MATEC Web of Conferences **61**, 04023 (2016) DOI: 10.1051/matecconf/20166104023
APOP2016

Analysis on Real time Perception Technology of Wireless Sensor Network in CPS

Zhou Benhai[1,a]

[1]Teaching Department of Computer Science, Shenyang Institute of Engineering, Shenyang City, Liaoning Province, 110136, China

Abstract. Cyber Physical Systems (CPS) combines physical and computing systems tightly. Node operating systems (OS) are fundamental units in CPS. There are still many problems unsolved when designing CPS especially CPS node OS in aspects of predictability, reliability, robustness, etc. Aiming at the problem, this paper proposes the effective shortest time priority algorithm and the adaptive shortest time priority algorithm. Experimental results show that, compared with the traditional FIFO (advanced first in first out) and LSF (least slack first) algorithm, the algorithm proposed in this paper effectively reduce the deadline miss ratio, as a result, the real-time performance of the CPS are effectively improved.

1 Introduction

Cyber and physical system (CPS) is a complex system, which contains the network and the physical environment. CPS is closely integrated with many technologies, such as computing, communication and control. The system must be aware of the constraints caused by environmental changes in real time. At present, the CPS application in the medical, military and other related countries and people's livelihood in important areas, has a very important role, which is a higher requirement for real-time performance of the system. Therefore, the research of CPS real-time scheduling algorithm can meet the requirements of high real-time performance of CPS system, and it is also conducive to accelerate the construction of information in various fields[1-7].

Since CPS have diverse computational and resource characteristics, they are complicated and there are still many research challenges. One example is how to describe architecture of cyber physical systems. In this paper, we propose a 4-layer architecture for CPS.

This layer is at the bottom of CPS architecture including a large number of CPS nodes on which node operating systems are running. This layer is directly responsible for the interaction with physical systems. Nodes collect physical data. They not only transmit data up to the upper layer, but also accept feedback data. They control physical operations of reactors. Node operating system controls interaction devices and other hardware facilities of nodes to complete the feedback loop.

2)Data Internet Layer

This layer should receive sensor data and send control data seamlessly to ensure information two-way exchange within horizontal and vertical systems transparently.

3) Information Integrating Layer

The data collected from lower layers are different in types and formats. Even some data are uncertain. So CPS must do the corresponding work (data cleaning, data reorganizing, data fusion, etc) to pick out useful information from raw data. Meanwhile, this layer also transforms the reverse interaction information to corresponding control data for distribution to the lower layer.

4)Application Service Layer

This layer is the interface layer between CPS and upper applications. It could encapsulate all functions from lower layers as various services to provide to applications as interface specifications or standards.

Real time scheduling is an important part of real-time system[6-8] and it is also a critical part of CPS system, which is the key to guarantee the real-time task. It is also a widely studied problem. In the current research, a lot of real-time scheduling methods are proposed for various task types. But in the CPS, important parameters of real-time tasks with the release time and deadline will be more dependent on nodes in the CPS location and migration time and other factors. Therefore, traditional real-time scheduling algorithms of the physical factors influence considered insufficient, easy to cause the system overload operation, which makes the real-time task deadline missed. The system's real-time perceiving performance is seriously affected, causing major production safety hazards as well as economic losses. In view of these problems, this paper proposes a CPS real-time scheduling algorithm considering the physical environment factors.

[a] Corresponding author: 13734610r@qq.com.

This paper is supported by Research foundation of Liaoning Provincial Education Department (L2014532, L2014520, L2014523), National Social Science Fund(15CGL050).

2 The real time scheduling model of CPS

In this paper, we propose a real-time scheduling algorithm for the mobile real-time service node (CPS), which has many nodes in the network. This paper proposes a CPS real-time scheduling algorithm, which enable the system to achieve the maximum deadline satisfaction rate. Therefore, based on the real-time scheduling algorithm under CPS network framework of the mobile real-time service node, the model is describe as follow:

l_i : slack time of the task T_i (exponential distribution average $1/l$).

e_i : execution time of task T_i (evenly distributed on [0, E]).

m_{Ti} : migration time (Uniform distribution on [0, M]).

DMR : Rate of meeting deadline(DMR=tasks of meeting deadline/all of tasks)

In the real time CPS of the mobile node, in the absence of task conflict, the real-time task DMR is the relaxation time of Ti, which is more than that of the real-time service node. Real time task DMR Ti is the probability that the relaxation time of Ti is greater than that of real-time service nodes. Given As distribution of l_{Ti} is le^{-lt} , the DMR of T_i （$DMR_{Ti}(l,m)$）is computed as follows:

$$DMR_{Ti}(l,\mathrm{m}) = \int_m^{\infty} le^{-lt}dt = e^{-lm} \tag{1}$$

As m is assumed to evenly distributed [0,M], an average DMR is:

$$Mean(DMR_{Ti}(l,\mathrm{m})) = \frac{1}{M}\int_0^M e^{-lm}dm = \frac{1}{lM}(1-e^{-lM}) \tag{2}$$

In case of conflict between the two real time tasks, the paper will give the FIFO (first in first service), LST (shortest time priority), and ELST (CPS effective shortest time first) algorithm to calculate the average DMR:

3 Improved effective LSF algorithm

Effective LSF algorithm is an optimal algorithm in real-time scheduling algorithm. However, when the real-time tasks are scheduled in CPS system, DMR is needed to consider factors of the physical environment. In the real-time service CPS, the real-time service node migration time is one of the physical factors that need to be considered. Therefore, the effective shortest relaxation time of the real-time service node is considered. Therefore, when the real-time task is scheduled, the task with the most short time of slack time is first scheduled. We can use $l_{elsf,\mathrm{T}i} = l_{Ti} - m_{Ti}$ denoting the effective slack time of T_i .

Calculation methods are as follows: As the distribution function of l_{Ti} be le^{-lt} , the distribution function of $l_{elsf,\mathrm{Ti}}$ >0 is below.

$$\frac{1}{M}\int_0^M le^{-l(t+m)}dm = \frac{e^{-lt}}{M}(1-e^{-lM}) \tag{3}$$

When the distribution of $l_{elsf,\mathrm{Ti}}$ in（-M,0）, the function is as follows:

$$\frac{1}{M}\int_{-t}^M le^{-l(t+m)}dm = \frac{1}{M}(1-e^{-l(M+t)}) \tag{4}$$

When T1 and T2 has not conflict, the DMR of T1 should satisfy the probability of $l_{elsf,\mathrm{T}_1} > 0$ which is described as follows.

$$\begin{aligned} Mean(DMR_{T_1}(l,\mathrm{m})) &= \frac{1}{M}\int_0^{\infty}\frac{e^{-lt}}{M}(1-e^{-lM})dt \\ &= \frac{1}{lM}(1-e^{-lM}) \end{aligned} \tag{5}$$

The deadline meet ratio of task B following task A is:

$$\begin{aligned} &Mean(DMR_{T2}(l,\mathrm{m}_1,\mathrm{m}_2,\mathrm{e}_1)) \\ &= \int_{m1+m2}^{\infty}\frac{e^{-lt}}{M}(1-\mathrm{e}^{-lM})\,\mathrm{dt} \\ &= \frac{(1-\mathrm{e}^{-l\mathrm{M}})}{lM}e^{-l(\mathrm{m}_1+e_1)} \end{aligned} \tag{6}$$

As we assume m1 and e1 are evenly distributed on [0,M] and [0, E], respectively, mean DMR of m2 is computed as:

$$\begin{aligned} &Mean(DMR_{T2}(l,\mathrm{m}_1,\mathrm{e}_1,\mathrm{m}_2)) \\ &= \frac{1}{ME}\int_{-t}^{\mathrm{E}}{}_{m_1+e_1}\frac{e^{-lt}}{M}(1-e^{-lM})?\,e^{-l(\mathrm{m}_1+e_1)}d_{m_1}d_{e_1} \\ &= \frac{(1-e^{-lM})^2(1-e^{-lE})}{l^3M^2E} \end{aligned} \tag{7}$$

It can be concluded from the results of the ELSF, average deadline meet ratio($Mean(DMR_{T1}(l,\mathrm{m}))$) of T1 equals average deadline meet ratio ($Mean(DMR_{T2}(l,\mathrm{m}_1,\mathrm{e}_1,\mathrm{m}_2))$) of T2. In addition to ELSF, the other parameters are the same as the FIFO algorithm and C algorithm except $l_{elsf,\mathrm{T}i} = l_{Ti} - m_{Ti}$. Also, FIFO algorithm applies $l_i > m_i$, and ELSF algorithm is applied by $l_{elsf,\mathrm{T}i} = l_{Ti} - m_{Ti}$. Both of DMR are the same.

Therefore, FIFO and ELSF algorithm to meet the deadline rate is the same. It can be concluded that the ELSF algorithm cannot effectively improve the real-time performance, so the optimization of ELSF algorithm is studied in this paper,

Let p be the probability of DMR of firstly scheduled T1.And then，let q be the probability of DMR of the secondly scheduled T2.

$$\begin{aligned} p &= Mean(DMR_{T1}(l,\mathrm{m})) = \int_0 \frac{(e^{-lt}1-e^{-lM})}{M}dt \\ &= \frac{1-e^{-lM}}{lM} \end{aligned} \tag{8}$$

$$
\begin{aligned}
q &= Mean(DMR_{T2}(l, \mathrm{m}_1, \mathrm{e}_1, \mathrm{m}_2)) \\
&= \frac{1}{ME}\int_{-t_{1}+e_1}^{E} \frac{e^{-lt}}{M}(1-e^{-lM})?\, e^{-l(\mathrm{m}_1+\mathrm{e}_1)} d_{m_1} d_{e_1} \\
&= \frac{(1-e^{-lM})^2(1-e^{-lE})}{l^3 M^2 E}
\end{aligned} \tag{9}
$$

Optimal ELSF algorithm is designed in this paper, considering the real-time service node in the mobile time and relaxation time, so as to improve the CPS nodes task deadline satisfaction rate. When the task T1 and T2 conflict, ELSF algorithm will maximize the system to meet the deadline rate. In order to T_1 ® T_2 task sequence, it will be divided into the following four scheduling methods.

4 The Description of Scheduling Method

Now , we describes the main fuction of optimal ELSF algorithm.

Initialization:

```
#include <stdio.h>
#include <iostream>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>
#include <math.h>
#define queuesize 10
#define movetime M
#define MAXTIME 150 /
#define MINLLF 9999
#define PRO_LRU_T 0
```

Function:

//find the least slack time process

```
process *elsf(sqqueue *dui, int currenttime)
{
sqqueue *q1 = dui;
process *currentpro, *pro;
int minllf = MINLLF , llf;
int pro_LRU_t = PRO_LRU_T;
int front = q1->front, rear=q1->rear;
currentpro = q1->data[front];
if( currenttime <= MAXTIME )
{
//compute currentpro，if there was only a process
if( front==rear )
{ return currentpro;
printf("%dms%s%d%d\n",currenttime,currentpro-
        >pname,currentpro->cycle,llf);
} //more than a process
else
{
do
{
if(front!=rear )
{
pro=q1->data[front];
if(pro->arivetime <= currenttime && currenttime<=
        MAXTIME)
{
//Slack time equals to period is multiplied by execution
times
llf=(pro->deadtime)*(pro->cycle)- pro->lefttime -
        currenttime;
if(minllf>=llf) //get the least slack time task
{ if( pro->LRU_t >= pro_LRU_t )
{
pro_LRU_t = pro->LRU_t ;
minllf=llf;
currentpro=pro;
} } }
front=(front+1)%queuesize; }
else
break;
}while(front!=rear ); //test queue} }
return currentpro; }
```

// get the preemption process

```
process *leastlaxityfirst(sqqueue *dui,int currenttime)
{
sqqueue *q1=dui;
process *pro=NULL, *nextpro;
int front= q1->front, rear= q1->rear ;
/*
Find the process which the slack time is zero.
*/
while(front!=rear)
{
nextpro=q1->data[front];
/*
pro->latestarttime
(pro->deadtime)*(pro->cycle) - pro->servetime;
pro->latestarttime - currenttime
pro->latestarttime - currenttime <= 0 ,
*/
if( nextpro->latestarttime <= currenttime )
break;
else
front=(front+1)%queuesize;
}
//if the queue is empty, return pro
if(front==rear)
return pro;
// if the queue is not empty, nextpro is the running
process which can preempte the others.
else
return nextpro;
}
```

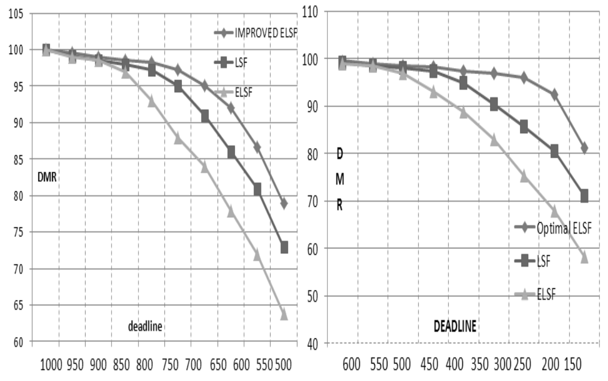
5. Experiments and analysis

In this paper, we test the performance of the real-time scheduling algorithm through simulation experiments. To verify the performance of the algorithm in the CPS system, the performance of the ELSF algorithm is verified in the case of multi task conflict. In order to verify the performance of real-time scheduling algorithm, the following parameters are configured in this paper:

Ei is denoted by execution time at node i with exponential distribution of average 1/ λ . In the experiments, we set the λ =0.02 and λ =0.2. Di is deadline at the node i with variable range. S is represented by the moving distance between computing node c and serviced node i. let s be $s = \sqrt{(x_i - x_c)^2 - (y_i - y_c)^2}$. The node C and service node I are distributed in a square of 100 by 100. Ri is release time, the value of Ri is 0. N is denoted by the number of simulations, which is 1000 times.

The paper uses DMR to test performance of optimal ELSF algorithm, LSF algorithm and LSF algorithm in task conflict situations. Experimental results are shown in fig 2.

The paper uses the DMR to test performance of optimal ELSF algorithm, LSF algorithm and LSF algorithm in task conflict situations. Experimental results are shown in below.



(a) (b)

Fig 1. (a)When cyber factor is quite big in comparison to physical factor.(b)When cyber factor is quite small in comparison to physical factor.

Experimental results show that the ELSF algorithm and LSF algorithm have different performance in the case of both information and physical factors (ei and mi), regardless of the specific gravity of the information factor, and the traditional LSF algorithm is better than the ELSF algorithm in the special condition. From the simulation results of Figure 3, the ELSF scheduling algorithm can show better performance when the information factor of CPS is relatively large. On the other hand, the traditional LSF algorithm has better performance than the ELSF algorithm when the physical factors are more than the information factor. However, in the above two cases,the optimization of the algorithm ELSF are able to exhibit better performance. The optimal ELSF scheduling algorithm can crease the DMR by up to 20% compared to other ways.

References

1. J. Stankovic, I. Lee, A. Mok, R. Rajkumar, Opportunities and obligations for physical computing systems, IEEE Computer, Volume 38, Issue 11, 23-31, 2005.
2. John A. Stankovic, When Sensor and Actuator Networks Cover the World, ETRI Journal, Vol. 30, No. 5, 627-633, 2008.
3. Fang-Jing Wua, Yu-Fen Kaob and Yu-Chee Tseng, From wireless sensor network towards cyber physical systems, Pervasive and Mobile Computing, Vol. 7, Issue 4, 397-413, 2011.
4. Edward AL and Sanjit AS, Introduction to Embedded Systems, A Cyber-Physical Systems Approach. NY: Lee & Seshia, 2011:119-158.
5. Hussain D, Illon T.Event. Cyber-Physical Systems Method. In proceedings of Object Component Service Oriented Real-Time Distributed Computing, Beijing, IEEE press,2011: 355-349.
6. Smita P, Prabha K. A Qos Based Mac Protocol for Wireless Multimedia Sensor Network[J]. Journal of Electronics and Communication Engineering, 2012,1(5): 30-35.
7. Yong F, Qiang G. Design of a Wireless Sensor Network Platform for Real-Time Multimedia Communication [C]. In Proceedings of International Conference on Communication, Electronics and Automation Engineering, NY, IEEE press, 2013:1305-1311.
8. Gann H. A Distributed Wireless Sensor Networks Mobile Communication Technology Research[C]. In Proceedings of Multimedia Information Networking and Security (MINES), UK, 2012, 203-207.
9. Shah, A. Cross-Layer Framework for QoS Support in Wireless Multimedia Sensor Networks[J], IEEE Transactions on Multimedia, 2012, 14(5):1442-1455.
10. Hong W, Research on Visual Minor Platform Based on ARM and DSP [D], Da Lian: Dalian University of Technology, 2009.
11. Eswaran A, Rajkumar R. Energy-aware memory firewalling for QoS-sensitive applications[C]. In Proceedings of the Euromicro Conference on Real-Time Systems, Palma de Mallorca, Spain:IEEE press, 2005:11–20.
12. Maria MG, Pedro M and David S. Model Checking Dynamic Memory Allocation in Operating Systems [J], Journal of automated reasoning, 2009,42(2):229-264.