

A Petri Net based Method for Analyzing Schedulability of Distributed Real-time Embedded Systems

Liqiong Chen

Department of Computer Science and Engineering, East China University of Science and Technology, Shanghai, China
Email: qiong_1207@126.com

Zhiqing Shao, Guisheng Fan and Hanhua Ma

Department of Computer Science and Engineering, East China University of Science and Technology, Shanghai, China
Email: zshao@ecust.edu.cn, gs_fan@126.com, mhanhua@163.com

Abstract—As computer systems become increasingly internetworked, a challenging problem faced by researchers and developers of distributed real-time and embedded (DRE) systems is devising and implementing an effective schedulability strategy that can meet real-time requirements in varying operational conditions. In this paper, an extended Place-timed Petri nets (*EPdPN*) is proposed for schedulability analysis in DRE systems. First, we can capture important features of DRE systems and describe them by the semantic model. Second, the key component in DRE systems such as task, the relations between task, communication between module and resource et al. can be modeled by using *EPdPN*. Third, we present the concept of greatest concurrent set and convert schedulability problem into the analysis of state graph by using proposed algorithm, which can work out the feasible solution of scheduling in DRE systems. Finally, a specific example is given to simulate analytical process by using *EPdPN*, the results show that the method can be a good solution to analyze the schedulability of DRE systems.

Index Terms—Distributed real-time and embedded system, Petri nets, schedulability, state graph, communication

I. INTRODUCTION

Schedulability is a key challenge in the analysis of distributed real-time embedded (*DRE*) systems. Major design parameters that influence schedulability include realtime properties, such as task execution times and communication delays [1, 2]. Basically, in a DRE system, if time constraints are not met, the consequences can be disastrous, including great damage of resources or even loss of human lives. For example, a brake-by-wire system in a car failing to react within a given time interval can result in a fatal accident [3].

Therefore it is useful to analyze time constraints of *DRE* systems early in the lifecycle. Despite recent

advances in *DRE* systems development, however, there remain significant challenges that make it hard to develop large-scale *DRE* systems for domains that require hard timing constraints. The key unresolved challenges include the lack of formal methods for effectively modeling, integrating, and verifying.

To address these challenges, We propose an Extended Place-timed Petri Nets (*EPdPN*) model and introduce dual priority to transition. We describe the tasks and the relation between task of DRE systems in detail and convert schedulability problems into analyzing reachability of *EPdPN* model. In particular, we abstract communication process as a non-preemptive task, and using *EPdPN* model to characterize scheduling mechanism and time delay of communication process. Finally, we propose the concept of greatest concurrent set, and give a heuristic algorithm to compute feasible scheduling, the algorithm is realized by constructing part of state graph, thereby reducing the complexity of computation.

The remainder of this paper is organized as follows. Section 2 presents the definition and semantics of *EPdPN* model. Section 3 shows how *EPdPN* can be used to model DRE systems. Section 4 proposes the concept of greatest concurrent set and gives a heuristic algorithm for computing feasible schedule. In section 5, a specific example is given to simulate the modeling and analysis process. Section 6 presents some related works while section 7 is conclusions.

II. COMPUTATION MODEL

Petri net [4] is a mathematical formalism, which allows to model for the features present in most concurrent and real-time systems, such as timing constraints, synchronization mechanisms, and shared resources, etc. According to the features of DRE systems, we will use *EPdPN* to establish a suitable static scheduling model in this paper.

This work was partially supported by the NSF of China under grants No.60373075, the Science-Technology Development Foundation of Shanghai of China under Grant No.06dz15004-1

Definition 1: A Petri net is a 5-tuple, $PN = (P, T, F, W, M_0)$, where [4]:

- (1) $P = \{p_1, p_2, \dots, p_n\}$ is a finite set of places, $n \geq 0$;
- (2) $T = \{t_1, t_2, \dots, t_m\}$ is a finite set of transition, $m \geq 0$ and $P \cup T \neq \emptyset, P \cap T = \emptyset$;
- (3) $F \subseteq (P \times T) \cup (T \times P)$ is a finite set of directed arcs;
- (4) $W : F \rightarrow N^*, N^*$ is a set of non-negative integer, which is called the weight of arc;
- (5) $M_0 : P \rightarrow N^*, M_0$ is the initial marking, which represents the initial state of system.

Definition 2: A 3-tuple $\Sigma = (PN; C, Pr)$ is called Extended Place-timed Petri nets (EPdPN) model if:

- (1) PN is a Petri net, which is called base Net of Σ ;
- (2) $C: P \rightarrow N^*$ is time function of place, $C(P_j) = c_j$ represents the token in P_j can not be used during the time from reaching the place to c_j ; C is called delay time of place.
- (3) $Pr: T \rightarrow (N^* \times N^*)$ is priority function of transition, $Pr(t_i) = (\alpha_i, \beta_i)$, and α_i, β_i are called primary and secondary priority of t_i .

In this paper, the firing of transition in EPdPN model is instantaneous and the transitions called are determined by its priority. By default, the delay time of place is 0; the priority of transition is (0,0). The time units can be set according to specific circumstances.

In the modeling for DRE systems, its function is composed by a serious of interconnected tasks, which are mapping into transition, and the message transfer between tasks are characterized by place and its token. The distribution of token in each place at Σ is called the marking of EPdPN, denoted by M . The marking $M(p)$ denotes the number of tokens in the place p . $M = M^a \cup M^u$, where M^a is the available tokens of M , M^u is the not available tokens of M ; for any $x \in (P \cup T)$, we denote the pre-set of x as $\bullet x = \{y | y \in (P \cup T) \wedge (y, x) \in F\}$ and the post-set of x as $x \bullet = \{y | y \in (P \cup T) \wedge (x, y) \in F\}$.

Because the tokens in EPdPN model including time factor, only using marking can't describe the state of model. In order to better characterize time properties of model, we introduce the concept of wait time to describe the state of EPdPN.

Definition 3: Let EPdPN reaches marking M at time θ , the place P_i has j tokens in marking M , P_i^k is the k th token of P_i . Vector $TS(P_i) = (TS_i^1, TS_i^2, \dots, TS_i^j)$ is the wait time of place P_i , where $TS(P_i^k) = \max\{c_i - (\theta - \xi_k), 0\}$, ξ_k is the time that token P_i^k generated. TS_i^k is the wait time of token P_i^k .

$TS_i^k = m$ presents that the system must wait m time units before using token P_i^k . $TS(P_i^k) = 0$ represents the token is available. Recorded $TS(M)$ as wait time set of places under marking M .

Definition 4: A pair $S = (M, TS)$ is called a state of Σ . Where M is marking, which describes the distribution of resources; $TS(M)$ is the time stamp of marking M , which depicts time properties of system.

Initial state $S_0 = (M_0, TS_0)$ where TS_0 is a zero vector, i.e., all tokens are available in the initial state. Two ways can be used to change state:

(1) time elapse, at the time $\theta + \omega$ (θ is the time reach the state S and $\omega > 0$), because the wait time of tokens have changed which makes the model reach a new state S' , denoted by $S[\omega > S']$.

(2) transition firing, the firing of transition t_i will generate a new marking, thus the model will reach a new state S' , denoted by $S[t_i > S']$.

$S[\omega > S'$ and $S[t_i > S''$ are denoted by $S[(t_i, \omega) > S''$.

Definition 5: Let $\Sigma = (PN; C, Pr)$ is an EPdPN model, for transition $t_i \in T$, if:

(1) $\forall p_j \in P: p_j \in \bullet t_i \rightarrow M^a \geq W(p_j, t_i)$, t_i is called strong enabled under marking M , denoted by $M[t_i >$, all transitions that have strong enabled under marking M are recorded as set $SET(M)$.

(2) $\forall p_j \in P: p_j \in \bullet t_i \rightarrow M \geq W(p_j, t_i) \wedge M^a < W(p_j, t_i)$, t_i is called weak enabled under marking M , denoted by $M[t_i \geq$, all transitions that have weak enabled under marking M are recorded as set $WET(M)$.

If transition t_i has weak enabled under marking M and at least pass through ω time units to be strong enabled, then ω is called the firing delay of transition t_i under marking M .

Definition 6: Let $\Sigma = (PN; C, Pr)$ is an EPdPN model, for transition $t_i \in T$, the firing of transition t_i is called effective firing iff it meets one of the following conditions:

(1) $t_i \in SET(M): \alpha_i \leq \min(\alpha_j) \wedge \beta_i \leq \min(\beta_k)$ where $t_j \in SET(M), t_k \in U(t_i)$

(2) $t_i \in WET(M): SET(M) = \emptyset \wedge \max(C(\bullet t_i)) \leq \max(C(\bullet t_j))$, $t_j \in WET(M)$

All the effective firing transitions under the state S are denoted by $FT(S)$.

Definition 7: A pair $S = (M, TS)$ is the state of Σ , at the time $\theta + \omega$ (θ is the time at reaching the state S , ω is the firing delay of transition t_i), the model will reach a new state $S' = (M', TS')$ by effectively firing transition t_i , denoted by $S[(t_i, \omega) > S'$, S' is called the reachable state of S , the computation of M', TS' are based on the following rules:

(1) Computing marking:

$$\forall P_j \in \bullet t_i \cup t_i \bullet, M'(P_j) = M(P_j) - W(P_j, t_i) + W(t_i, P_j)$$

(2) Computing wait time:

First, adding time stamp to new generated marking: $TS'(P_i^k) = c_i$, P_i^k is generated when firing transition t_i ;

Second, modifying the time stamp of tokens which are generated before the firing of transition t_i : $TS'(P_i^k) = \max\{(TS(P_i^k) - \omega), 0\}, TS(P_i^k) \geq 0$.

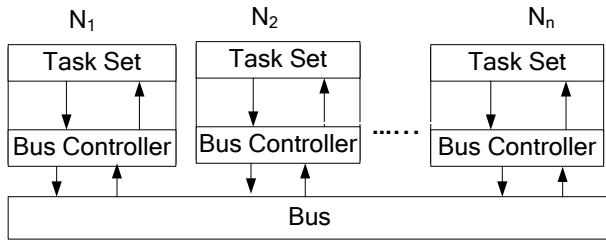
The computation of time stamp is mainly based on the generated time of token: for the newly generated tokens, the wait time is equal to the delay time of the corresponding place; if the token is generated before the firing of transition and not be removed; the corresponding wait time needs to be adjusted.

III. MODEL CONSTRUCTION

In this section, we will use EPdPN model to describe tasks, resource and communication mechanism of the DRE system, thus forming the model of the whole application.

A . DRE System Model

DRE systems can be regarded as a number of modules; each module also contains a number of partially ordered, serial or parallel implemented sub tasks [5, 6]. The function of DRE systems will be distributed to a number of interrelated embedded devices, each device is responsible for certain functions, and has certain autonomy, but relies on the computation of other embedded devices. The structural model of DRE system is shown in Fig. 1:



The EPdPN Model of Task TK_i

Among them, DRE system has n tasks; each task is composed by a series of interrelated sub tasks set and a bus controller. The effective and reliable communication between tasks is done by bus and bus controller. The communication process can follow different protocols, but we don't focus on modeling communication protocol in this paper, therefore, we can assume the communication process between tasks is [7]: the task will send message to the cache of bus controller, and assign a priority to each message, the bus always send message which has high-priority; the system use nondestructive bus arbitration, when there are two nodes in the bus send message to the network at the same time, the low priority nodes will initiate to stop sending messages, while high-priority nodes can continue to send message. We can give following definitions based on the features of DRE systems:

Definition 8: DRE system model is a 7-tuple $\Omega = \{TK, N, RS, RL, TC, RT, SP\}$:

- (1) TK is a finite tasks set;
- (2) $N = \{N_1, N_2, \dots, N_n\}$ is the collection of tasks set, N_i is the corresponding task set of module i ;
- (3) RS is a finite resources set;
- (4) RL is the relation between tasks, which may be sequence, choice, parallel et al;
- (5) $TC: TK \rightarrow (N^* \times N^* \times N^*)$ is the time function of task, $TC(TK_i) = (r_i, ec_i, d_i)$, where r_i, ec_i, d_i represent the release time, running time and deadline of task respectively.
- (6) $RT: TK \rightarrow RS^*$ is the resources function, whose function is to assign necessary resources to each tasks, RS^* represents the multiple sets of resources, that is, a task can use multiple sources;
- (7) $SP: TK \rightarrow N^*$ is the static priority of task, $SP(TK_i) = SP_i$.

In this paper, we assume the tasks in a DRE system have the following characteristics:

- (1) The task has time constraints including release time, running time and deadline.
- (2) Static priority schedule is adopted to realize preemption.
- (3) A task may also need other resources in addition to processor, such as variables or buffer; meanwhile each task has two ways to access the resources: exclusive access and sharing access.
- (4) Each task can not suspend itself before completion.
- (5) The overhead of task switching can be neglected.

B. Modeling EPdPN

On the above DRE system model, we will abstract and model for tasks, communication between modules and resource, and forming the whole application according to the relation between tasks.

(1) Modeling Task TK_i

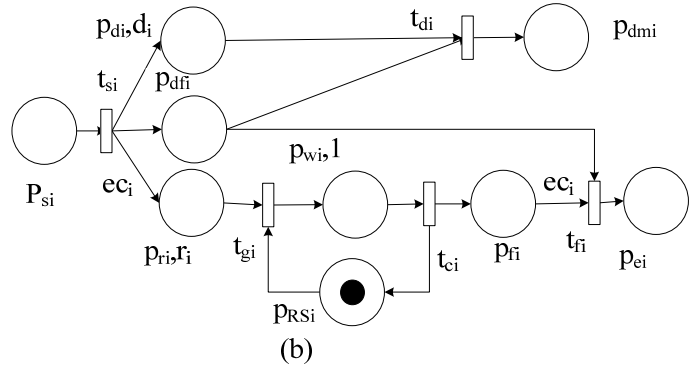
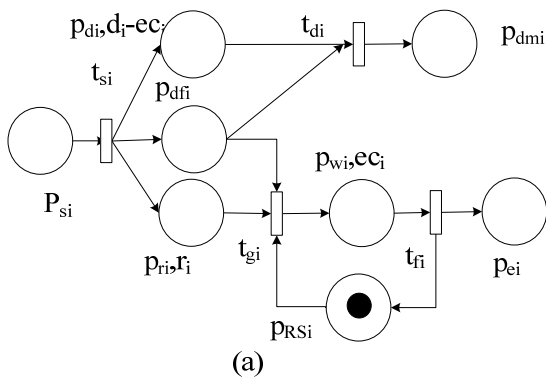
According to the ways that task accesses resources, it can be divided into preemptive task and non-preemptive task. Preemptive task refers that the task can be interrupted by high-priority task during operating, while non-preemptive task refers that the task can not be interrupted by other tasks once operating; they must wait until the task automatic release resources after completion. Below we will model two types of task respectively.

Let the EPdPN model of task TK_i is $\Sigma_i = (PN_i; C_i, Pr_i)$, where $PN_i = (P_i, T_i, F_i, W_i, M_{0i})$. In the modeling of non-preemptive task (Fig.2(a)): $P_i = \{P_{si}, P_{dfi}, P_{di}, P_{dmi}, P_{ri}, P_{wi}, P_{ei}, P_{RSi}\}$, $T_i = \{t_{si}, t_{di}, t_{gi}, t_{ci}\}$. $c_{di} = d_i - ec_i$ explains if the time to deadline is less than the operation time of task, the overtime operation (t_{di}) will be executed. $c_{wi} = ec_i$ explains the operation of task requires ec_i time units. In the modeling of preemptive task (Fig.2(b)): $P_i = \{P_{si}, P_{dfi}, P_{di}, P_{dmi}, P_{ri}, P_{wi}, P_{fi}, P_{ei}, P_{RSi}\}$, $T_i = \{t_{si}, t_{di}, t_{gi}, t_{ci}, t_{fi}\}$. If task TK_i has executed one time unit and high-priority task also competes its resources, TK_i will release resources. The system will terminate and output the result after executing ec_i time units accumulatively. Table I lists the actual meaning of these transition and place. Among them, Place P_{RSi} is modeling for reusable resources that task TK_i required during processing, which can be added based on actual requirement. In order to simplify the model, we will model the preemptive task whose priority is (0,0) as non-preemptive task.

Based on the EPdPN model of task, we can allocate priority to the corresponding transition. In this paper, the allocation rules of priority are:

- 1) The primary priority of all transitions in Σ_i is equivalent to the priority of the corresponding task;
- 2) The secondary priority is divided into seven levels based on its importance: $\beta_{ri} = \beta_{ci} = 0, \beta_{si} = 1, \beta_{gi} = 4, \beta_{di} = 6$.

The purpose of secondary priority is to balance the firing of each transition by considering that they may have different importance. From rule 2), we can draw that the level of normal termination operation and running operation are highest, while the level of overtime processing is lowest. The other three levels are used in the communication process, we will introduce them in the next section.



(2) Modeling Basic Relations

Assuming each task of DRE system has constructed the corresponding EPdPN model which is shown in Fig. 2. We will compose EPdPN model of each task by their relations. Suppose the EPdPN model of task TK_i and TK_j are Σ_i and Σ_j , the composed EPdPN model is $\Sigma = (PN; C, Pr)$, where $PN = (P, T, F, W, M_0)$. According to the basic relation between tasks (sequence, choice, concurrency), we can have the following mapping:

If the relation between task TK_i and TK_j is sequence, which means TK_j can be fired only after executing task TK_i , then we can add a place to connect the termination operation of task TK_i and access resources operation of task TK_j , which is shown in Fig. 3(a).

If the relation between task TK_i and TK_j is choice, which means only one task can be chosen to fire. As shown in Fig. 3(b), P_{sij} is the common condition of TK_i and TK_j , while P_{eij} is the output.

If the relation between TK_i and TK_j is parallel. The corresponding EPdPN model is shown in Fig. 3(c), where the function of P_{sij} , t_{sij} , t_{eij} and P_{eij} is to control the parallel operation of tasks.

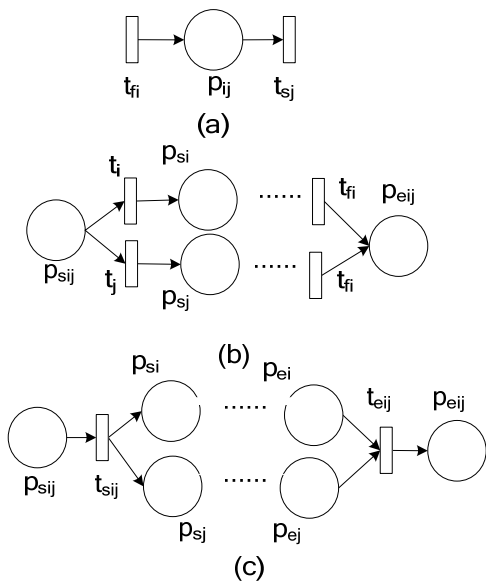


Figure 3. The EPdPN Model of Basic Relations

(3) Modeling Communication Process

In this paper, we abstract the communication process between modules as a communication task TK^{ft} , which is a 3-tuple (TK_f, TK_r, sm) , where TK_f, TK_r, sm represent the message sender, receiver and required time respectively. We can construct the corresponding EPdPN model of task TK^{ft} , which is shown in Fig. 4. $\Sigma = (PN; C, Pr)$, $PN = (P, T, F, W, M_0)$ where $P_i = \{P_{bi}, P_{wgbi}, P_{cij}, P_{bus}, P_{bj}\}$, $T_i = \{t_{wbi}, t_{gbi}, t_{cij}\}$. Let $c_{cij} = sm_{ij}$ and the delay time of rest places is 0. P_{bus} is modeling for bus; P_{wgbi} represents the state that is waiting for getting tokens, while P_{cij} is the communication state. The primary priority of transition $t_{wbi}, t_{gbi}, t_{cij}$ are equal to the priority of receiver while the secondary priority is: 2, 3, 5.

(4) Modeling Resource

In this paper, we assume the model doesn't have memory limit when the tasks are executed, so memory is not considered as resource. For the sharing resources such as cache, processor, bus, and so on, we establish a place P_{RS} . If the number of sharing resource is n , then we set $M_0(P_{RS}) = n$ in the initial marking. For example, if task TK_i need call the number of resources RS is w , we may set $\bullet t_{gi} = \bullet t_{gi} \cup \{P_{RS}\}$, $\bullet t_{ci} = \bullet t_{ci} \cup \{P_{RS}\}$, $W(P_{RS}, t_{gi}) = W(P_{RS}, t_{ci}) = w$.

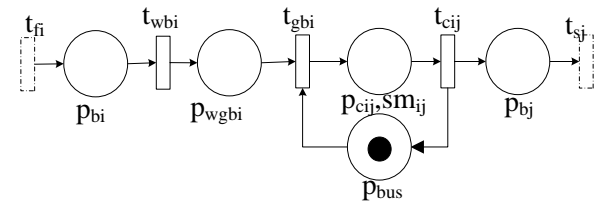


Figure 4. The EPdPN Model of Communication Process

(5) Forming the Whole Application

The main purpose of this section is to form the EPdPN model of whole application based on the above model. The corresponding Model is shown in Fig. 5. First, we introduce the initial place P_s and transition t_s which represent the beginning operation of the whole application, and $\bullet t_s = \{P_s\}$, $t_s \bullet = \{P_{si} | P_{si} = \emptyset, i=1, 2, \dots, n\}$, $\bullet P_s = \emptyset$, $P_s \bullet = \{t_s\}$, $M_0(p_s) = 1$; Second, we introduce the termination place P_e and transition t_e which represent the termination operation of whole application, and $\bullet t_e = \{P_{ei} | P_{ei} = \emptyset, i=1, 2, \dots, n\}$, $t_e \bullet = \{P_e\}$, $\bullet P_e = \{t_s\}$, $P_e \bullet = \emptyset$. Thus forming the EPdPN model of the whole application, which will lay theory foundation for the analysis of model properties.

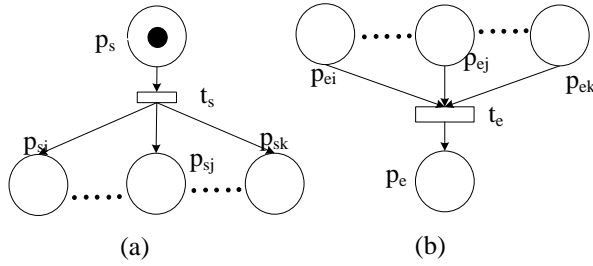


Figure 5. The EPdPN Model of Start and End

IV. MODEL ANALYSIS

Schedulability is an important characteristic for guarantying the reliable of DRE systems. We first give the method for merging the firing of transition, and computing a feasible schedule of system.

In this section, EPdPN model is starting from initial state S_0 and will generate new state through effectively firing enabled transitions, thus establishing a state space (known as state graph). The construction of EPdPN's state graph is to outline the different firing sequences of transition, which make the complexity of computation exponential growth with the number of transitions increasing. Therefore, the concept of greatest concurrent set is introduced in this paper.

Definition 9: In the state $S = (M, TS)$, transition $t_i, t_j \in FT(S)$ are called parallel, if:

- (1) $S[t_i > S' \rightarrow t_j \in FT(S')$;
- (2) $S[t_j > S'' \rightarrow FT(S'')$

Which is denoted as $t_i \delta t_j$. Otherwise, we call transition t_i, t_j are conflict, denoted by $t_i \otimes t_j$.

The firing of transition in the greatest concurrent set can not affect the firing of other transitions, denoted by $S[H > S']$. Let the number of transitions in H be n . We must compute $n!$ states by using traditional analysis methods, however, we only need compute $n-1$ states by using the greatest concurrent set. So we can reduce the complexity of computation by using the greatest concurrent set.

We will introduce several special states before analyzing the schedulability of EPdPN model. Let $\Sigma = (PN; C, Pr)$ is an EPdPN model, $S = (M, TS)$ is a state of Σ :

- (1) if $\exists p_{dmi} \in P$ which makes $M(p_{dmi})=1$, then S is called overtime state;
- (2) if $\exists t_{di} \in FT(S)$, then S is called dangerous state;
- (3) if $M(p_e)=1$, then S is called normal termination state, denoted by S^F .

Overtime state means that the operation can not be completed before the deadline, the model will reach overtime state when starts from the dangerous state; while the normal termination state is the state that all tasks have completed before the deadline; the other states in the system are called normal state.

Definition 11: Let the EPdPN model of Ω is $\Sigma = (PN; C, Pr)$, then Ω is schedulable iff S^F is reachable in Σ .

The model is schedulable if all tasks can complete before their deadlines, which is called a feasible schedule.

The feasible schedule is corresponding to a path from state S_0 to S^F in Σ .

TABLE I. COMPUTING THE FEASIBLE SCHEDULING

Input: a bounded EPdPN $\Sigma = (PN; C, Pr)$
Output: the feasible scheduling for service composition
Schedule_Computer(S)
1: marking S
2: If (S == S^F) then
3: return FS;
4: Else if (FT(S) == \emptyset) then
5: return false;
6: Else if all the child nodes of S have been marked then
7: top(State);
8: If $\exists t_{gi} \in FT(S)$ then
9: Top(FS);
10: ST = ST - ω ;
11: $S_{new} = Gettop(State)$;
12: End If
13: If $S_{new} = S_0$ then
14: return false;
15: Else
16: Update_State(S_{new});
17: End If
18: Else Update_State(S);
19: End If
Update_State(S) //Update feasible scheduling
20: calculate the greatest parallel set of S;
21: Seq_Computer(S, FT(S), H);
22: ST = ST + ω ;
23: If $\exists t_{gi} \in \delta$ then Push(FS, (Task _i , ..., Task _j , ST));
24: Push(State, S_{new});
25: Sch_Computer(S_{new});
26: End If
Seq_Computer(S, F, H)
26: If F = H then $\delta = H$;
27: Else optional choose a transition t_i
28: $\delta = H \cup \{t_i\}$;
29: $F = F - t_i$;
30: $S_{new} = S[(\delta, \omega) >]$;
31: End If
32: If S_{new} has been marked then
33: Seq_Computer(S, F, H);
34: Else if $\exists t_{di} \in FT(S_{new})$ then
35: marked S_{new} ;
36: Seq_Computer(S, F, H);
37: Else return S_{new}, δ, ω ;
38: End If

Therefore, we can get the necessary part of state graph based on depth-first-search algorithm, thus getting the path. As the path is got from part of reachable graph, it may not be optimal.

The algorithm is based on the state graph of EPdPN model, which takes the initial state as root node, and

gradually computing every state in feasible schedule, we can do following operations for the current state S :

Step 1: If S is the normal termination state, then outputting the feasible schedule, otherwise go to Step 2;

Step 2: Computing firing set $FT(S)$ of S , if $FT(S)$ is empty set, then outputting error info, otherwise go to Step 3;

Step 3: If S is a dangerous state, then stepping back and updating feasible schedule;

Step 4: If S is a normal state, then computing the new state and continue to judge its state.

The algorithm is composed by three functions: function $Compute_Schedule(S)$ is to determine to output or continuously operate based on current state; function $Update_State(S)$ is to update the feasible schedule; while function $Compute_Sequce(S,F,H)$ is to compute next state. In order to get the result as early as possible in limited time, we must assure the model does not exist deadlock and loop. These two problems can be solved by modeling process in this paper. First, we assume tasks can be fired after obtaining all necessary resources, and it will release resource once completing(non-preemptive task) or running a time unit (preemptive task), thus breaking one of the necessary conditions of deadlock generated. While we don't consider task that has no time limit in modeling process, so there is no loop in this paper. In summary, the algorithm can be finished in limited time.

V. EXAMPLE

In this section we illustrate the feasibility of analysis process by an example. Let $\Omega=\{TK,N,RS,RL,TC,RT,SP\}$, the system model as show in Fig. 6. where $N_1 = TK_0, TK_1, TK_2, TK_3; N_2 = TK_4, TK_5$, the special constrains are shown in Table II.

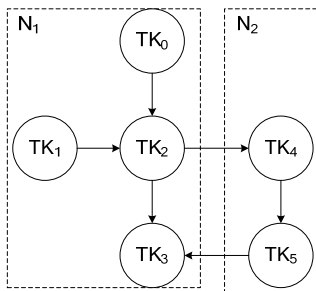


Figure 6. The EPDPN Model of Start and End

TABLE II. COMPUTING THE FEASIBLE SCHEDULING

Task	Running Time	Deadline	Priority
TK_0	2	6	1
TK_1	3	8	0
TK_2	4	12	0
TK_3	8	18	0
TK_4	4	6	0
TK_5	2	20	0
Sequence	$(TK_1,TK_2),(TK_2, TK_3),(TK_0, TK_3),(TK_4,TK_5)$		
Communication	$(TK_4,TK_2)=1, (TK_3,TK_5)=1$		
Preemptive	(TK_0,TK_1)		

$r_1 = I$, the release time of other tasks is 0. Task TK_1 and TK_2 are competing for sharing resource RS_{01} . Because the priority of task TK_1 is 0, we can regard it as non-preemptive task during the process of modeling.

Among them, the communication refers the message sending between the modules. As the tasks in this system are relatively small, we can assume that task will idle once release the processor. So we could not consider the processor in the modeling of resource. Using the above modeling methods, we can construct the EPDPN model of Ω , which is shown in Fig. 7.

We can get a feasible schedule of Ω by using feasible

TABLE III. COMPUTING STEP

Step	S	FS	ST	S_{new}
1	S_0	Null	0	$S_1=S_0[t_s >$
2	S_1	$(Task_0,Task_4,0)$	0	$S_2=S_1[(t_{g0},t_{g4})>$
3	S_2		1	$S_3=S_2[(t_{c0},1)>$
4	S_3	$(Task_1,1)$	1	$S_4=S_3[t_{g1}>$
5	S_4		4	$S_5=S_4[(t_{c1},t_{f4},3)>$
6	S_5	$(Task_0,Task^{42},4)$	4	$S_6=S_5[(t_{g0},t_{gb4})>$
7	S_6		5	$S_7=S_6[(t_{c0},t_{c42},I)>$
8	S_7	$(Task_2,5)$	5	$S_8=S_7[(t_{f0},t_{g2})>$
9	S_8		9	$S_9=S_8[(t_{f2},4)>$
10	S_9	$(Task_3,9)$	9	$S_{10}=S_9[t_{g3}>$
11	S_{10}		17	$S_{11}=S_{10}[(t_{f3},8)>$
12	S_{11}	$(Task^{35},17)$	17	$S_{12}=S_{11}[t_{gb3}>$
13	S_{12}		18	$S_{13}=S_{12}[(t_{c35},I)>$
14	S_{13}	$(Task_5,18)$	18	$S_{14}=S_{13}[t_{g5}>$
15	S_{14}		20	$S_{end}=S_{14}[(t_e,2)>$
16	S_{end}			

scheduling algorithm of Table I, the feasible scheduling computation steps of Fig. 7 are shown in Table III.

S, FS, ST, S_{new} represent current state, global time, new step of feasible schedule and new state respectively. We can draw that the model is scheduling by analyzing Table III, which is: $(TK_0, TK_4, 0) \rightarrow (TK_1, 1) \rightarrow (TK_0, TK^{42}, 4) \rightarrow (TK_2, 5) \rightarrow (TK_3, 9) \rightarrow (TK^{35}, 17) \rightarrow (TK_5, 18)$.

Due to space constraints, we only simulate for a simple example, but it sufficient to explain the accuracy and effectiveness of our proposed method for modeling and analyzing DRE systems. From the simulation process, we can get that EPDPN model can clearly express each component of DRE system. If we use general method for computing, the output of such a possible scheduling needs 34 steps, but here only using 15 steps, which explains the proposed heuristics algorithm in this paper can output results in limited time.

VI. RELATED WORKS

In recent year, there has been some related works used for DRE systems design including the non-formal methods and formal methods. The followings are related

non-formal methods: In [8, 9, 10], the authors proposed a global framework to support Distributed Real-time and Embedded application development based on methods,

verification tools to verify the properties of system. However, comparing with other formal methods, the VDM may be more difficult to understand and grasp for

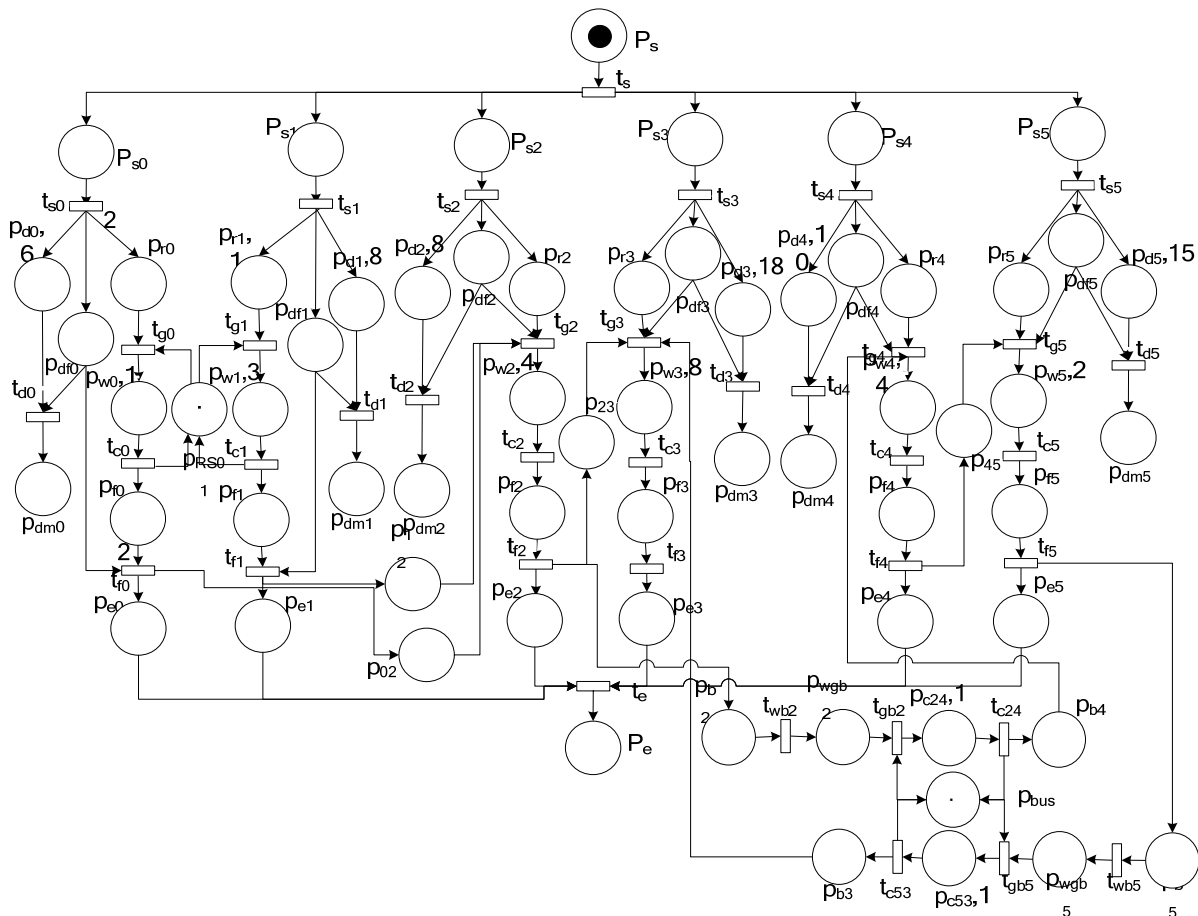


Figure 7. The EPdPN Model Σ of Ω

such as pattern, component middleware and model-driven. And analyzed quality of service of Distributed Real-time and Embedded systems (energy, fault-tolerate, scheduling etc). But these methods only focus on high-level system, and lack the formal semantic of model. P. Paul et al [11, 12] used task graph to describe tasks and relations between tasks in *DRE* systems. More specifically, it discussed the schedulability analysis of *DRE* systems, and introduced several design optimization problems characteristic of this class of systems. However, the task graph lacks of rigorous mathematical foundation, so that the specification of its systems may contain ambiguous, vague, contradictory description of the requirement. Although the non-formal methods has made some corresponding results in the design of *DRE* systems, but they may cause some of the semantic ambiguity, in order to solve this problem, there has been some formal methods: G. Madl et al [13, 14] used Time Automata (*TA*) to model various components of the non-preemptive real-time distributed embedded systems and converted the system scheduling problem into *TA* state reachability. As the *TA* model implied the existence of global clock, it unfits for modeling the Distributed Systems. In [15, 16] used the time extended of Vienna Development Method (*VDM++*) to stipulate *DRE* systems, and used *VDM*

developers. A resource-based time Petri Net is proposed in [17] to model the *DRE* systems and analyzed the corresponding semantic, properties. But it didn't describe the communication between modules which is the key issue of *DRE* systems.

VII. CONCLUSIONS

The main contributions of this paper are: (1) attributing to the transition of Place-timed Petri Net to better describe the schedule strategy of the *DRE* systems by adding static priority; (2) summarizing the modeling steps of *DRE* systems in detail; (3) describing the characteristics against the EPdPN model and proposing the greatest concurrent set of the state, thus reducing the complexity of computation. Using this method for modeling and analyzing *DRE* systems has the following advantages: (1) with modular functionality and a high degree of reusability; (2) with a rigorous mathematical foundation, which can be easily used to analyze and verify the established model.

The study of *DRE* systems is still underway at present, the following two aspects are the main work in the next phase: (1) further improving this method, and considering the fault-tolerant of each task to guarantee the more

soundness of schedulability; (2) developing the corresponding tools to support modeling.

ACKNOWLEDGMENT

The research described here was partially supported by the NSF of China under grants No.60373075, the Science-Technology Development Foundation of Shanghai of China under Grant No.06dz15004-1.

REFERENCES

- [1] X. Wang, Y. Chen, C. Lu, and X. Koutsoukos. "Fc-orb: A robust distributed real-time embedded middleware with end-to-end utilization control." *Journal of Systems and Software*, 2007, 80(7). pp. 938–950.
- [2] E. P. Freitas, M. A. Wehrmeister, C. E. Pereira, F. R. Wagner, E. T. Silva and F. C. Carvalho, "Using Aspect-Oriented Concepts in the Requirements Analysis of Distributed Real-Time Embedded Systems," IFIP International Federation for Information Processing, Embedded System Design: Topics, Techniques and Trends. Boston, Springer.vol. 231, pp. 221-230, 2007.
- [3] E. P. Carlos and C. Luigi, "Distributed real-time embedded systems: Recent advances, future trends and their impact on manufacturing plant control," *Annual Reviews in Control*, 2007, 31(1). pp. 81–92.
- [4] T. Murata. Petri nets: Properties, analysis and application. In *Proceedings of the IEEE*, volume 77, 1989, pp. 541–580.
- [5] 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*, 2007, 73(2). pp.171–185.
- [6] Z. Ying, D. TRobert, and C. Krishnendu, "Energy-aware deterministic fault tolerance in distributed real-time embedded systems," In *Proceedings of the 41st annual conference on Design automation*. New York, ACM. pp. 550–555, 2004.
- [7] G. Trombetti, A. Gokhale, D. Schmidt, J. Greenwald, J. Hatcliff, G. Jung and G. Singh, "An Integrated Model-Driven Development Environment for Composing and Validating Distributed Real-Time and Embedded Systems," *Model-Driven Software Development*. Berlin Heidelberg, Springer. pp. 329-361.2005.
- [8] W. Bin, W. Zhaohui, and C. Wenzhi, "Component model optimization for distributed real-time embedded software," *IEEE International Conference on Systems, Man and Cybernetics*, IEEE Computer Society. Washington. Vol.2. pp. 1158–1163, 2004.
- [9] O. S. Unsal, I. Koren and C. M. Krishna, "Power-Aware Replication of Data Structures in Distributed Embedded Real-Time Systems," *Lecture Notes in Computer Science, Parallel and Distributed Processing*. Berlin Heidelberg, Springer. vol. 1800. pp. 839-846. 2000.
- [10] L. Patrick, B. Jaiganesh, D. C. Schmidt, T. Gautam, G. Aniruddha, and D. Thomas, "A multi-layered resource management framework for dynamic resource management in enterprise dre systems," *Journal of Systems and Software*. 2007, 80(7). pp. 984–996,
- [11] P. Paul, H. P. Kare, I. Viacheslav, and E. Petru, "Scheduling and voltage scaling for energy/reliability trade-offs in faulttolerant time-triggered embedded systems," In *Proceedings of the 5th IEEE/ACM international conference on Hardware/software codesign and system synthesis*. New York, ACM. pp. 233–238, 2007.
- [12] P. Paul, E. Petru, P. Zebo, and P. Traian, "Analysis and optimization of distributed real-time embedded systems," *ACM Transactions on Design Automation of Electronic Systems*. 2006, 11(3). pp. 593–625,
- [13] G. Madl, N. Dutt and S. Abdelwahed, "Performance estimation of distributed real-time embedded systems by discrete event simulations," In *Proceedings of the 7th ACM /IEEE international conference on Embedded software*. New York, ACM. pp. 183–192, 2007.
- [14] G. Madl, S. Abdelwahed and D. C. Schmidt, "Verifying Distributed Real-time Properties of Embedded Systems via Graph Transformations and Model Checking," *Real-Time Systems*. vol. 33. pp. 77–100. 2006.
- [15] V. Marcel, G. L. Peter, and H. Jozef, "Modeling and validating distributed embedded real-time systems with vdm++," In *Proceedings of Formal Methods*. Heidelberg, Springer-Verlag. pp. 147-162. 2006.
- [16] M.Verhoef, and Larsen, "P.G.: Enhancing VDM++ for Modeling Distributed Embedded Realtime Systems.," *Technical Report*, Radboud University Nijmegen (March 2006) A preliminary version of this report is available on-line at <http://www.cs.ru.nl/~marcelv/vdm/>.
- [17] Z. Haitao and A. YunFeng, "Time analysis of scheduling sequences based on petri nets for distributed real-time embedded systems," In *Proceedings of the 2nd IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications* Washington. IEEE Computer Society. pp. 1–5. 2006

Liqiong Chen. She was born in 1982, Ph. D. candidate. Her research interests include distributed computing, embedded systems and formal methods.

Zhiqiong Shao. He was born in 1967, professor, Ph. D. supervisor, IEEE senior member, ACM member, China Computer Federation senior member. His research interests include software engineering, information security and formal methods.

Guisheng Fan. He was born in 1980, Ph. D. candidate. His research interests include service oriented computing, distributed computing and formal methods.

Xiuying Wang. She was born in 1970, Ph. D. candidate. His research interests include service oriented computing, distributed computing and formal methods.