Liveness and finiteness of Large Systems with S_Petri Net *

Peijiang Yuan[1] Xiaoming Fu[3] Lichen Zhang[2]
([1] Institute of AI & PR, Science center, Shantou University, 515063 Shantou, Guangdong province, China.)
([3] Department of Computer Science and Technology, Tsinghua Univsity,100084, Beijing, China)
([2] Guangdong Institute of Technology, Computer Science, Guangzhou, China 51002)
(E-mail: g_hhli@mailserv.stu.edu.cn xmfu@263.net)

Abstract: In this paper, a real-time scheduling method with composite Petri net (S_Petri Net) is addressed. With S_Petri net, different sizes of Petri nets express the system modules based on necessity. So the whole system could be built simply and effectively. To avoid conflict and deadlock, a theorem is proved so that the liveness and finiteness can be met easily. Finally an application of real-time scheduling on railway station with S_Petri net is discussed.
Keywords: real-time scheduling, Petri net, S_Petri net, liveness, finiteness

1. Introduction

With the development of Information Technology, more and more resources need to be shared to build compact and effective system. Scholars and engineers aim at finding good scheduling method to make use of the available resources.

The validity of a real-time system depends not only on the computation and logic, but also on the deadline of each task. So the schedule part of a real-time system is different from other common systems greatly. Recently more and more schedule methods are proposed according to the necessity such as task synchronization, instant overload or hybrid schedule method, Priority Exchange (J.P.Lechoczky et.al), Deferrable Server, Spacing Server (Sprunt et.al). Those methods keep the bandwidth of non-cyclic tasks. Thus the performance of such systems could be improved commendably. Yet when the system is overload, it will degrade rapidly.

Scheduling with Petri Net begins with 1980s. The earliest model is usually denoted as M/G or P/T net. Those methods are difficult to analyze and hard to implement. Shen and Liu proposed the division method to avoid such problems. But their methods cause rigid relationship between the parts and the integer.

With common Petri, we can express concurrency and conflict clearly and build contact-free, conflict-free, deadlock-free and fine resource allocation model that could meet most demands. But it can't analyze completely the time requirement of the transitions and status of the places because of the result of data transition. Extend Time Petri Net such as Melin and Farber Net offer an appropriate access time description method, yet it failed in simulating the character of important models. High level Petri net, for example, Colored Petri net and Predicate/Transition Petri net, on imposing data in token pool, could build formalized system model so that time constraint and transitions of control token can be coped with well. But they are rigid and difficult to design especially in large systems. So we use composite Petri net (S-Petri Net). With S_Petri net we can divide a large system into several modules. Time Petri nets are obtained from Petri nets by associating each transition of Petri nets with two times representing real-time constraints.

In this paper, according to the traffic prognostication, a real-time scheduling method with composite Petri net is addressed. With S_Petri net, different

* This work is partly supported by Natural Science Foundation of P.R.China (69874042), Natural Science Foundation of Guang Dong Province P.R.China (970378) and Guangdong Higher Education Department Foundation (9811)

sizes of Petri nets express the system modules based on necessity. So the whole system could be built simply and effectively. To avoid conflict and deadlock, a theorem is proved so that the liveness and finiteness can be met easily. Finally an application of real-time scheduling on railway station with S_Petri net is discussed. Experiments show that with this non-consumable resources scheduling mechanism, the performance of a parallel real-time system could be improved 30%.

The rest of this paper is organized as follows: Section 2 defines the S_Petri net that we consider in this paper. In section 3, a model based on traffic is addressed. A problem is also proposed in this section. Section 4 presents an example of S_Petri net in railway scheduling. Some experimental results are shown in section 5. Finally, we summarize the discussion.

2. S_Petri nets

Petri nets were first defined in [2] and used for scheduling in [3][4][5]. It is obviously that with Petri Nets concurrency can be modeled easily and the operation time of device can be expressed naturally. In order to simplify the problem, we introduce some notations.

Definition 1:

$$Net(S,T;F) :\Leftrightarrow S \cup T \neq \varnothing \wedge S \cap T = \varnothing$$

$$\wedge F \in S \times T \cup T \times S \wedge \dim(F)$$

$$\cup cod(F) = S \cup T$$

Where: dim(F) is the definition domain and cod(F) is the range of $Net(S,T;F)$.

Definition 2:

M enables t:

$$M[t >:\Leftrightarrow \forall s \in {}^{\bullet}t : M(s) \geq W(s,t) \wedge$$

$$\forall s \in t^{\bullet} : M(s) + W(t,s) \leq K(s)$$

Successor:

$$M[t > M' :\Leftrightarrow \forall s \in S :$$

$$M'(s) = \begin{cases} M(s) - W(s,t) & s \in {}^{\bullet}t - t^{\bullet} \\ M(s) + W(t,s) & s \in t^{\bullet} - {}^{\bullet}t \\ M(s) - W(t,s) + W(t,s) & s \in {}^{\bullet}t \cap t^{\bullet} \\ M(s) & s \notin {}^{\bullet}t^{\bullet} \end{cases}$$

In $\sum = Net(S,T:F,K,M)$, if $\forall s \in S, \exists K < \infty, M(s) \leq K$, we call $\sum$ is finite;

On condition that $\forall M \in R(M_0), \exists M' \in R(M) : M'[t >$, $\sum$ is live.

Definition 3:

Reachable tree: Reachable tree of $\sum$ is a full leaf tree structure that is created by the root marking according to certain rules. Reachability tree is finite. The proof can be seen in [2].

Definition 4:

S_Petri net:

$$\textstyle\sum_s = (N_s, M_0) :\Leftrightarrow N_1 = (S_1, T_1 : F_1) \wedge$$

$$N_2 = (S_2, T_2 : F_2) \wedge N_s = (S, T : F) \wedge$$

$$S_0 = S_1 \cap S_2 \neq \varnothing \wedge T_1 \cap T_2 = \varnothing \wedge$$

$$S = S_1 \cup S_2 \wedge T = T_1 \cup T_2 \wedge$$

$$F = F_1 \cup F_2 \wedge (\forall s \in S_0, M_0'(s) = M_0''(s),$$

$$M(s) = \begin{cases} M_0'(s), s \in S_1 \\ M_0''(s), s \notin S_2 \end{cases})$$

Obviously, if $\sum_1$, $\sum_2$ are live and finite, on condition that S_Petri net $\sum_s$ is live and finite, non-consumed resources can be shared so that N_s doesn't affect the proper procedure of N_1,N_2. Thus the common non-consumed resources could be completely shared by the system without causing conflict, contact and deadlock. Such systems would be built effective and economically. Two principles about the liveness and finite of S_Petri nets will be addressed in the following section.

Definition 5:

Inverse Dual order (s_j , s_i): Define (s_i, s_j) the dual order of $\sum_1$ about S_0, (s_j , s_i) is the dual order of $\sum_2$ about S_0 , we call (s_j , s_i) the inverse dual order of $\sum_1$ $\sum_2$ about S_0

Theorem:

Let $\sum_1$,$\sum_2$ be live and finite Petri Net. On condition that $\forall s \in S_0, M_0(s) = 0$, $\sum_s$ is live and finite.

Proof: Since $\forall s \in S_0, M_0(s) = 0$,from the definition 4, we can conclude that $M_0^{'}(s) = M_0^{"}(s) = 0$. It means that before composition, the resources of S_0 have been captured by N_1,N_2. While $\sum_1$ and $\sum_2$ is live, so $\forall s \in S_0, \exists M^{'} \in R(M_0^{'}), M^{"} \in R(M_0^{"})$, we get $M^{'}(s) = 1, M^{"}(s) = 1$, that is $\forall s \in S_0, \exists M \in R(M_0), M(s) \geq 2$, $t^{'} \in s^{\bullet} \cap T_1, t^{"} \in s^{\bullet} \cap T_2, t^{\bullet}$ and ${}^{\bullet}t$ can be fired in M concurrently but free of conflict. So that Petri net N_1 and N_2 can work independently. $\sum_s$ is live and finite.

3. KRC Model With S_Petri nets

3.1 scheduling method

Non-consumed resources are usually shared by all the procedures of a certain real-time system. So the allocation, scheduling and management are the most important method to improve the efficiency. The common schemes of real-time scheduling are three types:

- Complete assign
- Completely shared
- Partly shared

In this paper, we use completely shared scheduling because most of the resources are non-consumed. To simplify the problem, we suppose that there lies an incessant clock that all the transitions are fired under this clock. This clock is independent of

3.2 traffic prognostic and vehicle increment

From [3] we know that vehicle increment is connected with traffic. So we build our model based on traffic prognostic. Calculate the traffic of proceeding years, we get a curve about the traffic increment and vehicle increment. The parameter of the curve can be got in the following way:

Let $y = a_0 + a_1 t + a_2 t^2 + \cdots + a_n t^n$ be the equation of the curve. Then n+1 parameters need to be determined. Sample N time intervals of the proceeding years, we get the increment curve. From the traffic increment curve, the current traffic and increment vehicle can be calculated and a KRC model is available.

1. Example of S_Petri nets in railway

Consider a railway station (region R), there are N tracks and one platform (region I). I⊂R. To avoid two train enter into region I at the same time, a gate is set and three sensors and a switch are used to control the gate. When the train enters region R and I, the sensors capture the signal and send it to the control device. Then the gate is operated to be open or close.

Figure 1 shows a predigest model of this problem: we call it KRC (Kernel of Railway Control) model. In this model, we only consider one train, enter region R and I in certain direction. It costs minimal time dm and maximal time dM to go through from the start point of region R to the start point of region I. Similarly, the train will spend minimal time hm and maximal time hM from the start point of region I and the end point of region I. Command go_up and go_down open and close the gate. As for the gate, once a command valid, the gate will be acted completely. All the interrupts are ignored.

We segment this model into two parts: Region R and Region I. From this S_Petri nets, we can get a reachability tree which shows the

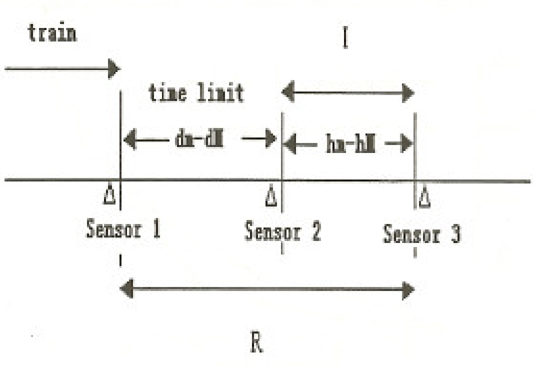

Figure1 Example of the model

time character. Each leaf represents a symbolic marking, notes as M (place)=T, M indicates Mark and place indicates position of the mark. T indicates the time of mark. A Petri net often has many marks. So constraints are set for the marks. For example, at the start of time, in S2, M(closing)=T2, M(In_R)=T1. It means when the gate is closing, the train is just in region R. So it must satisfy the constraint: T2=T1+dm-r. Where dm is the least time that the train needs to go through from the start of region R and the start of the region I.

From the reachable tree, we know that the time limit of each transition could be expressed clearly. According the reachability tree of the KRC model, the request and release of resource of each task can be guaranteed easily. Since this model is on-limits, it is obviously that large size of S_Petri net could be built freely according to the necessity of the industry. From the theorems that proved above we know that this S_Petri net is live and finite. So there is no conflict and deadlock. This model is safe and effective.

5. Experiments and Conclusion

We have implemented the proposed method on a personal computer in JAVA. In this experiment, we suppose that there are N tracks and M trains for scheduling. Each train request for tracks at the average rate of λ_i. All the requests form a Poisson queue. If there is no resource left, the request will be put into the buffer till certain resources are released.

We define M as N= λ M (λ is a constant) for simplicity. Each train request for tracks at the average rate of λ_i. The system is free of conflict and on-limits at the cost of the increasing complexity of the system according to the number of states. Since the speed of computers is being improved, this is not a big problem.

In this paper, a real-time scheduling method with composite Petri net is addressed. With S_Petri net, different sizes of Petri nets express the system modules based on necessity. So the whole system could be built simply and effectively. With the theorem proved above, the system could be built free of conflict and deadlock. Experiments show that this method could be used well especially when non-consumed resources need to be shared. Future work should emphasize on the reduction of reachable tree and find better solution to express the transition and fire.

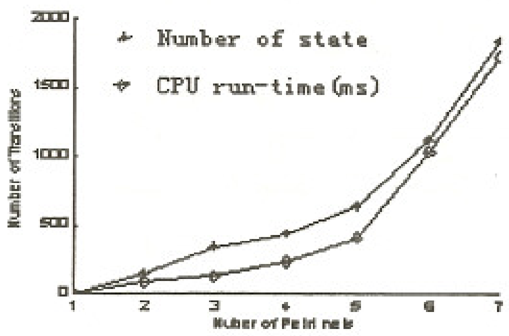

Fig.2 CPU run time and transitons

6. Reference

1. Ramamritham, K.,Stankovic, J.A.,and Zhao,W., Distributed scheduling of tasks with deadlines and resource requirements.IEEE Trans.Comput.C-38,8,1110-1123.
2. Yasukichi Okawa, Tomohiro Yoneda, international Journal of Mini and Microcomputer, (148-156) Vol.18, No.3,1996
3. Yuan chongyi, Principle of Petri net, publish house of electronics industry, P.R.China.
4. B.Berthomieu & M.diaz, Modeiling and verification of time dependent systems using time Petri nets, IEEE Trans. on software Eng., 17(3), 259-273,1991
5. L.Lamport, "A Fast Mutual Exclusion Algorithm," ACM Trans. Computer Systems, vol.5, No.1, pp.1-11, Feb. 1987.
6. Yuan Peijiang, Zhang Lichen, et.al, Computer Engineering and Application, Fault Tolerance of Distributed real-time Network, Vol.35, Supplement, 1999, pp. 153-158
7. Nicholas M., Wei Zhao, Version Selection Schemes for Hard Real-Time Communications. Proc. Of 12th Real-Time Systems Symposium, IEEE Computer Society Press,1991
8. Development of Chinese transport and post industry, National publication, Dec. 1989
9. D.J.Evans * and W.U.N.Butt, "Dynamic load balancing using task-transfer probabilities", in Parallel Computing, 1993, pp 897-905