant No. 2011k06-33.

REFERENCES

Atzori, L., A. Iera and G. Morabito, 2010. The internet of things: A survey. Comput. Networks, 54: 2787-2805.

Dai, G.L., R.J. Liu, C.C. Zhao and C.J. Hu, 2008. Timing constraints specification and verification for Web service compositions. Proceedings of the IEEE Asia-Pacific Services Computing Conference, December 9-12, 2008, Yilan, Taiwan, pp: 315-322.

Funk, M., P.V.D. Putten and C. Henk, 2009. Analytics for the internet of things. Proceedings of the 27th International Conference on Human Factors in Computing Systems, April 4-9, 2009, Boston, MA., USA., pp: 4195-4200.

Garcia-Macias, J.A., J. Alvarez-Lozano, P. Estrada-Martinez and E. Aviles-Lopez, 2011. Browsing the internet of things with sentient visors. Computer, 44: 46-52.

Giner, P., C. Cetina, J. Fons and V. Pelechano, 2010. Developing mobile workflow support in the internet of things. IEEE Pervasive Comput., 9: 18-26.

Guo, B., D.Q. Zhang, Z.W. Yu, Y.J. Liang, Z. Wang and X.S. Zhou, 2012. From the internet of things to embedded intelligence. World Wide Web. 10.1007/s11280-012-0188-y

Juan, E.Y.T., J.J.P. Tsai, T. Murata and Y. Zhou, 2001. Reduction methods for real-time systems using delay time Petri nets. IEEE Trans. Software Eng., 27: 422-448.

Kortuem, G., F. Kawsar, D. Fitton and V. Sundramoorthy, 2010. Smart objects as building blocks for the internet of things. IEEE Internet Comput., 14: 44-51.

Li, L.X., Z. Jin and G. Li, 2011. Modeling and verifying services of internet of things based on timed automata. Chin. J. Comput., 34: 1365-1377.

Li, X., J. Liu, Q.Z. Sheng, S. Zeadally and W.C. Zhong, 2009. TMS-RFID: Temporal management of large-scale RFID applications. Inform. Syst. Front., 13: 481-500.

Munoz-Organero, M., G.A. Ramirez-Gonzalez, P.J. Munoz-Merino and C.D. Kloos, 2010. Recommender system based on space-time similarities. IEEE Pervasive Comput., 9: 81-87.

Shen, S.B., Y.Q. Mao, Q.L. Fan, P. Zong and W. Huang, 2010. The concept model and architecture of the internet of things. Chin. J. Nanjing Univ. Posts Telecommun., 30: 1-8.

Tsai, J.J.P., S.J. Yang and Y.H. Chang, 1995. Timing constraint Petri nets and their application to schedulability analysis of real-time system specifications. IEEE Trans. Software Eng., 21: 32-49.