An Approach to Formally Modeling and Verifying Distributed Real-time Embedded Software

Liqiong Chen
Department of Computer Science and Information Engineering, Shanghai Institute of Technology, Shanghai, China
Email: lqchen@sit.edu.cn

Guisheng Fan[1,2] and Yunxiang Liu[3]
[1]Department of Computer Science and Engineering, East China University of Science and Technology, Shanghai, China
[2] Shanghai Key Laboratory of Computer Software Evaluating and Testing, Shanghai, China
[3] Department of Computer Science and Information Engineering, Shanghai Institute of Technology, Shanghai, China
Email: gsfan@ecust.edu.cn, yxliu@sit.edu.cn

***Abstract*—As computer systems become increasingly inter-networked, Distributed Real-time Embedded (DRE) systems has became increasingly common and important, a challenging problem faced by researchers and developers of DRE software is devising and implementing an method that can effectively analyze requirements in varying operational conditions. In this paper, a Hierarchical Distributed Real-time Embedded net (HDRE-net) is proposed as software analysis tool. The basic task, function module and communication process are modeled by using HDRE-net, thus forming the whole application through the synthesis operation of Petri net. Time Reachability Graph is adopted to analyze the correctness of HDRE-net, the basic properties of DRE software are also considered. Finally, a specific example is given to simulate the analysis process, and the results show that the method can be a good solution to analyze DRE software.**

***Index Terms*—Distributed real-time and embedded system; Petri net; Time Reachability Graph; modeling; verifying**

I. INTRODUCTION

As computer systems become increasingly inter-networked, most of critical systems in the world are embedded systems that control physical, chemical, biological, or defense processes and devices in real-time. Increasingly, these embedded systems are part of larger DRE system [1]. A typical DRE software will consist of multiple subsystems, which may be concentrated in a highly localized area, distributed over a wide geographic region, or may involve combinations of both local and distributed deployment. These subsystems will communicate with each other to exchange information and carry out coordinated actions [2].

Because DRE software is often characterized by complexity and volatility of requirements, developers require tools that support the rapid evaluation of design models against system-level temporal and functional properties. Such a validation activity helps identify requirement. However, a challenging problem faced by researchers and developers of DRE software is how to devise and implement a method that can effectively analyze requirements in varying operational conditions. Basically, in a DRE system, if the basic properties are not met, the consequences can be disastrous, including great damage of resources or even loss of human lives [3].

Therefore it is necessary to model and analyze DRE software early in the lifecycle. In these early development phases, the cost effectiveness and ease of use of validation tools is significant, as well as the level of rigor supplied by the modeling language and environment. Despite recent advances in DRE systems development, however, there remain significant challenges that make it hard to develop large-scale DRE software [4, 5]. The key unresolved challenges include the lack of methods for effectively modeling, integrating, and verifying.

To address these challenges, we extend for Place Timed Petri net and propose a Hierarchical Distributed Real-time Embedded net (HDRE-net). The tasks and the relationships between tasks of DRE software are described in detail. In particular, we abstract communication process as a task, and using HDRE-net to describe resource and time delay of communication process, thus forming the whole application. Based on the constructed model, Time Reachability Graph is adopted to analyze the correctness of HDRE-net, the basic properties of DRE software are also considered.

The remainder of this paper is organized as follows. Section 2 presents the computation model. Section 3 shows how HDRE-net can be used to model DRE systems. Section 4 proposes analysis technologies of DRE software. 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.

doi:10.4304/jsw.5.9.990-997

II. BASIC CONCEPT AND DEFINITION

A. Definition of HDRE-net

Timed Petri nets (TPN) is a mathematical formalism, which allows to model the features present in most concurrent and real-time systems, such as concurrent, asynchronism and distribution, etc. Some recent researches indicate that TPN is powerful enough to describe behavioral features of DRE software [6, 7]. The basic concepts of it can refer to [8]. In this paper, we extend for TPN and establish a model for analyzing DRE software.

Definition 1: A tuple $\Sigma=(TPN,I,\gamma)$ is called Distributed Real-time Embedded net (DRE-net) iff:

(1) $TPN=(PN, C, M_0)$ is a Timed Place Petri net;

(2) $I \subset P$ is a special place, which is called the interface of Σ and denoted by the dotted circle;

(3) γ is the priority function of transition. $\gamma(t_i)=(\alpha_i,\beta_i)$, where α_i,β_i are called the primary and secondary priority of transition t_i;

The distribution of token in each place at time θ is called the marking of DRE-net model, 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 unavailable 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 $x^\bullet=\{y|y\in (P\cup T)\wedge(x,y)\in F\}$.

Definition 2: A six tuple $\Omega=(\Sigma,\Gamma,TI,TA,PI,PA)$ is called Hierarchical Distributed Real-time Embedded Net (HDRE-net) model, where:

(1) Σ is DRE-net model, which describes the basic structure of Ω;

(2) $\Gamma=\{\Gamma_i|i\in Z^*\}$ is the finite set of DRE-net and HDRE-net, each element is called a page of Ω;

(3) $TI\subset T$ is the set of substituted operation, each page of HDRE-net corresponds to a substituted node and denoted by the double rectangle;

(4) *TA* is the page allocation function, whose function is to allocate the corresponding page to the substituted node;

(5) $PI\subset P$ is the set of interface node, which describes the input and output of substituted node, and denoted by double circle;

(6) *PA* is the mapping function of interface, which maps the interface node into the input and output of the operation.

From the definition, we can get that DRE-net is a special case of HDRE-net, that is, Γ in HDRE-net model is empty. In this paper, the firing of transition in DRE-net model is instantaneous and the invocation of transition is determined by its priority. By default, the delay time of place is 0; the priority of transition is (0,0). The time unit can be set according to the specific circumstances. We will analyze the operation mechanism of HDRE-net model in the following.

B. Operation mechanism of HDRE-net

Because the tokens in HDRE-net model include time factor, only using marking can't sufficient to describe the state of the system. In order to better describe time characteristics, we introduce the concept of wait time in this paper.

Definition 3: Let Ω be a HDRE-net model, which reaches marking *M* at time θ, $\forall P_i\in P$, place P_i has j tokens in marking *M*, P_i^k is the *k*th token of P_i. Vector :

$TS(P_i)=(TS_i^1,TS_i^2,\ldots\ldots,TS_i^j)$

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.

$TS(P_i^k)$ is the wait time of token P_i^k. $TS(P_i^k)=m$ explains the model must wait *M* time units before using token P_i^k. While $TS(P_i^k)=0$ represents the token is available. Recorded $TS(M,\theta)$ as the wait time set of places under marking *M*.

Definition 4: Let Ω be a HDRE-net model, a pair $S = (M, TS)$ is called a state of Ω at time θ. Where *M* is marking, which describes the distribution of resources; $TS(M, \theta)$ 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 $\theta+\omega$ ($\omega > 0$), 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 $S[t_i>S'$.

The state *S* reaches state S" by firing transition t_i after waiting ω time units is denoted by $S[(t_i, \omega)>S''$.

Definition 5: Let Ω be a HDRE-net model, $S=(M, TS)$ is a state of Ω at time θ, for transition $t_i\in T$, iff:

(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)*.

The set $ET(M)=SET(M)\cup WET(M)$. 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 firing delay of transition t_i under marking *M*. From the above definition, we can draw that the firing delay of strong enabled transition is 0. $\omega=max(TS({}^\bullet t_i))$ is called firing delay of transition t_i under state *S*, denoted by $FD(S,t_i)$.

Definition 6: Let Ω be a HDRE-net model, $S=(M, TS)$ is a state of Ω at time θ, $\forall t_i\in ET(S),\omega\in N^*$, the firing of transition t_i is effective 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) $t_i\in WET(M)$: $SET(M)=\Phi\wedge FD(S,t_i)\leq FD(S,t_j)$, $t_j\in WET(M)$

The set $U(t_i)=\{t_k|\ t_k\in SET(M)\wedge\alpha_k=\alpha_i\}$ represents the set of transitions whose primary priority are equal to the primary priority of t_i. All the effective firing transitions under state *S* are denoted by $FT(S)$.

Definition 7: Let Ω be a HDRE-net model, $S=(M, TS)$ is a state of Ω at time θ, the model will reach a new state $S'=(M',TS')$ by effectively firing enabled 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 wait time to the new generated marking:

$TS'(P_i^k)=c_i$, P_i^k is generated when firing transition t_i;

Second, modifying the wait time 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 wait time 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 wait time needs to be adjusted.

III. Formally Modeling DRE Software

In this section, we will use HDRE-net model to describe task, resource and communication mechanism of DRE system, thus forming the model of the whole application. In order to distinguish substituted nodes and interface nodes, which is marked by the capital letters, and others place and transition are marked by the lowercase letters.

A. Requirements of DRE software

DRE software can be regarded as a number of modules; each module also contains a number of partially ordered, serial or parallel implemented sub tasks [9, 10]. 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. 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. In this paper, we assume the communication between tasks is done by Controller Area Network (CAN).

As DRE software has strong performance requirements such as predictability, efficiency, reliability and security, et al. Therefore, it is necessary to consider above characteristics when describe the requirements of DRE software.

Definition 8: DRE software requirement model is a 7-tuple $\Xi=\{TK,NT,RS,RL,CP,RT,D\}$:

(1) TK is a finite tasks set;

(2) $N=\{N_1,N_2,\ldots\ldots N_n\}$ is the collection of tasks set, N_i is the corresponding task set of module i, the jth task of module N_i is denoted by $TK_{i,j}$;

(3) RS is the finite resources set;

(4) RL is the relation between tasks, which may be sequence, choice, parallel et al;

(5) $CP:TK\rightarrow(N^*\times N^*)$ is the property function of task, which describes the running time and priority of task;

(6) $RT:TK\rightarrow RS^*$ is the resource function, whose function is to assign necessary resources to each task, RS^* represents the multiple set of resource, that is, a task can use multiple sources;

(7) D is the operation deadline of whole application.

In order to clearly describe the modeling and analysis process of DRE software, we assume the tasks in DRE system have the following characteristics:

(1) Static priority schedule is adopted to realize preemption.

(2) 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 resource: exclusive access and sharing access.

(3) Each task can not suspend itself before completion.

(4) The overhead of task switching can be neglected.

B. Modeling DRE software

In this section, we will use HDRE-net to abstract and model for task, communication between tasks, resource of DRE software, then composing the HDRE-net model of each module according to the relationships between modules, thus forming the model of whole application. In order to distinguish the page of transition and place in each net, the task and module is marked in the corresponding transition and place. The transition and place introduced in the system model will not mark. For example, the beginning transition of task $TK_{i,j}$ is denoted by $t^{i,j}_{st}$.

The model of task $TK_{i,j}$ is shown in Figure 1, where place $p^{i,j}_{ac}$ describes the running state of task $TK_{i,j}$; while transition $t^{i,j}_{st}$, $t^{i,j}_{en}$ describe the beginning and termination position of each task.

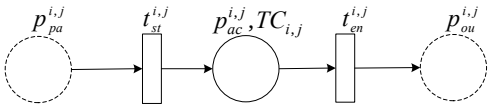

Figure 1. DRE-net Model of Task

Assuming each task of DRE software has constructed the corresponding HDRE-net model which is shown in Figure 1. We will compose HDRE-net model of each task by their relationship.

Operator > represents the sequence relationship: If the firing of task $TK_{i,j}$ can lead to the firing of task $TK_{i,k}$, then the relationships between task $TK_{i,j}$ and $TK_{i,k}$ is sequence. $TK_{i,j}$ is the forward task of $TK_{i,k}$, while $TK_{i,k}$ is the afterward task of $TK_{i,j}$. The set $Forw(TK_{i,j})$, $Back(TK_{i,j})\subset TK$ are the forward and afterward task set of task $TK_{i,j}$. The HDRE-net model of $TK_{i,j}>TK_{i,k}$ is shown in Figure 2(a), the substituted node $TK_{i,j}$ and $TK_{i,k}$ corresponds to the page of task $TK_{i,j}$ and $TK_{i,k}$, while interface node $P^{i,j}_{pa}$, $P^{i,j}_{ou}$ represent the input and output of substituted node $TK_{i,j}$, and mapped into the interface $p^{i,j}_{pa}$, $p^{i,j}_{ou}$ of task $TK_{i,j}$. Because the relation between task and substituted node is one by one, the substituted node is also called task in the following. In order to describe the sequence

relationship, we introduce transition t_{ou} to transfer the result of task $TK_{i,j}$ to the input interface of task $TK_{i,k}$.

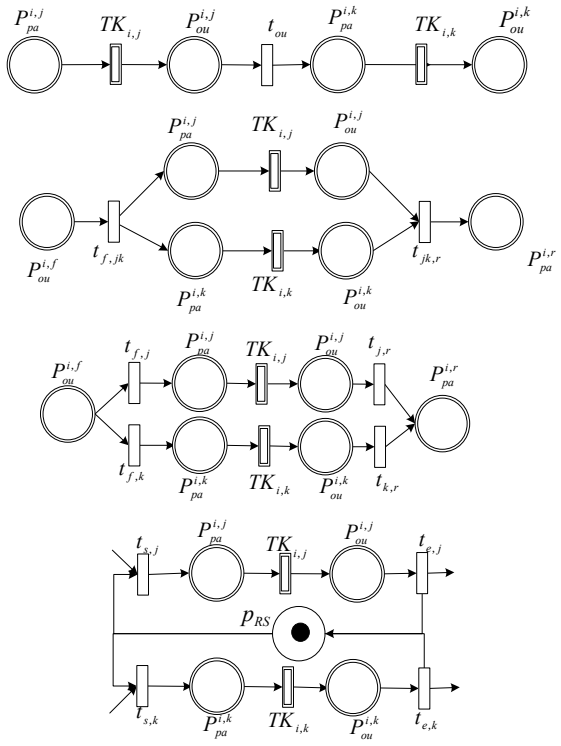

Figure 2. HDRE-net Model of basic Relation

Operator || represents the parallel relationship: If the relation between $TK_{i,j}$ and $TK_{i,k}$ is parallel. Let task $TK_{i,f}$ be the public forward task of task $TK_{i,j}$ and $TK_{i,k}$, while task $TK_{i,r}$ is the public afterward task of task $TK_{i,j}$ and $TK_{i,k}$. The corresponding HDRE-net model of $TK_{i,j} \| TK_{i,k}$ is shown in Figure 2(b), we introduce transition t_{fjk} and $t_{jk,r}$, which make ${}^\bullet t_{fjk}=P^{i,f}{}_{ou}$, $t_{f,jk}{}^\bullet=\{P^{i,j}{}_{pa}, P^{i,k}{}_{pa}\}$, $t_{jk,r}{}^\bullet= P^{i,r}{}_{pa}$, ${}^\bullet t_{jk,r}= \{P^{i,j}{}_{ou}, P^{i,k}{}_{ou}\}$.

Operator $TK_{i,j}+TK_{i,k}$ represents the choice relationship, which means only one task can be chosen to fire. If task $TK_{i,j} \Diamond TK_{i,k}$, which explains there has exclusive relation between them. The corresponding model of $TK_{i,j}+TK_{i,k}$ and $TK_{i,j} \Diamond TK_{i,k}$ are shown in Figure 2(c)-(d).

We will construct the HDRE-net model of each module according to the requirement model and the relationships between tasks, which is shown in Figure 3. The operation process of module N_i is: Calling the tasks in the module according to the task and relationships between tasks after initializing; meanwhile the local clock begins to time, if all task has finished operating in the deadline, then calling termination operation to make it be in normal termination state, otherwise, calling overtime handling operation to make it be in overtime state.

In Figure 3, we introduce place $p^i{}_{st}$, $p^i{}_{en}$ to describe the beginning and termination operation of module. Place $p^i{}_d$ is used to control running time of module N_i, place $p^i{}_{dm}$ describes overtime state of module N_i, transition $t^i{}_d$ describes overtime handling operation of module N_i, where ${}^\bullet p^i{}_d={}^\bullet p^i{}_{df}=t^i{}_{st}$, $p^i{}_{df}{}^\bullet=\{t^i{}_d, t^i{}_{en}\}$, ${}^\bullet t^i{}_d=p^i{}_d$, $t^i{}_d{}^\bullet= p^i{}_{dm}$. If task $TK_{i,j}$ doesn't have forward task, then ${}^\bullet t^i{}_{st}={}^\bullet t^i{}_{st} \cup p^{i,j}{}_{pa}$; If task $TK_{i,j}$ doesn't have afterward task, then $t^i{}_{st}{}^\bullet= t^i{}_{st}{}^\bullet \cup p^{i,j}{}_{ou}$.

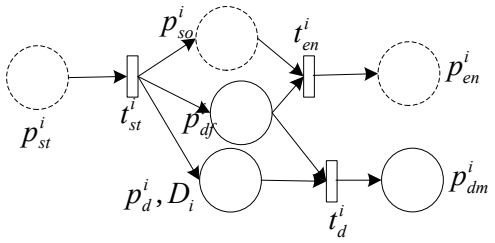

Figure 3. HDRE-net Model of Module N_i

According to the communication principle of CAN bus, the communication process of task $TK_{i,j}$ sending message to task $TK_{g,f}$ is abstracted as a communication task $TK_{i,j\to g,f}$, and set running time $m_{i,j\to g,f}$ of $TK_{i,j\to g,f}$ according to the size of message. The HDRE-net model of task $TK_{i,j\to g,f}$ is shown in Figure 4. In Figure 4, place $p^{i,j}{}_{ou}$ and $p^{g,f}{}_{pa}$ are the input and output interface of task $TK_{i,j\to g,f}$, while place p_b and p_p describe idle buffer and bus token resource. The execution process of communication task $TK_{i,j\to g,f}$ is: beginning to operation($t^{i,j\to g,f}{}_{st}$) after getting data packet and entering into idle buffer waiting position $p^{i,j\to g,f}{}_{gp}$. Once getting the idle buffer, then writing data into buffer($t^{i,j\to g,f}{}_{gp}$) and entering into waiting bus token position ($p^{i,j\to g,f}{}_{wgb}$); the system will release buffer and bus token($t^{i,j\to g,f}{}_{en}$) after

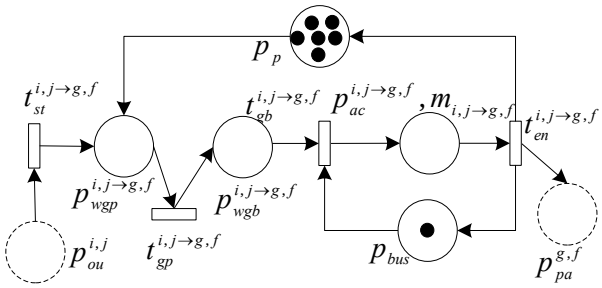

Figure 4. HDRE-net Model of Communication Process

finishing data transmission.

In the HDRE-net model, the token represents resource. According to the characteristics of DRE software, the steps of modeling resource are: for the sharing resources *rs* such as cache, processor, bus, data, I/O and so on, we establish a place p_{rs} to represent the distribution of resource. According to the function *RT* in requirement model, we can determine the relationship between tasks and resources; the message is defined as a resource in the model.

We can form the HDRE-net model of whole application based on the above model. The operation processes of whole application are:

(1) Each module is abstracted as a substituted node;

(2) Constructing data process of the whole application according to the relationships between modules;

(3) Introducing place p_{st}, p_{en} to describe the beginning and termination state of application, transition t_{st}, t_{en}

represent the beginning and termination operation of whole application.

(4) Analyzing the resource of the DRE software, and adding the corresponding place and arcs;

(5) Adding the model of communication task according to the communication requirements between tasks;

(6) Setting initial marking $M_0(p_{st})=1$ and the distribution of initial resource.

In this paper, the allocation rules of transition's priority are as follows: all transitions in a module have the same primary priority, that is, all transitions in a module, including internal transition of the task, have the same main priority; the secondary priority of termination operation has the highest priority, including the termination operation of task, module and communication task, which is 0; Overtime handling has the lowest priority, for example, if the secondary priority is divided into k levels, then the secondary priority of overtime handling transitions in all modules is set to k, the secondary priority of the rest transitions in the module were less than k; the primary priority of communication task can be set according to the actual requirements, and does not follow the above rules, but the main priority of communication tasks which have conflict are not the same, the priority of all inner transitions of communication model are equal; the priority of transition that introduced to describe the process is equal to (0,0).

IV. Analysis Technologies of DRE software

A. Timed reachability graph

For the bounded HDRE-net, because its reachability state set $R(S_0)$ is a finite set, therefore, $R(S_0)$ is viewed as a vertex set, and the direct reachability relation between states is viewed as arc set, and the transition is marked in the corresponding arc, which constructs a directed arc. The directed graph is called Timed Reachable Graph in HDRE-net model. We can analyze state change, transition firing sequence and implementation time of system by using Timed Reachable Graph, thus getting the related properties of HDRE-net model.

According to the construction algorithm of Petri net reachability graph, we can construct Timed Reachable Graph of HDRE-net model. The model uses initial state as root node, and gradually computes each node in Timed Reachable Graph, and does following operations for current state S:

(1) Marking S to ensure that each state only has been visited once;

(2) Computing effective firing set $FT(S)$ of S;

(3) Computing next state S' by arbitrarily choose a transition from $FT(S)$;

(4) If S' has been in the constructed node set, then directly adding the corresponding arc and the side marked, otherwise S' is added into the node set, and add the corresponding arc and side marked.

We can analyze different properties of Ω by using RG(Ω).

B. Basic Properties of DRE Software

The main purpose of modeling by using Petri net is to analyze the properties and function of actual system. This section discusses the basic properties of HDRE-net model, these properties has closely related with DRE software.

In this paper, places are used to represent the resource of DRE software, message storage location, and also can be used to indicate the availability of resources. Making sure whether these storage has overflowed or whether the capacity of resources have overflowed are very important. The boundedness of model is to check whether the described resource of HDRE-net model has overflowed. We will prove the constructed HDRE-net model is bounded in the following.

Property 1: HDRE-net model is bounded.

Proof: According to definition of boundness, we can prove that each place in the model is bounded in different cases, thus getting the model is bounded. Many resources (such as bus, input device of hardware and buffer) are shared resource in DRE software. In those system, there may easily result in deadlock and complete halt, therefore, it is necessary to analyze resource scheduling rules and deadlock-free properties.

Property 2: HDRE-net model ensure that each resource can allocate to one task one time.

Proof: Because the firing of transitions in the model is instant, and the resource in the model is characterized by a token. According to the semantic of HDRE-net model, we can get there doesn't exist a token that can be consumed by two tokens. That is, HDRE-net model ensure that each resource can be allocated to one task one time.

Property 3: HDRE-net model is deadlock-free.

Proof: Because tasks in HDRE-net model can execute only after getting all required resources, and will not require additional resources during proceeding, then HDRE-net model does not meet one of necessary condition of deadlock generated: the transition is obstructed due to requiring resource, and not releases the got resources. Therefore, HDRE-net model does not have deadlock. So the model does not have deadlock.

Property 4: The constructed HDRE-net model can ensure the principles of CAN bus protocol.

Proof: According to the principles of CAN bus protocol: (1) only one task sends message in the bus at any time; (2) If two tasks need to sent message at the same time, then low-priority nodes need stop sending. Because CAN bus is modeling as resource in this paper, we can get principle (1) is established according to property 2 and 3, and because $C(p_{bus})= C(p_{wgb})=0$, two tasks need send message means that there are two transitions that has bus tokens has strong firing, from the firing rules of HDRE-net model, we can get the higher priority transition will send, and the lower priority transition will wait, thus ensuring principle(2).

Schedulability is an important characteristic for guarantying the reliability of DRE systems. We will introduce several special states before analyzing the schedulability of HDRE-net model.

Let Ω be a HDRE-net model, $S=(M,TS)$ is a state of Ω at time θ:

(1) if $\exists\ p^i_{dm} \in P$ which makes $M(p^i_{dm})=1$, then S is called the overtime state;

(2) if $t^i_d \in FT(S)$, then S is called the dangerous state;

(3) if $M(p_{en})=1$, then S is called the 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 the normal state.

Definition 9: Let Ω be a HDRE-net model, then Ω is schedulable iff S^F is reachable in Ω.

The model is schedulable if all modules 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 Ω. 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 Timed Reachability Graph of HDRE-net 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 it 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 the feasible schedule;

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

actual case -Electronic Toll Collection (ETC) as an example. ETC system is an advanced system which consists of high-tech equipment and software such as microwave technology, electronics technology, computer technology, communications and network technology, and can achieve the function of automatically charging the cost of road without stopping vehicle. Strictly speaking, ETC application is the typical DRE software.

The workflow of Export Control Lane (ELC) is: the system will inform lane computer of sending start-up instructions to antenna controller when trigger coil detects the pass of vehicles. The read information from OBE will be sent to data processing center for data processing and charging.

According to the actual requirements, we can divide the entire application into four function modules. Module 1 responses for controlling auxiliary equipment, including traffic lights, display screen, lever and coil; Module 2 responses for reading OBE data. Module 3 responses for processing charged data. Module 4 is used to capture the peccancy vehicles.

According to requirement model of ELC sub-system, we can model for task, module and communication process, and construct the HDRE-net model of ELC sub-system by merging the corresponding interface. Because the conflict task is less, therefore, the setting of priority can reduce level, and the communication buffer is set to 3. Because Module 3 need process a large number of data, ARM9 is used in this module, and the rest modules use 8051 Single-Chip. All modules use SJA100 as bus controller. According to the structure of ELC subsystem, and combining with the completed functions of each module, we can divide task set for each function module, and the corresponding bound is given in table I, in which time unit TTU is 2ms.

TABLE I. REQUIREMENT MODEL OF ELC SUB-SYSTEM

Actual Mapping of Task	Transition	TC(TTU)	Actual Mapping of Task	Transition	TC(TTU)
invoking	TK1,1	1	invoking	TK2,1	1
traffic light display	TK1,2	1	antenna trigger	TK2,2	1
trigger coil induction vehicle	TK1,3	2	connecting OBE	TK2,3	1
sending message to module 2	TK1,4	1	reading data	TK2,4	3
receiving processing results	TK1,5	1	sending data	TK2,5	3
displaying information screen	TK1,6	2	receiving data	TK2,6	1
open level	TK1,7	2	writing data	TK2,7	3
level block	TK1,8	2	disconnect	TK2,8	1
downloading latest data	TK3,1	2	invoking	TK4,1	1
receiving inherent information	TK3,2	1	front capturing	TK4,2	3
querying pull in record	TK3,3	1	side capturing	TK4,3	3
computing charge	TK3,4	2	storing picture	TK4,4	2
connecting account	TK3,5	2	sending data to monitor	TK4,5	2
qmount deducted	TK3,6	2	communication task 1	TK1,4→2,1	0.5
sending processing results	TK3,7	2	communication task 2	TK2,5→3,2	0.5
error info	TK3,8	2	communication task 3	TK3,7→1,5	0.5
storing results and uploading	TK3,9	2	communication task 4	TK3,7→2,6	0.5
			communication task 5	TK3,8→4,1	0.5

V. EXPERIMENTS

In order to better describe the above modeling process and explain the correctness of analysis process, we use an

We can model for task, module and communication process, and construct the HDRE-net model of ELC sub-system by merging the corresponding interface, which is shown in Figure 5. Because the conflict task is less,

therefore, the setting of priority can reduce level, and the communication buffer is set to 3.

According to the construction algorithm of Time Reachability Graph and the HDRE-net model of ELC sub-system, we can get the corresponding Time Reachability Graph. Figure 6 lists part of Time Reachability Graph. The model has 559 states, according to the definition of the basic properties and its analysis method, we can draw the model is bound, deadlock free and live. The principle of CAN bus is also met. And we can draw that the model is scheduling by analyzing Table I.

VI. Related Works

In recent year, there has been some related works used for DRE systems design including formal methods. Subramonian et al[11] presents a reusable library of formal models based on Timed Automaton, which have developed to capture essential timing and concurrency semantics of foundational middleware building blocks provided by the ACE framework. Madl et al[12] investigates how DRE systems can be represented as discrete event systems (DES) in continuous time, and proposes an automated method for the performance evaluation of such systems. Liu[13] leverages the Two-Level Grammar (TLG) specification language and the Vienna Development Method (VDM) a formal methodology for developing DRE components and develops system code generation. UniFrame. Slaby et al [14] used task graph to describe tasks and relations between tasks in DRE systems, and evaluate the performance of component based enterprise DRE systems and reduce time/effort in the integration phase. Liu et al[15] used a timed colored Petri Net-based modeling toolkit describes all specifications consistently and automatically generates component bridges for DRE system integration, and a grammar-based formalism specifies context behaviors and validates integrated systems using sufficient context-related test cases. Hugues et al [16] propose a DRE system design framework based on middleware technology, which use Petri nets for modeling and verification DRE system.

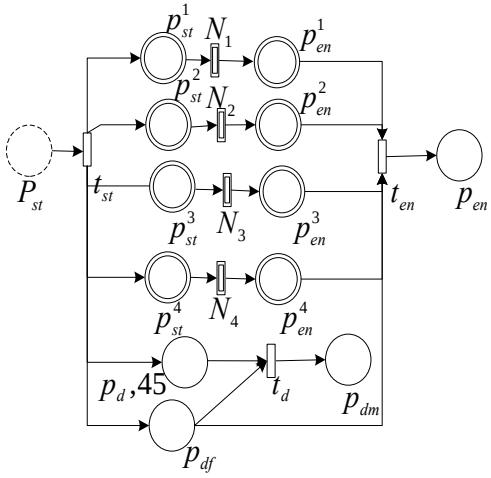

Figure 5. HDRE-net Model of ELC Sub-System

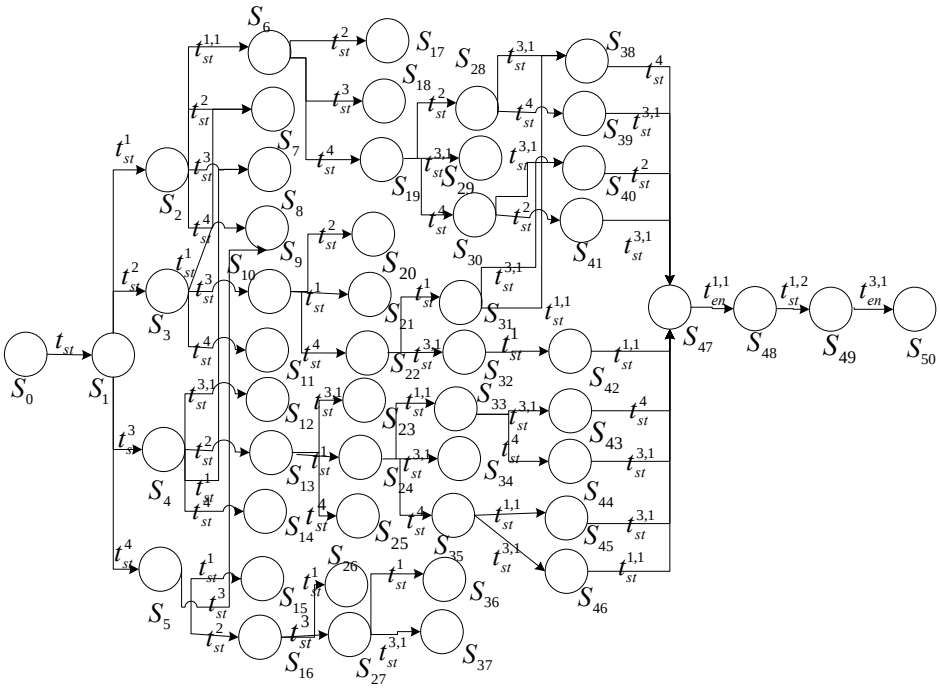

Figure 6. Time Reachability Graph for HDRE-net Model of ELC Sub-System

VII. Conclusion

In this paper, we have proposed a HDRE-net to model and analyze DRE software. This approach is based on a formal model, Petri net, that allows to consider different

components of DRE software. The main contributions of this paper are: (1) we proposed a HDRE-net model to better describe DRE software, thus providing effective management to DRE systems; (2) summarizing the modeling process of DRE software in detail, such as task, resource and communication process; (3) analyzing the correctness of constructed model and related properties of DRE software, thus ensuring the correctness of designment. Using this method to model and analyze the DRE systems has the following advantages: (1) with modular functionality and a high degree of reusability; (2) with a rigorous mathematical foundation, easy to analyze and verify the established model; (3) reduce the computational complexity.

The study of Distributed Real-time and Embedded systems is still underway at present. Our current research is focused on exploring formal method as means to improve its mapping into DRE's architecture. The following two aspects are the main work in the next phase: (1) further improves this method, consider the fault-tolerant of each task to guarantee the system's schedulability; (2) developing the corresponding tools to support the modeling.

ACKNOWLEDGMENT

This paper is supported by key Foundation of Shanghai Educational Committee (07ZZ164, 06OZ016), Foundation of Shanghai Institute of Technology (YJ2004-05) and key subject of Shanghai Institute of Technology (Computer science and technology), Fund of Key Laboratory of Shanghai Science and Technology (09DZ2272600).

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] V. Subramonian, C. Gill, C. Sánchez, et al., “Reusable models for timing and liveness analysis of middleware for distributed real-time and embedded systems,” In Proceedings of the 6th ACM & IEEE International conference on Embedded Software. New York, ACM. pp. 252 – 261, 2006.

[12] G. Madl, N. Dutt, 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, ACM. pp. 183-192, 2007.

[13] S. Liu, “Validation of distributed real-time and embedded system composition in UniFrame,” In Proceedings of the 42nd Annual Southeast Regional Conference. New York, ACM. pp. 303–304, 2004.

[14] J. M. Slaby, S. Baker, J. Hill, et al, “Applying system execution modeling tools to evaluate enterprise distributed real-time and embedded system QoS,” In Proceedings of the 12th IEEE International Conference on Embedded and Real-time Computing Systems and Applications. IEEE. pp. 350-362, 2006.

[15] S. Liu, B. R. Bryant, M. Auguston, et al, “A component-based approach for constructing high-confidence distributed real-time and embedded systems,” In Proceedings of Monterey Workshop on Reliable Systems on Unreliable Networked Platforms. Heidelberg, Springer-Verlag. pp. 225-247. 2007.

[16] J. Hugues, B. Zalila, L. Pautet, et al, “From the prototype to the final embedded system using the ocarina AADL tool suite,” ACM Transactions on Embedded Computing Systems, 2008, 7(4). no. 42.

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

Yunxiang Liu. He was born in 1967, professor, Ph. D. supervisor, IEEE senior member. His research interests include software engineering, information security and formal methods.