Scheduling Real-Time Embedded Systems Based on TCPNIA

Nianhua Yang[1, 2], Huiqun Yu[1], Hua Sun[1], Zhilin Qian[1]
[1]Department of Computer Science and Engineering
East China University of Science and Technology
Shanghai 200237, China
[2]Shanghai Key Laboratory of Computer Software Evaluating and Testing
Shanghai 201112, China
cnynh@163.com, yhq@ecust.edu.cn, xj_sh@163.com, ssssdc@163.com

***Abstract*—TCPNIA (Timed Colored Petri Nets with Inhibitor Arcs, TCPNIA) is a model for specifying real-time embedded systems. It integrates features of colored Petri nets, timed Petri nets and inhibitor arcs. The methods for modeling modules of systems using TCPNIA are proposed. A depth-first scheduling algorithm for TCPNIA is proposed. The system level resource's influence to schedulable path is considered. Different data in tokens will change an execution path and call different data operational functions. The influence from data operational functions is considered in the scheduling algorithm. The time requirement upper bound of a schedulable path can be calculated in the algorithm. Tasks' parallel executions have been considered when the time upper bound of the path is calculated. The soundness of the algorithm is proved. The time and space complexities of the algorithm are also analyzed. The compositional conditions and method for composing schedulable path are given. A case study shows the applicability and feasibility of the method.**

***Index Terms*—embedded system; real-time property; Petri net; modeling; scheduling**

I. INTRODUCTION

Over the last years, embedded systems have received considerable attention. These systems are usually mission and safety-critical. The logical and functional results of the computation must be correct. Moreover it should meet certain time constraints. In hard real-time systems, the violation whether to functions or to time constraints may lead to catastrophic consequence, such as equipment damage, environment pollution, or even loss of human lives. Resource consumption has also to be considered, due to the running costs of the system and environment problems.

Designing systems with such characteristics is a difficult task. Design method with well-defined formal semantics is needed for specification and implementation of the system. Petri nets [1] are formal models based on strict mathematical theories. They are powerful models and appropriate for modeling systems with parallelization, synchronization and confliction. Plenty of theoretical results and practical tools have been developed around Petri nets. Many real-time embedded systems modeling techniques have been proposed [2-3]. They are extended from Petri nets. But some of them are absence of the ability to describe the tasks' priorities, others can not use the traditional Petri nets analysis methods.

Timed Petri nets [4] allow transitions to be executed for a period, which can be used to express events' time consumption. Colored Petri nets [5] allow tokens to have multiple data and have the ability to specify complicated computing. Inhibitor arcs can be used for describing events temporal order [1]. Timed colored Petri nets with inhibitor arcs (TCPNIA) [6] integrates the above three models to facilitate modeling and analyzing real-time embedded systems. In addition, the existing Petri nets techniques can be adopted to analyze properties of the system.

Scheduling plays an important role in real-time embedded systems. It can produce feasible paths guaranteeing time and resource requirements. But previous scheduling algorithms mainly concentrated on time Petri nets. Scheduling on timed Petri nets is rarely researched. Further more, previous scheduling algorithms only considered the influences from the logic conditions of the system, ignoring the influences from the data and functional properties. Pre-runtime scheduling can reduce time and resource consumption in run time.

This paper proposes a depth-first heuristic search pre-runtime scheduling algorithm without generating the whole state space. It provides a scheduling method for timed Petri nets. To calculate time consumption, multi-processors' parallel executions are considered in the algorithm. The influences to the scheduling result from the input data and modules' data operational functions are included in the algorithm.

The main contributions of this paper include:

- TCPNIA is used to represent real-time embedded systems. Modular modeling methods for complicated systems using TCPNIA are proposed. They can reduce the complexity of modeling and enhance the reusability of modules.

The work was supported by the NSF of China under grants No. 60773094 and 60473055, Shanghai Shuguang Program under grant No. 07SG32.
Corresponding author: Huiqun Yu.

- A novel scheduling algorithm for TCPNIA is proposed. The algorithm considers influences from data and their operational functions. The influence from system level abstract resource's constraints to scheduling result is also considered. The influences from data operational functions and system level abstract resource are rarely considered in present literatures. The scheduling algorithm is also applied to traditional timed Petri nets.
- The time requirement bounds of a schedulable path in a TCPNIA model can be got directly from the end state in the scheduling algorithm. The transitions' parallel executions are considered during calculating the time bounds in a schedulable path.

The rest of this paper is organized as follows: Section II describes related work about modeling and scheduling real-time embedded systems and indicates the differences between previous works and our method; Section III describes syntax and semantics of TCPNIA; The modeling methods of a real-time embedded system using TCPNIA are described in Section IV; Section V proposes a scheduling algorithm and analyses the algorithm's soundness and compositional properties; Section VI presents a realistic case to illustrate the feasibility of our method; And Section VII concludes this paper.

II. Related Work

Many models have been proposed to represent embedded systems. Particularly, Petri nets have been extended to model such systems. Peng et al. [7] introduce two separate graphs to describe data operation and control process respectively. But it can not reflect the interrelations of data and control information explicitly. PRES/PRES+ (Petri net based Representation for Embedded Systems, PRES) [2, 8] can represent data operation and control process in a single figure. They include data information in a token and describe time information explicitly. But they can not describe events' temporal orders. It can't express complicated data and control flows perfectly. DFN (Dual Flow Net, DFN) [3] introduces data transitions and control transitions separately to process data and control information respectively in the same net. But it's semantics is different obviously from traditional Petri nets. The analysis techniques of traditional Petri nets can't be utilized in such a model. EPRES (Extended PRES, EPRES) [9], developed from PRES+, also can not utilize the analysis techniques of traditional Petri nets. TCPNIA integrates the ability of timed, colored Petri nets and inhibitor arcs to facilitate modeling and analyzing real-time embedded systems. Data, control information and their temporal relations can be represented explicitly in a single figure. In addition, the existing Petri nets techniques can be adopted to analyze properties of the system represented with TCPNIA.

A lot of scheduling algorithms [10-14] have been developed for real-time systems in the previous decades. They do not consider the influence of system's data operational functions. Some of them [12-14] even do not consider the resource constraints. Xu et al. [14] propose a compositional approach to the schedulability analysis of real-time systems modeled in time Petri nets. The absolute time constraints are checked to determine the schedulability of a path. An absolute time consuming calculating method is provided. Jejurikar et al. [11] propose an energy-aware task scheduling method for embedded real-time systems. Like many of other energy-aware scheduling methods, that paper only focused on DVS (Dynamic Voltage Scaling, DVS) of a processor, and did not consider the other resources in system level. Further more, previous researches of scheduling methods mostly focus on time Petri nets. The scheduling method based on timed Petri nets is rarely considered.

In this paper, we propose a novel pre-runtime scheduling algorithm for TCPNIA, which is extended from timed Petri nets. It has considered resource constraints and data operational functions' influences. The method focuses on abstract resources, which can be fuel, power et al. It can also be extended to process multi types of resources. Different data values in tokens can change the execution path. And the system will call different data operational functions based on different input data. Thus, the schedulability of a TCPNIA model will be influenced by the data operational functions.

Similar to Xu's method [14], absolute time conception is used to calculate the actual time consumption of a schedulable path. The absolute time requirement from the initial state is calculated when deciding the schedulability of a transition. The method to calculate the absolute time used by Xu can not be used for TCPNIA for the semantics differences between timed Petri nets and time Petri nets. We use tokens' updated time and transitions' time consumption domain bounds to calculate time requirement bounds of a schedulable path in a TCPNIA model. The parallel executions have been considered in the calculating method. The time consumption bounds of the schedulable path are equal to the timestamp's bounds of the final state's marking. It can be got directly from the last state in the scheduling algorithm.

III. Computational Model

Colored Petri nets, timed Petri nets and inhibitor arcs are integrated into TCPNIA [6]. The syntax and semantics of TCPNIA are given in this section.

A. TCPNIA

A timed colored Petri net with inhibitor arcs is a tuple $TCPNIA =< P, T, A, B, I, W, G, F, R, D, M_0 >$, where P is a finite set of places; T is a finite set of transitions; $A \subseteq (T \times P)$ is a set of output arcs; $B \subseteq (P \times T)$ is a set of input arcs; $I \subset P \times T$ is the set of inhibitor arcs, which satisfies $I \bigcap (A \bigcup B) = \varnothing$; $W : (A \bigcup B) \rightarrow 1$ is the weight of arcs; $G : 2^Z \rightarrow \{true, false\}$ is a set of guard functions defined on transitions, or $g_i = true$ if the transition t_i has not been given a guard function explicitly; F is the set of transitions' data operational functions; $R : T \rightarrow \mathbb{R}^*$ is a set of non-negative real numbers, which represent the resource consumptions of the transitions; $D = [\alpha(t), \beta(t)]$ is the set of transitions' execution time domains, which implies that if a transition t begins to execute, it should last for the time of $d(t)$, where $\alpha(t) \le d(t) \le \beta(t)$; M_0 is the initial marking. The capacity of each place is restricted to one.

The content of a token can be a real number or a Boolean value. The real number in a token is used as the input of guard functions or data operational functions. The Boolean value is only used for the purpose of model's logic control. ${}^\circ t = \{p \in P \mid (p,t) \in B\}$ is used to denote the common pre-set of t. $t^\circ = \{p \in P \mid (t,p) \in A\}$ is used to denote the common post-set of t. ${}^* t = \{p \mid (p,t) \in I\}$ is used to denote the inhibitory pre-set of t. $\tau(p)$ is used to represent the content's type of the place p. Once the type of a special place is given, it will not be changed. We give a rule that each place in the post-set of a special transition has the same data type, expressed by $\tau(t^\circ)$. The token number of the place p in the marking M is denoted as $token(M,p)$. The value of $token(M,p)$ is 1 or 0. $\nu(M,p)$ is used to represent the content value of place p in the marking M.

The guard function $g_t \in G$ is defined on the transition t. It maps all the input values to a Boolean value, i.e. $g_t : \tau(p_1) \times \tau(p_2) \times \cdots \times \tau(p_a) \to \{true, false\}$, where $\{p \mid p \in {}^\circ t\} = \{p_1, p_2, \cdots p_a\}$. After firing the transition t, the value in each post-set place of t is decided by $f_t : \tau(p_1) \times \tau(p_2) \times \cdots \times \tau(p_a) \to \tau(t^\circ)$, where $f_t \in F$, ${}^\circ t = \{p_1, p_2 \cdots, p_a\}$. If f_t is not defined, the value can be any element in $\tau(p)$, where $p \in t^\circ$.

B. Enabled Conditions and Firing Rules

Let us assume the current marking is M. Some transitions may be enabled in the marking M. And some of them may fire concurrently.

A transition t is said to be enabled if and only if each place p_i in the common pre-set of t has a token, each place p_j in the inhibitory pre-set of t is empty, each of its out places different from the places in the common pre-set is empty, and the values in common pre-set of t satisfy $g_t = true$.

Supposing a transition t has a common pre-set ${}^\circ t = \{p_1, \quad p_2, \cdots, p_a\}$, an inhibitory pre-set ${}^* t = \{p_1, p_2, \cdots p_b\}$, and a common post-set $t^\circ = \{p_1, p_2, \cdots, p_c\}$. If it is enabled, it can be fired immediately and lasted for $d(t)$ time units, where $\alpha(t) \le d(t) \le \beta(t)$.

The firing of an enabled transition t changes the marking M into a new marking M', lasting for $d(t)$ time units and consuming $r(t)$ resource units, which is denoted as $M \xrightarrow{t, d(t), r(t)} M'$. As a result of firing the transition t, next three events occur in turn:

a) $\forall p_i \in {}^\circ t \Rightarrow M'(p_i) = 0$;

b) $\forall p_j \in t^\circ \Rightarrow M'(p_j) = 1$;

c) $\forall p_j \in t^\circ$, if the data operational function f_t is defined, then $\nu(M', p_j) = f_t(p_1, p_2, \cdots, p_a)$, otherwise $\nu(M', p_j) = 1$.

C. States in a TCPNIA

The timestamp of the initial marking M_0 is supposed to be zero.

A state S of a TCPNIA can be defined as a pair $S = (M, TS)$, where:

a) M is a marking of a TCPNIA, and

b) $TS : M \to \mathbb{R}^*$ represents the timestamp of the marking M, where M is reached from the initial marking M_0, TS is the abstract time relative to the initial state, $L \le TS \le U$ and $L, U \in \mathbb{R}^*$.

The timestamp domain of the marking M are denoted as $[L(M), U(M)]$.

IV. Modeling Real-Time Embedded Systems

This section describes modular modeling methods for complicated systems using TCPNIA. They can reduce the complexity of modeling and enhance the reusability of modules. They also provide foundation for hierarchical modeling and reusability of analysis results.

A. Graphical Notations

TCPNIA is denoted by weighted directed graph. Place is expressed with a circle. Transition is expressed with a rectangle. An inhibitor arc connects a place to a transition and is represented by a dashed line terminating with a small circle instead of an arrowhead at the transition. The expression on the arc represents the variable of the value in the related place. The predicate formula in the left (or top) of the rectangle denotes the guard function of the related transition. The function formula in the right (or bottom) of the rectangle denotes the data operational function of the related transition. The symbol $[\alpha(t), \beta(t)]$, at the side of the rectangle, denotes the domain of the related transition's firing duration time, where $\alpha(t)$ and $\beta(t)$ are two non-negative real numbers and satisfy the following constraints $\alpha(t) \in \mathbb{R}^* \wedge \beta(t) \in \mathbb{R}^* \wedge \alpha(t) \le \beta(t)$. The symbol $< R(t) >$ represents the resource requirement to finish the firing of the transition, where $R(t) \in \mathbb{R}^*$ is a non-negative real number. It is marked at the side of the rectangle.

B. Data and Control Modeling

Embedded systems specification usually consists of control functions and data operational functions. TCPNIA can specify data operational functions and control functions simultaneously in the same figure. A place's marking represents the logic value, i.e. the empty of the mark represents false, otherwise it represents true. The token's value represents data information used by operational function. The transition with data operational function represents data operation, otherwise it represents control function. Guard function reflects the influence from data information to control logic. Fig. 1(a) shows an example of data operation with guard function modeled with TCPNIA. Fig. 1(b) shows an example of control function modeled with TCPNIA.

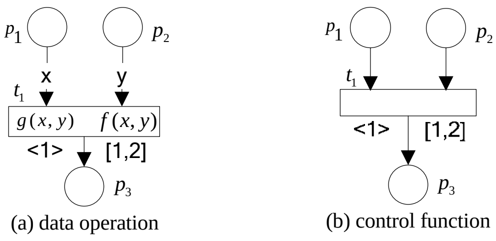

Figure 1. Example of data and control modeling

C. Modules Modeling

A real-time embedded system is composed by several different types of components, which are called modules [15]. Conflict, parallel and sequence modules' modeling methods are similar to the traditional Petri nets [1]. We propose here the following modules: branch, synchronization, inhibitor, loop, mutual exclusion and communication modules. They are depicted in Fig. 2 and simply described as follows.

1) Branch Module

Supposing that the system has n tasks after the task t_1, such as shown in Fig. 2(a). These tasks enabled at the same time after the firing of task t_1. Also, they need the out of t_1 as their input data or guard conditions. As a result, the n tasks get the same enabled time. Branch module can represent this situation.

2) Synchronization Module

Usually parallel tasks need to be synchronized with each other at some point. The synchronization module, shown in Fig. 2(b), states that all the previous tasks have finished, and they need to be synchronized for assembly.

3) Inhibitor Module

The net shown in Fig. 2(c) represents a system with priority. The transition t_1 has priority over the transition t_2. The transition t_1 can fire as long as place p_1 has a token and p_3 is empty. But if t_2 want to fire, not only p_2 should have a token and p_4 should be empty, but also p_1 should be empty. It has been shown that such a system can only be molded with inhibitor arcs[16], like the arc from p_1 to t_2 in Fig. 2(c).

4) Loop Module

Fig. 2(d) shows that the transition t_1 may be called several times continuously when the guard condition is satisfied. The transition t_1 may represent a module. During that time, other transitions, whose input places include p_1, can not be executed, because their guard conditions can not be satisfied, for $g_1 = \neg g_2$ in Fig. 2(d).

5) Mutual Exclusion Module

It is common in embedded systems that some modules should execute mutually exclusively. Fig. 2(e) shows that $Module_1$ and $Module_2$ can not execute at the same time. One of them can begin the execution only after the other has finished the execution or the other is idle. The place p_0 is the control place of the mutually exclusive modules.

6) Communication Module

In the TCPNIA model, the communication of data or control information between modules is realized by sharing the common places. The shared common places are out places of one module and the input places of the other module. To communicate between $Module_1$ and $Module_2$ in Fig. 2(f), the following two conditions should be fulfilled: (a) $m = n$; (b) $\tau(p_{1i}) = \tau(p_{2i})$, where $i = 1, 2, \cdots, m$.

D. Time and Resource Modeling

Any task needs a time duration to finish a special function. The time a transition t needs to finish the firing is expressed by $d(t)$, where $\alpha(t) \le d(t) \le \beta(t)$. The real number domain $[\alpha(t), \beta(t)]$ is labeled at the side of the transition t, such as the mark $[1, 2]$ in Fig. 2(b).

Resource requirement is also labeled at the transition in the model. It represents the resource consumption in the firing of the corresponding transition, like the mark $<1>$ in Fig. 2(b).

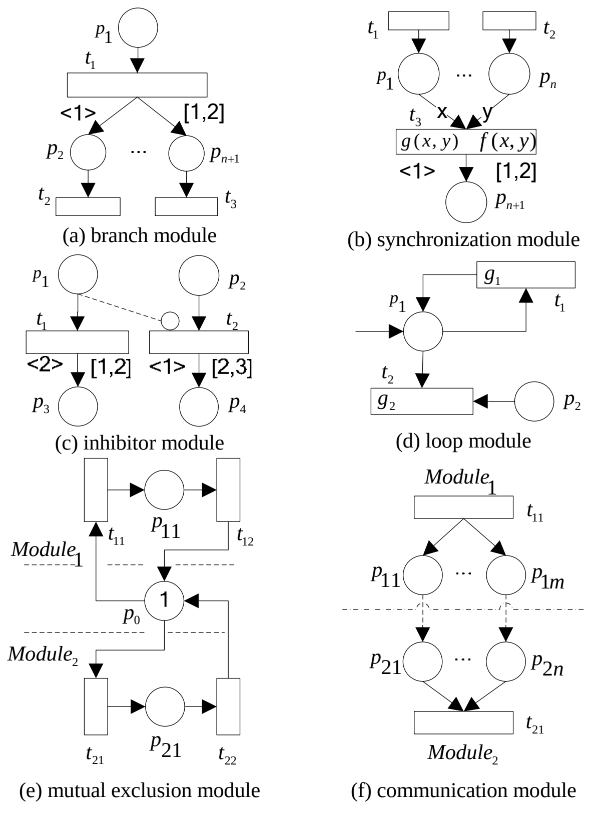

Figure 2. Proposed modeling modules

V. Scheduling Analysis

We can get the TCPNIA model of a real-time embedded system through composing the modules build through the methods in Section IV. The model is not only used for representing the given specification, but also for analyzing and verifying the system properties. This section will give a scheduling method of a TCPNIA model. Verification methods will not be discussed in this paper for the space limitation.

To conveniently describe the scheduling algorithm, we give the definition of equivalence between two markings in a TCPNIA model.

Definition 1 Two markings, M_1 and M_2, of a TCPNIA model are said to be equal if and only if

a) $\forall p \in P$, $token(M_1, p) = token(M_2, p)$; and

b) $\forall p \in P$, $v(M_1, p) = v(M_2, p)$.

A. Scheduling Algorithm

Scheduling is to find a feasible path from M_0 to M_{end} in a TCPINA model, satisfying time and resource constraints. A depth-first search algorithm for scheduling a TCPINA model is proposed in Fig. 3. The path can be got from part of the reachable states using this algorithm.

```
boolean scheduling(TCPNIA, $M$, $M_{end}$, R, $Ti$) {
    if( $M == M_{end}$ ) return true;
    enSet = en(M); //en(M) represents all enabled transitions in marking M
    while (not empty(enSet)){
        t=get(enSet); //get an enabled transition arbitrarily
        enSet = enSet - t;
        if(( $Ti - (U_f(t) + \beta(t)) \geq 0$ ) and ( $R - r(t) \geq 0$ ) and
                (scheduling(TCPNIA,firing(M,t), $M_{end}$, $R - r(t)$, $Ti$ )){
            put(stack, t);
            updateVariables(lp,up);
            $L(M') = \max(lp(M'))$;
            $U(M') = \max(up(M'))$;
            put(L, $L(M')$ );
            put(U, $U(M')$ );
        }
    }
    return false;
}
```

Figure 3. Scheduling algorithm of TCPNIA

Temporary variables lp and up are introduced to save the temporary results in the algorithm. $lp(M, p)$ and $up(M, p)$ represent the low and upper absolute time domain bounds when the token of place p is updated. They are set to be zero in the initial marking M_0.

Supposing a transition t has a common pre-set ${}^{\circ}t = \{p_1, p_2, \cdots, p_a\}$, an inhibitory pre-set ${}^{*}t = \{p_1, p_2, \cdots p_b\}$, and a common post-set $t^{\circ} = \{p_1, p_2, \cdots, p_c\}$. If it is enabled, it can be fired in the absolute time domain $[L_f(t), U_f(t)]$, where

$$L_f(t) = \max(\max_{i=1}^{a}(lp(M, p_i)), \max_{j=1}^{b}(lp(M, q_j)), \max_{k=1}^{c}(lp(M, q_k)))$$

and

$$U_f(t) = \max(\max_{i=1}^{a}(up(M, p_i)), \max_{j=1}^{b}(up(M, q_j)), \max_{k=1}^{c}(up(M, q_k)))$$

. After the firing of t, lp and up is updated through the process $updateVarialbes(lp, up)$ in the algorithm in Fig. 3. In the process $updateVarialbes(lp, up)$, the variables are updated as follows:

a) $\forall p_i \in {}^{\circ}t \Rightarrow lp(M', p_i) = L_f(t) + \alpha(t)$ and $up(M', p_i) = U_f(t) + \beta(t)$;

b) $\forall p_j \in t^{\circ} \Rightarrow lp(M', p_j) = L_f(t) + \alpha(t)$ and $up(M', p_j) = U_f(t) + \beta(t)$.

The algorithm described in Fig. 3 is recursive. This algorithm will give a feasible path from the initial state M to the target state M_{end}, with the total time constraint Ti and resource constraint R. It can be detailed with following steps:

a) If current marking is equal to the target marking, i.e. $M == M_{end}$, then the algorithm will return true and exit.

b) Put all enabled transitions in M in the variable enSet.

c) Get one of enabled transition t from enSet, and remove it from the set.

d) If the time upper bound and resource requirement can be satisfied, the algorithm try to recursively test the following state marking after fire t, i.e. recursively call the algorithm with new initial state marking, time and resource constraints. If a feasible path δ can be found through the recursive method, put every transition in the path δ in a stack respectively. The most early time and last time when a new marking M' can be got through the path δ are also put in a stack respectively.

e) If the feasible path does not exist, the algorithm will return false, otherwise return true.

f) If the algorithm returns true, the sequential transitions can be got from the variable stack.

Supposing we get a schedulable path, which includes $t_1 t_2 \cdots t_i \cdots t_n$, from M_0 to M_{end}. The resource consumption can be calculated by $\sum_{i=1}^{n} S(t_i)$. And the required time domain of system's executing is $[L(M_{end}), U(M_{end})]$, which can be got directly from the timestamp's bounds of the end state.

In the algorithm, the data value of a token is considered through the comparison of two markings. It means that the data operational function of each transition will influence the scheduling result. Further more, the algorithm uses the time domain bounds of a transition and its enabled time domain bounds to calculate the time constraints in a schedulable path. It has considered the parallel execution of transitions. The time consuming domain of a schedulable path can't be calculated by $[\sum_{i=1}^{n}\alpha(t_i),\sum_{i=1}^{n}\beta(t_i)]$, which did not consider the parallel execution instance.

B. Soundness of the Algorithm

Theorem 1 The scheduling algorithm will terminate in finite steps.

Proof: Every transition's execution in a TCPNIA model needs some time and resource. Given the time and resource limitations, the time will be expired or the resource will be exhausted after finite steps in the algorithm. The algorithm will terminate at that time. In the other situation, the algorithm will terminate due to no live transition exists after several steps. Or the algorithm will find a reasonable path, and then terminate. So the Theorem 1 holds.

Theorem 2 If the scheduling algorithm returns true, the access path is schedulable.

Proof:

a) The algorithm selects an enabled transition in every iterative step under the marking at that time. So every marking in the trace $M_0t_1M_1t_2M_2\cdots t_nM_n$ can be reached from the marking M_0, i.e. $t_1t_2\cdots t_n$ is a schedulable path without considering time and resource constraints.

b) Before the transition is selected, the algorithm affirms that the amount of remained resource is large than the resource requirement of the transition. The amount of remained resource is subtracted after adding the transition in the path. So the path got from the algorithm satisfying resource constraints.

c) Before adding a transition, the time requirement upper bound and the latest enabled time of the transition are considered through the inequation $Ti-(U_f(t)+\beta(t))\geq 0$. So the time constraint is satisfied in the path got through the algorithm in Fig. 3.

Besides, the algorithm returns true, so M_n is the target marking in the trace $M_0t_1M_1t_2M_2\cdots t_nM_n$. Thus, Theorem 2 holds.

C. Complexity of the Algorithm

The state space size is $O(k^m)$ in a Petri net with m places [17], where k is the states number of each place. But in the algorithm of Fig. 3, only partial places are visited and only partial states of each place are used. So the state space size is less than $O(k^m)$.

The TCPNIA model is supposed to have n transitions. The scheduling algorithm will be called $n^2/4$ times recursively at the most. Once the algorithm is called, all the n transitions' enabled conditions are checked. To decide whether a transition is enabled, m places' tokens are visited at the most. So the algorithm's time complexity is $O(n^3\times m)$ at worst. Once the algorithm is called, it will save n enabled transitions at the most. So the algorithm's space complexity is $O(n^3)$ at the worst.

D. Compositional Analysis of Scheduling

In this section, we will discuss how to conduct a scheduling of TCPNIA by decomposing the reachable markings. The similar method is first used by Xu [14]. The analysis of some repeated subsequence can be simplified through decomposition and composition. It can enhance component's reusability and reduce scheduling complexity.

Theorem 3 Let $\delta_1=(M_{10}t_{11}M_{11}\cdots t_{1m}M_{1m})(m\geq 1)$ and $\delta_2=(M_{20}t_{21}M_{21}\cdots t_{2n}M_{2n})(n\geq 1)$ be two schedulable sequences, where M_{10} and M_{20} are reachable from M_0. $\delta_1\delta_2$ is schedulable if and only if $M_{1m}==M_{20}$, $\sum_{i=1}^{m}R(t_{1i})+\sum_{j=1}^{n}R(t_{2j})\leq R_1+R_2\leq R$ and $U(M_{1m})+U(M_{2n})\leq T_1+T_2\leq T$, where R is the resource consumption constraint of the transitions from M_{10} to M_{2n}, T is the time consumption upper bound of the transitions from M_{10} to M_{2n}.

Proof:

a) Necessary condition:

$\delta_1=(M_{10}t_{11}M_{11}\cdots t_{1m}M_{1m})(m\geq 1)$ and $\delta_2=(M_{20}t_{21}\cdots t_{2n}M_{2n})(n\geq 1)$ are schedulable. Supposing that $\delta_1\delta_2=(M_{10}t_{11}M_{11}\cdots t_{1m}M_{20}t_{21}M_{21}\cdots t_{2n}M_{2n})$ is also schedulable. From the path $M_{10}t_{11}M_{11}\cdots t_{1m}$, the system can get only one marking, so we can get $M_{1m}==M_{20}$, $\sum_{i=1}^{m}R(t_{1i})+\sum_{j=1}^{n}R(t_{2j})\leq R_1+R_2\leq R$ and $U(M_{1m})+U(M_{2n})\leq T_1+T_2\leq T$.

b) Sufficient condition:

$\delta_1=(M_{10}t_{11}M_{11}\cdots t_{1m}M_{1m})(m\geq 1)$, $\delta_2=(M_{20}t_{21}M_{21}\cdots t_{2n}\quad M_{2n})(n\geq 1)$. Besides M_{10} and M_{20} are reachable from M_0. Moreover, $M_{1m}==M_{20}$. So M_{2n} can be reached from M_{10} through the path $t_{11}t_{12}\cdots t_{1m}t_{21}t_{22}\cdots t_{2n}$. In the condition, $\delta_1=(M_{10}t_{11}M_{11}\cdots t_{1m}M_{1m})$ is schedulable. So we can get $\sum_{i=1}^{m}R(t_{1i})\leq R_1$. In the same way, we can get $\sum_{j=1}^{n}R(t_{2j})\leq R_2$, $U(M_{1m})\leq T_1$ and $U(M_{2n})\leq T_2$. Such that, $\sum_{i=1}^{m}R(t_{1i})+\sum_{j=1}^{n}R(t_{2j})\leq R_1+R_2\leq R$ and $U(M_{1m})+U(M_{2n})\leq T_1+T_2\leq T$.

According to step a and b, Theorem 3 holds.

According to Theorem 3, the schedulability of sequence $\delta=\delta_1\delta_2\cdots\delta_i\cdots\delta_n$ can be decided respectively by $\delta_i(i=1..n)$ and the markings between them.

VI. CASE STUDY

This section describes how to model and schedule a train pulling in system using TCPNIA. Based on the scheduling result, the model is analyzed and amended to meet time and resource constraints.

A. *System Requirement*

A train pulling in system is a real-time embedded system composed of a number of control equipments. There are two special railways, slow railway and fast railway, such as shown in Fig. 4. The slow railway is special for slow trains. And the fast railway is special for fast trains. It has two platforms for trains to stop. One is special for slow trains. The other is a common platform both for slow and fast trains. But the fast train has the priority to use the common platform. Each platform can be provided only for one train to stop at the same time. The fast train can only stop at the common platform. The slow train can stop at any platforms as long as they are available. When the slow train has stop at the platform, the fuel meter should be checked. Adding fuel is necessary if and only if the fuel amount is less than five units. After leaving the platform, the slow train should go on the slow railway. And the fast train will go on the fast railway also. The scheduling task is to find a feasible path, from waiting for stopping to leaving the platform, in the model to meet the following constraints.

1) Safety Conditions

Two slow trains can stop at different platforms at the same time. One slow train and one fast train can be in slow specific platform and common platform respectively in parallel. There's no more than one train in the same platform at any moment.

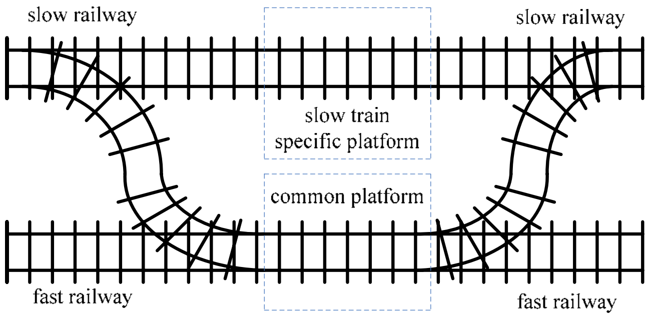

Figure 4. A simple railway station example

2) Time Constraints

The slow train can be in the platform no more than 15 time units. And the fast train can't in the platform more than 8 time units.

3) Resource Constraints

The slow train should consume less than 15 units resource every time it stops at the platform. The fast train should consume no more than 25 units resource during the stopping.

B. *Modeling*

Using the techniques described in Section IV, we can get the system's TCPNIA model, which is shown in Fig. 5. Conflict, parallel, sequence, branch, synchronization, inhibitor, loop, mutual exclusion and communication modeling methodologies described in Section IV are used. The inhibitor arc (p_2, t_7) represents that a fast train has the priority to use the common platform. The detailed processes of modeling are ignored for the space limitation of the paper. The places and transitions are described in Tab. I. A transition may represent a module. A transition's time requirement domain and resource requirement are marked at the side of the transition in Fig. 5. Some resource requirements are not marked in the model, it means that the resource requirements can be ignored during the execution of the corresponding transitions. If we compose several modules, some loop modules will be shown clearly.

At every platform, there is fuel meter checking and fuel adding equipment. It can be abstracted as a common module which is shown in Fig 6.

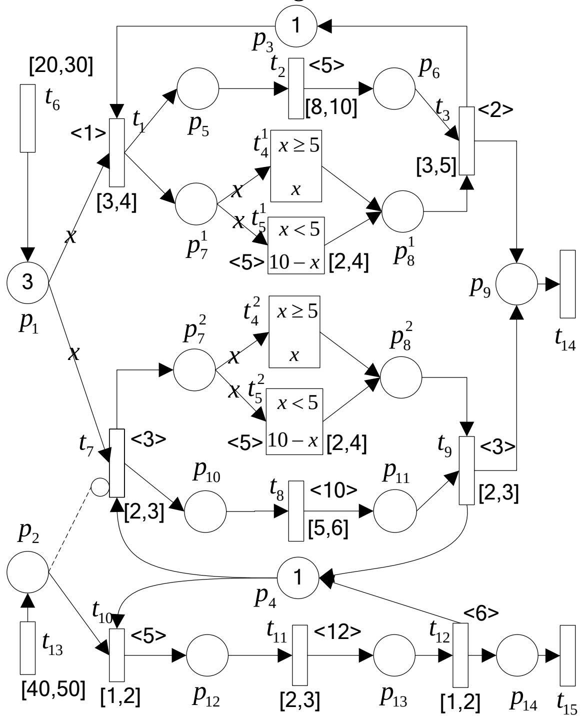

Figure 5. TCPNIA model of a train pulling in system

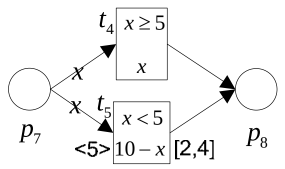

Figure 6. Fuel checking and adding

The safety conditions can be represented by CTL (Computation Tree Logic, CTL) [16] formulas. The temporal logic formulas can be verified using model checking [18] tools. The model checker will return whether CTL formulas are satisfied in the TCPNIA model shown in Fig. 5. We ignore the details of model checking process in this paper.

TABLE I. LEGEND FOR FIG. 5

Place / Transition	Description
p_1	A slow train waiting for pulling in
p_2	A fast train waiting for pulling in
p_3	Slow train specific platform
p_4	Common platform with the fast train having priority
p_5 , p_{10}	Slow train has entered into the platform
p_6 , p_{11}	Slow train has finished passengers up and down
p_7^1 , p_7^2	Slow train is waiting for checking fuel meter
p_8^1 , p_8^2	Slow train has finished fuel adding
p_9	Slow train has left the platform
p_{12}	Fast train has entered into the platform
p_{13}	Fast train has finished passengers up and down
p_{14}	Fast train has left the platform
t_1	Slow train entering the slow train specific platform
t_2 , t_8 , t_{11}	Passengers up and down
t_3 , t_9	Slow train is leaving the platform
t_4^1 , p_4^2	Check the fuel meter, but not add fuel
t_5^1 , t_5^2	Check the fuel meter and add fuel
t_6	A slow train is arriving
t_7	Slow train is entering the common platform
t_{10}	Fast train is entering the common platform
t_{12}	Fast train is leaving the platform
t_{13}	A fast train is arriving
t_{14}	A slow train is going on the slow railway after leaving the platform
t_{15}	A fast train is going on the fast railway after leaving the platform

C. Scheduling

To schedule the situation only a slow train is waiting for stopping, we suppose that the initial marking of the system model is the same as shown in Fig. 5, i.e. $token(M_0, p_1) = 1$, $v(M_0, p_1) = 3$, $token(M_0, p_3) = 1$, $v(M_0, p_3) = 1$, $token(M_0, p_4) = 1$ and $v(M_0, p_4) = 1$. When p_9 has a token, the slow train has finished stopping and leaved the platform. It is denoted as M_e . At that time, $token(M_e, p_3) = 1$, $v(M_e, p_3) = 1$, $token(M_e, p_4) = 1$, $v(M_e, p_4) = 1$, $token(M_e, p_9) = 1$ and $v(M_e, p_9) = 1$. The timestamp of M_0 is set to zero initially. The end state can be denoted as $S_{end} = (M_{end}, TS_{end})$. According to the requirement in Section VI(A), the condition $TS_{end} \le 15$ should be met. The scheduling task is to find a path $\delta = t_{l0} t_{l1} \cdots t_{lm}$, which satisfies $S_0 \xrightarrow{\delta} S_{end}$ and meets the resource constraint.

At such an initial situation, the slow train can stop either platform, as long as the system execution path meets time and resource constraints. We can call the scheduling algorithm in Fig. 3 by calling the function $scheduling(TCPNIA, M_0, M_e, 15,15)$. The algorithm will return false for the resource insufficiency. If we change the resource requirement of t_8 to no more than nine, the algorithm will return the path $t_7 t_8 t_5^2 t_9$, or $t_7 t_5^2 t_8 t_9$ randomly. And $9 \le TS_{end} \le 12$, which means that the time consumption is less than 15 time units. And it meets the limitation of fifteen time units. On the other hand, if we change the time requirements' domains of t_1, t_2 and t_3 to be $[3, 3.5]$, $[8, 8]$ and $[3, 3.5]$ respectively, the algorithm may get the path $t_1 t_2 t_5^1 t_3$ or $t_1 t_5^1 t_2 t_3$ randomly. And it will consume thirteen resource units. At that time $14 \le TS_{end} \le 15$, which also meets the limitation of fifteen time units.

If we change the initial marking to the instance that $token(M_0, p_1) = 0$, $token(M_0, p_2) = 1$, $v(M_0, p_2) = 1$, and the other places' markings are the same as in Fig. 5. The timestamp of M_0 is also set to be zero initially. The target marking M_e satisfies the following conditions: $token(M_e, p_2) = 0$, $token(M_e, p_{14}) = 1$, $v(M_e, p_{14}) = 1$ and the other tokens remain the initial values. According to the requirements in Section VI(A), $TS_{end} \le 8$ should be met.

By calling the function $scheduling(TCPNIA, M_0, M_e, 25, 8)$, the algorithm will return true and give a schedulable path $\delta = t_{10} t_{11} t_{12}$. In the state S_{end} , $0 \le TS_{end} \le 7$. And it consumes 23 units resource. In the path $M_0 t_{10} M_1$, the marking M_1 satisfy the conditions: $token(M_1, p_2) = 0$, $token(M_1, p_{12}) = 1$ and $v(M_1, p_{12}) = 1$. The algorithm can get a schedulable path $\delta_1 = t_{10}$ from the initial marking M_0 to the marking M_1, with 2 time units bound and 5 resource units bound. And it can also get a schedulable path $\delta_2 = t_{11} t_{12}$ from the marking M_1 to the marking M_e, with 5 time units bound and 18 resource units bound. Then the time consuming amount satisfies $2 + 5 = 7 \le 8$, and the resource requirement satisfies $5 + 18 = 23 \le 25$. So the conditions in the Theorem 3 are satisfied. Thus the schedulability of $\delta = t_{10} t_{11} t_{12}$ can be got from the composition of δ_1 and δ_2 .

VII. CONCLUSION

TCPNIA integrates features of colored Petri nets, timed Petri nets and inhibitor arcs. So it can represent multi types of data, control information and resources in a single figure. The temporal relations can also be represented in the figure conveniently. Moreover, analysis methods of traditional Petri nets can also be used in TCPNIA. Modular modeling methods for complicated systems using TCPNIA are proposed. They can reduce the complexity of modeling and enhance the reusability of modules.

To schedule the system described with TCPNIA, a depth-first heuristic search algorithm is proposed. The algorithm has considered not only the factors of resource and time constraints, but also the influences from data operational functions. The influences from data operational functions and system level abstract resource are rarely considered in present literatures. The time requirement domain bounds of a schedulable path can be got directly from the timestamp domain bounds of the end state in the algorithm. Transitions' parallel executions are considered when calculating the time requirement bounds of the schedulable path. The proposed scheduling method can also be used for traditional timed Petri nets model. The case study has shown the feasibility of our method.

We are planning to study automatic task construction method for a real-time embedded system modeled by TCPNIA in the future. Model driven architecture will be used in software development life cycle to enhance systems' reliability, performance, dependability et al. So we will concentrate on model transformations from UML 2.0 to TCPNIA in the future. Thus, the properties described in UML model can be verified and analyzed formally through TCPNIA model.

REFERENCES

[1] T. Murata, "Petri nets: Properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, 1989, pp. 541-580.

[2] L.A. Cortés, P. Eles and Z. Peng, "Modeling and formal verification of embedded systems based on a Petri net representation," *Journal of Systems Architecture*, vol. 49, no. 12-15, 2003, pp. 571-598.

[3] M. Varea, B.M. Al-Hashimi, L.A. Cortés, P. Eles and Z. Peng, "Dual flow nets: Modeling the control/data-flow relation in embedded systems," *ACM Transactions on Embedded Computing Systems*, vol. 5, no. 1, 2006, pp. 54-81.

[4] H.H. Chin, C.-F. Huang and A.A. Jafari, "Implementing timed Petri net in security information systems," *Proc. Thirty-Ninth Southeastern Symposium on System Theory*, IEEE Press, 2007, pp. 214-220.

[5] K. Jensen, L.M. Kristensen and L. Wells, "Coloured Petri nets and CPN tools for modelling and validation of concurrent systems," *International Journal on Software Tools for Technology Transfer*, vol. 9, no. 3, 2007, pp. 213-254.

[6] Y. Nian-hua and Y. Hui-qun, "Modeling and verification of embedded systems using timed colored Petri net with inhibitor arcs," *Journal of East China University of Science and Technology*, vol. 36, no. 3, 2010, pp. 411-417.

[7] Z. Peng and K. Kuchcinski, "Automated transformation of algorithms into register-transfer level implementations," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 13, no. 2, 1994, pp. 150-166.

[8] L.A. Cortés, P. Eles and Z. Peng, "A Petri net based model for heterogeneous embedded systems," *Proc. 17th IEEE NORCHIP Conference*, 1999 of Conference, pp. 248-255.

[9] S. Liu and C. Mu, "An extended Petri net EPRES for embedded system modeling," *Proc. Proceedings of the Fifth IEEE International Symposium on Embedded Computing*, IEEE Computer Society, 2008, pp. 9-13.

[10] E. Tavares, B. Silva, P. Maciel and P. Dallegrave, "Software synthesis for hard real-time embedded systems with energy constraints," *Proc. 20th International Symposium on Computer Architecture and High Performance Computing*, IEEE computer society, 2008, pp. 115-122.

[11] R. Jejurikar and R. Gupta, "Energy-aware task scheduling with task synchronization for embedded real-time systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 6, 2006, pp. 1024-1037.

[12] M. Bertogna, M. Cirinei and G. Lipari, "Schedulability analysis of global scheduling algorithms on multiprocessor platforms," *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, no. 4, 2009, pp. 553-566.

[13] E. Tavares, R. Barreto, P. Maciel, J. Meuse Oliveira, L. Amorim, et al., "Software synthesis for hard real-time embedded systems with multiple processors," *SIGSOFT Software Engineering Notes* vol. 32, no. 2, 2007, pp. 1-10.

[14] D. Xu, X. He and Y. Deng, "Compositional schedulability analysis of real-time systems using time Petri nets," *IEEE Transactions on Software Engineering*, vol. 28, no. 10, 2002, pp. 984-996.

[15] K. Balasubramanian, J. Balasubramanian, J. Parsons, A. Gokhale and D.C. Schmidt, "A platform-independent component modeling language for distributed real-time and embedded systems," *Journal of Computer and System Sciences*, vol. 73, no. 2, 2007, pp. 171-185.

[16] T. Agerwala, *A complete model for representing the coordination of asynchronous processes*, Baltimore: Johns Hopkins University, Hopkins Computer Science Program, Research Report, 1974.

[17] A. Valmari, "The state explosion problem," *Lectures on Petri nets I: Basic models*, Lecture notes in computer science 1491, Springer-Verlag, 1998, pp. 429-528.

[18] E.M. Clarke, O. Grumberg and D.E. Long, "Model checking and abstraction," *ACM Transactions on Programming Languages and Systems*, vol. 16, no. 5, 2002, pp. 1512-1542.