Context-Aware Scheduling Algorithm in Smart Home System

ZHENG Hong[1], PAN Li[2], WANG Jingxiao[1]

[1]Department of Computer Science and Engineering, East China University of Science and Technology, Shanghai 200237, China
[2]Department of Information and Communication Engineering, Hunan Institute of Science and Technology, Yueyang 414006, China

Abstract: The main idea of pervasive computing is to make computing exist everywhere in the physical world. The smart home system is an important realisation of pervasive computing whose aim is to provide system users with an intelligent life experience. The key technique used to realise this is context awareness. Contexts in the living space can provide large amounts of information regarding users' behaviours and habits. Together with an information system, it can automatically execute many common operations of applications, instead of users, and can make the applications "smart". However, since contexts in the environment are diverse and sensitive, it is difficult to choose the ones that are most useful to the users' current activity. A proper scheduling strategy should first consider the users' demand. This paper proposes a context-aware scheduling algorithm that is based on correlation, with the purpose of improving the utilization rate of context collections. Experiments show that with the priority based on correlation in low-level contexts, the scheduling of reasoning tasks can reduce the cost of transmission.

Key words: pervasive computing; smart-home; reasoning scheduling; correlation

I. INTRODUCTION

The idea of pervasive computing is to bring about the existence of computers anywhere in our physical world without sensation of users, users are the centre. Every computing unit will cater to the users' habit and make autonomous interactions with users' activities. In that case, people in the pervasive environment can concentrate on their working or life enjoying without spending energy on operating the machines. Furthermore, pervasive computing also will provide users customised information services by perceiving and predicting users' demand. Pervasive computing infrastructure in the home should be able to sense the context in which specific situations take place and adapt to them according to its location of use, the people and objects that are around, and changes of those entities over time, how to make an effective use of the context information is an important and difficult topic. Context scheduling plays a vital important role in this field. The main purpose of context scheduling in smart home environment is to make a sequence of tasks performing. With the sequence, applications in the system can provide proper services to the users with effective using of context information collected from the environment. The context scheduling algorithm in smart home system should consider not only the user preference but also the cost of context transmission. In scheduling theory, a real-time system comprises a set of real-time tasks; each task consists of an infinite or finite stream of jobs. The task set can be scheduled by a number of policies including fixed priority or dynamic priority algorithms [1-3]. The common dynamic priority scheduling algo-

Received: 2012-12-14
Revised: 2013-03-01
Editor: Ben Falchuk

This paper proposes a priority scheduling algorithm based on context correlation to improve the average utilization of low-level context in the pervasive computing environment, such as smart home system. Experimental results show that the proposed algorithm can reduce the times of cache replacement and the times of data transmission.

rithms do not consider context-aware of pervasive environment. This paper focuses the scheduling problem on the communication process between low-level context and the context reasoning module, and proposes a context priority scheduling algorithm based on correlation with the purpose to solve the problem of behaviour disorder of system caused by the conflict in context reasoning, and to improve the utilization rate of context collections in the smart home environment.

The rest of this paper is organised as follows. In Section II we review introduce related works and discuss a context-aware model in smart home. In Section III we present an improved context-aware scheduling algorithm based on correlation, and illustrate the algorithm on an example. Section IV provides experimental support for our improved context-aware scheduling algorithm. Finally, some conclusions are drawn in Section V.

II. Context-Aware in Smart Home System

2.1 Related work

Many applications of pervasive computing have to access some related contexts in order to provide the right services at the right time and the right places. However, there are some challenges for applications which need to store a lot of information accessed in the future previously, and to expend the information of communication channel timely in smart home. These challenges make the design of smart home applications much more difficult than other applications. The smart home system should effectively integrate communication and computing networks among the previously separated electronic devices in home and incorporate core tenets of pervasive computing. Many researchers have focused on context-aware smart applications. Georgia Tech's Aware Home experimented with many technologies for capturing low-level sensor data about human activities but did not focus on automated activity recognition. The MIT House project developed an instrumented condominium

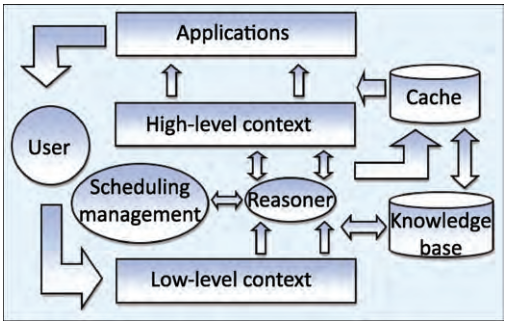

Fig.1 *Model of the context scheduling in smart home system*

for studying activity recognition in a naturalistic environment. This space is designed as a living laboratory from which experiments in activity recognition can be conducted in a naturalistic manner [4-9]. A context-aware smart home always utilises contextual information of the involved entities to adapt its behaviour and provide context-aware services. Context-aware in smart home system should have the ability to forecast user behaviours actually. For example, whether a homeowner is at work or on vacation, the smart home will alert him to what is going on, and security systems can be built to provide an immense amount of help in an emergency. And the cost to realise the system must be reduced to the level that user can accept.

Some researchers have more focused on context-aware information abstraction and pretreatment rather than context scheduling. Ref. [4] presented a distributed system model for context-aware mobile computing, which can collect information from sensors, reason from known database, and thus adopt corresponding activities. A. Ranganathan et al. proposed a context model based on first order predicate calculus. They use first order model to express complex rules, which involves contexts. This knowledge expression enables automated inductive and deductive reasoning to be easily done on contextual information [8]. Ref. [9] clarified the notion of context and laid out foundations for the design and development of context-aware applications. In this paper, the context-aware scheduling algorithm based on correlation in smart home environment also

inherits this method.

2.2 An interactive system model of context-aware in smart home system

In general, the context is a meaning of the broad concept that can be considered context is affecting the behaviour of a group of computing entities implicit input. Context is typically the location, identity, and state of people, groups, and computational and physical. objects. And the context in home environment can be divided into two categories: physical context, including the position, temperature, humidity, noise, etc.; complex context, including the human behaviour, user's facial expression and so on. Ref. [10] provided a basic model of context model which describe the implementation process of context collection in a common intelligent environment as shown Figure 1. To enhance user's experience, the context must be given different priorities taken from our previous work [6]. The higher priority means that the context is more sensitive to the user and easily to bring about user's special behaviour. Users' habits would be gathered in the historical records by contrasting the data access. Then the priority of context would be modified in order to make sure that context has high priority is the one that sensitive to the users. The communicating channels also should be given. By collecting feedback information from users directly, some personalised adjusts would be given to the priority of context.

2.3 Preference of user experience

In smart home environment, man is the centre of the whole system. Services are customised by the actions and preference of the user. Without user's behaviour and preference, the pervasive computing environment will not provide service with high-level personalization. The sensors cannot easily collect those indirect-contexts directly; however, we can learn the user preference by backtracking the historical records.

Different form the application layer, user will not have directly interaction with the reasoning and scheduling module. We can only collect user preference by tracking the application access. The more appearance of one context the more sensitive it will be to the user in the system.

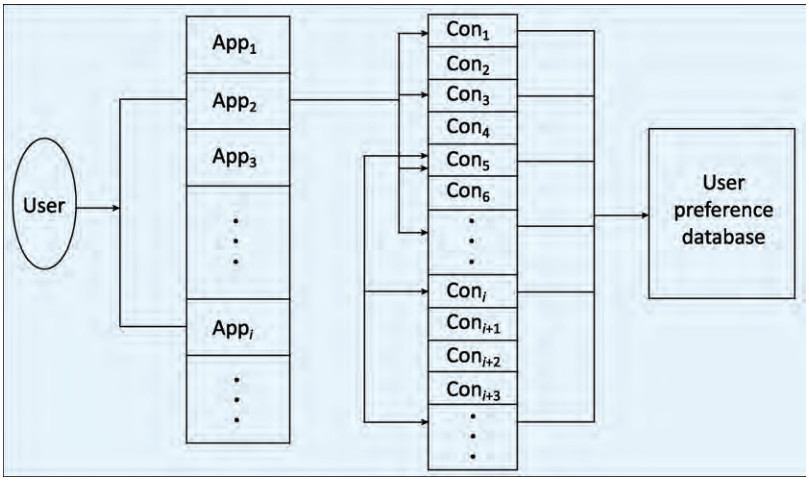

Fig.2 *Obtain the user preference*

2.4 Illustrative use case

Air-conditioner, TV set, lighting system, water heater, cleaning robot, security system and other intelligent devices are placed in the smart home system. At 8:00 a.m., Miss WANG leaves home and goes to work. System judges no one in the house then opens the security system. Cleaning robot begins to sweep the floor, and other devices turn off to keep safe. At 7:00 p.m., when Miss WANG comes back home, the system perceives that there are user activities in the house. Then it will turn on the lighting system where user being. The air-conditioner works, and the TV set begin to broadcast the program that she preferring. At 11:00 p.m., she goes to sleep. System turns on security system and turns off the TV set. Air-conditioner keeps on working but switch to "sleep" mode.

In these common scenarios [11], smart home system controls intelligent devices with proper applications. These applications learn different scenes from contexts information in the environment. The lighting system mostly has no need to work during the daytime. But in the evening, it becomes the most important

device for user. It is better to let the robot do cleaning work when no one is in the house and let the air-conditioner work when user back home. Some context information affects other one's importance. Such as time, luminance and user activities, we should give these contexts high priority in order to realise the dynamic regulation of the priority of intelligent devices in different scenes. Task scheduling strategy should consider these relationships between different contexts and modify execution sequence with the environment information changing.

III. Priority Scheduling Algorithm of Context-Aware

3.1 Algorithm process and basic definition

Having presented context information types in smart home system, we now discuss how applications can effectively use context information. Context reasoning is the essential part of context-aware computing. And task scheduling is the important constituent part of any system. Also, context reasoning needs an effective strategy to manage reasoning tasks. The more important task has the higher priority, and if they have the same priority, whether it is disruptive becomes the primary basis of scheduling. The main steps of priority calculation are shown as Algorithm 1.

Algorithm 1 The main steps of priority calculation

```
1:Priority contextPriority (context c_j){
2:    while (task queue is not empty)
3:        T=head of the task queue
4:        while (queue T is not empty)
5:            if (queue T has correlation with c_j)
          //if task T need context c_j to reason, it
          //has correlation with c_j
6:                put T into queue T_r
                      T=T->next
7:            end if
8:        end while
9:        if (queue T_r is not empty)
10:           while (queue T_r is not empty)
11:               t=t+Te_j
                  n=n+1
          //calculate the whole execution time
      //and number of tasks having correlation with c_i
                  T_r=T_r->next
12:           end while
13:           Te_ave=t/n
14:       end if
15:       trans_i=Tt_j/Tt_ave
16:       P_j=(r_j/Te_ave)*(n/N)*trans_j+C
17:       return P_i
18:   end while
19:}
```

To evaluate the priority of reasoning task, some attributes will be considered:

First, the direct correlation of low-level contexts among tasks is useful. In pervasive computing environment, transferring cost probably restrict the efficiency of system processing. Choose the task having the most same low-level contexts with current task as the next task to be scheduling will reduce the cache replacement times.

Second, low-level contexts with long refresh period have high reusability. Give high priority to these contexts also can reduce cache replacement times.

Third, times a low-level context appearing in tasks queue can show the importance of the context to the system.

In this paper, we give basic symbols definition as follows:

1) Define c_j as the low-level context, j is the number of the context;

2) Define t_i as the query task, i is the number of it;

3) Define r_j as the refreshing period of c_j. That means c_j will lose efficacy when exceeding r_i, and need to be collected again;

4) Define Tt_j as the transmission cost of once refreshing c_j, and Tt_{ave} as the average transmission cost of all the contexts;

5) Define N as the number of all tasks in the query queue, and n_j as the number of tasks which having correlation with low-level context c_j;

6) Define Te_i as the execution time of t_i, and Te_{ave} as the average execution time of t_i;

3.2 Priority of the contexts

In the smart home environment, contexts have different updating ratios. Compared with the updating frequently ones, contexts with long refreshing period are more valuable to be sto-

red in the cache. Long efficient time allows more applications access contexts in one refreshing period. But, if not paid much attention by the system, a context even with long effective duration should not have a high priority. With the purpose of increasing the low-level context utilization ratio and reducing the cost of communications between sensors and reasoner, we give high priority to the context that has high access rate during one refreshing time. Besides, refreshing context with higher transmission cost will make more time cost. So we give high priority to the context having high transmission cost. The priority of context will be defined with three main qualifications: the refreshing period, the transmission cost of context and the times accessed by applications.

Given a context c_j, r_j is its refreshing period. Define its sustainable freshness F_j as follows:

$$F_j = \frac{r_j}{Te_{\text{ave}}} \tag{1}$$

where F indicates the efficiency of access of context c_j. Within one refreshing period, the more tasks which related to c_j can be executed the more valuable for c_j to be stored in the cache. We give the expression of Te_{ave} as follows:

$$Te_{\text{ave}} = \frac{\sum_{i=1}^{n} t_i}{n} \tag{2}$$

Another important parameter to calculate the priority of c_j is the visit possibility. Define A_r as the percentage of tasks having correlation with context c_j among all the tasks. A_r predicts that how many tasks in the task queue will query the value of c_j:

$$A_r = \frac{n}{N} \tag{3}$$

Define trans_j as the indication of transmission cost of c_j:

$$\text{trans}_j = \frac{Tt_j}{Tt_{\text{ave}}} \tag{4}$$

Higher value of trans_i means that if c_j lose efficacy, higher cost to refresh it should be paid.

With the three basic parameters above, this paper gives the definition of the priority P_j to the context c_j as follows:

$$P_j = F_j \times A_r \times \text{trans}_j + C \tag{5}$$

For any low-level context c_j, the tasks having short execution time will have higher implementation rate within one refreshing period than the ones having long execution time. So we give the former higher priority to be reasoned. But if the refreshing period of the context is very short, the reasoning result will not be valid for long. So we reduce the priority of contexts in this kind. In addition, if the context has long time to be ignored by the applications, this means that we have no need to give high priority to it. Another parameter C in this expression means the user preference. For different living habits, users also have their own concern. If the users care about some context in the environment (such as temperature) specially, they can give the high priority to it.

3.3 Description of task and partition of task set

This paper use an acyclic directed graph $G(V, E)$ to represent a group of tasks which should be scheduled. V is the nonempty finite set of nodes. E is the set of directed lines in different nodes. The set of nodes $V=\{\text{task}_i\}$ ($i=1, 2, \cdots, n$) means a group of tasks which should be scheduled. The set of lines $E=\{e_{ij}\}$ means the precedence relationship in different tasks. e_{ij} shows that task_i should finish before task_j starts. If e_{ij} exist in the directed graph, t_i is the pre-order task of t_j, and t_j is the successor task of t_i.

This paper gives the description of a task with four-tuple: task_i (set queue number, contexts matrix, execution time, priority). In considering the precedence relationship in different tasks, the set queue number of task is given as follows:

1) Choose the tasks in $G(V, E)$ where in-degree is 0 and put them into the set queue TQ_1;

2) Decreased the out-degree of all the successor tasks in TQ_1 by 1, then choose the ones that in-degree is 0 and put them into the set queue TQ_2;

3) Repeat 1)-2) until all tasks in $G(V, E)$ are in the corresponding set queues TQ_i ($i=1, 2, \cdots, n$).

Table I *The priority of low-level contexts in assumption*

Low-level contexts	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
Priority	6	3	4	4	1	2	5	3

Table II *The relationship table of context-aware elements*

	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8
t_1	1	1	0	0	1	0	0	1
t_2	1	0	1	0	1	0	0	1
t_3	0	1	0	0	1	1	1	0
t_4	0	0	1	1	1	1	0	0
t_5	0	0	1	0	0	0	1	0
t_6	1	0	0	0	1	1	1	0

The lower set queue number means higher priority of the task queue. For reducing the "BUMP" phenomenon [12] in the scheduling process, the task queue with higher priority will be operated prior at the ones with lower priority.

3.4 Scheduling algorithm based on correlation

Each reasoning task has correlation with several low-level contexts. In this scheduling algorithm, priority of the task will inherit form the contexts it related to. That means a reasoning task may relate to different low-level contexts, the priority of it can be defined from low-level contexts that used for reasoning [12]. For example, assume a task t_{light} is for controlling the light in the restroom. During the daytime, it may have a low priority because the low-level context "light" will not be requested many times; but at night, "light" became the important context for system users. Its priority became higher than in the daytime. By inheriting, the priority of t_{light} also rises.

Suppose some low-level contexts in the smart home environment and their priorities are given in Table I.

We established the relation based on two-dimensional correlation table according to the different corpus C, In Table II, the horizontal direction contains different low-level contexts, and the vertical direction contains the tasks that reasoning from low-level context to high-level context. As the name suggests, correlation is used to characterise the relationship between each reasoning task. Given a visit to low-level contexts corpus $C=(c_1, c_2, c_3, \cdots, c_n)$, and reasoning task queue $TQ=(t_1, t_2, t_3, \cdots, t_n)$. If t_i has relationship to c_j, we mark $t_i \times c_j=1$ at the interaction of the line correspond to t_i and the column correspond to c_j in the relationship table; if not, we mark $t_i \times c_j=0$ in the same place. Given a task queue of high-level context query request, and each of them is related to the sequences of low-level context as follows: $t_1(c_1, c_2, c_5, c_8)$, $t_2(c_1, c_3, c_5, c_8)$, $t_3(c_2, c_5, c_6, c_7)$, $t_4(c_3, c_4, c_5, c_6)$, $t_5(c_3, c_7)$, $t_6(c_1, c_5, c_6, c_7)$.

The relationship graph of corresponding to correlation is shown in Table II.

Priority scheduling algorithm will be built based on relations and the priority of the various elements. From the graph, we know the two-dimensional relation table quantifies the correlation between the tasks. Firstly, group these contextual elements. Secondly, start scheduling management. Algorithm 2 describes the main steps of the scheduling algorithm based on correlation.

The most important step of the algorithm is to calculate the priority Pt_i of task t_i. In this paper, the priority of task will be generated with 5 attributes:

1) Pc_i is the priority which indicates the correlation between task t_i and the current task. Add together the priority of all contexts that both associating with t_i and the current task, and then divide the number of low-level contexts N_{con} which having been added together. This is the basic priority of task t_i:

$$Pc_i = \frac{\sum P_j}{N_{\text{con}}} \tag{6}$$

2) Considered that the same task would have different importance at different time, the priority of it should be fixed automatically. As was talked above, t_{light} is much more important to the user during the night than in the daytime. And the application controlling the light is sure to run much frequently at night. So, when a task t_i has been executed recently, we give it a higher priority Pf_i.

The more times it has been executed, the higher priority it has.

$$Pf_i = 1 - a^{\frac{m_i}{M - m_i}} \quad (7)$$

In Eq. (7), M indicates the number of recent tasks having been executed; m_i is the times that t_i has been executed in recent M executed tasks; $a \in (0, 1)$ is the parameter.

As m_i increasing, the growth rate of Pf_i reduces until approaching 0 (when m_i equal to M). That means t_i has completely held the processor.

Algorithm 2 The main steps of the scheduling algorithm

```
1:Schedule taskSchedule(){
2:    while (task queue is not empty)
3:    Con=head of contexts queue of current task
4:        while (queue Con is not empty)
5:            search Con in task queue
6:            if (task has correlation with Con )
7:                put task into queue T_can
8:            end if
9:        end while
//pick up the tasks having correlation with current task
10:       while (T_can is not empty)
11:           Con_can=head of contexts queue of
              candidate task
12:           while (Con_canis not empty)
13:               if (Con_can is in Con)
14:                   T_can->Pt+=Con_can
                      ->priority
15:                   N_con++
15:               end if
16:           end while // calculate the Pc
17:   m=count (recently executed times of T_can)
18:   T_can->Pt+=(1-a^(m/(M-m)))*C_1
//calculate the Pf
17:   T_can->Pt+=count (wait time of T_can)*C_2
//calculate the Pw
18:   d=count (times T_can has been dropped)
19:           T_can->Pt+=e^d*C_3//calculate the Pd
20:           T_can->Pt+=T_can->E
17:       end while
          //calculate the priority of tasks

18:       if (T_can is not empty)
19:           sort T_can and put the highest one into
          TASK
20:           return (TASK)
21:       end if
22:   end while
23:}
```

3) To solve the problem of "starvation", we give task t_i an addition priority Pw_i increasing with its waiting time:

$$Pw_i = \text{wait}_i \quad (8)$$

wait_i is the indication of the time that task t_i has been waiting for execution. The longer t_i has been waiting, the higher the value of wait_i is. They bear the linear relation.

4) If task t_i has cannot be executed within its deadline, the task will be dropped. When the applications apply for the same reason task again, priority compensation will be given to the task. Define Drop_i as the times task t_i has been dropped. Then we give the Pd_i as:

$$Pd_i = e^{\text{Drop}_i} \quad (9)$$

As with the previous rule, this priority compensation Pd_i is in the purpose of solving the "starvation" problem. Using the exponential function, Pd_i will increase fast as the Drop_i increasing. In this way, after a less times of dropping, t_i will have a high priority with Pd_i.

5) Another thing must be considered is the "emergency" (such as alarm, doorbell and phone ring). We give these emergencies a special priority parameter E.

We give E a high value to ensure that the emergency can have the highest priority.

With these 5 attributes, the reasoning task t_i will generate its priority Pt_i for scheduling as follow:

$$Pt_i = Pc_i + C_1 \times Pf_i + C_2 \times Pw_i + C_3 \times Pd_i + E \quad (10)$$

In Eq. (10), C_1, C_2, C_3 are the weight factors. They indicate the weight of each addition priority contribute to the task priority.

IV. Experimental Analysis

4.1 Basic experiment scene

In this section, we present experimental evaluation of the scheduling algorithm described above.

This experiment scene is based on the smart space model [13]. In the smart space, various context-aware applications work in cooperation. In this paper, we assume that 50 contexts concerned by the user are distributed in the space, 30 potential task requested by system applications might occur in random. Different from the model [14], in this experiment we

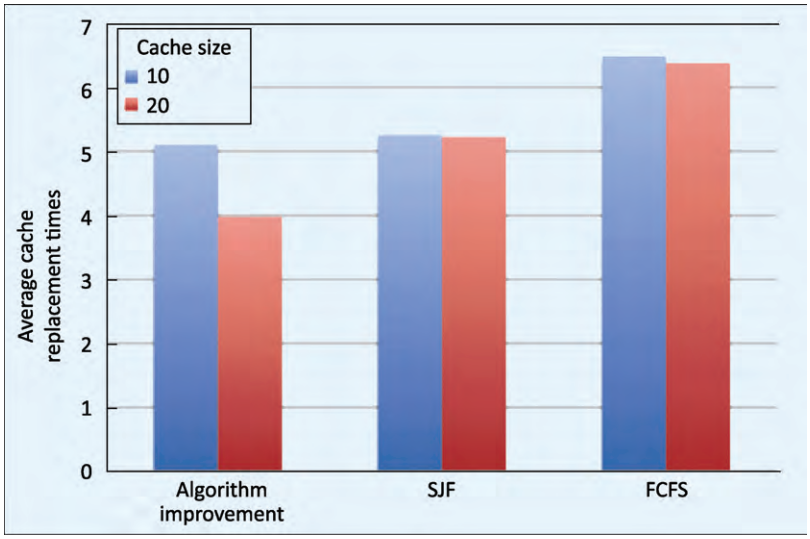

Fig.3 *Performance comparison with different cache sizes*

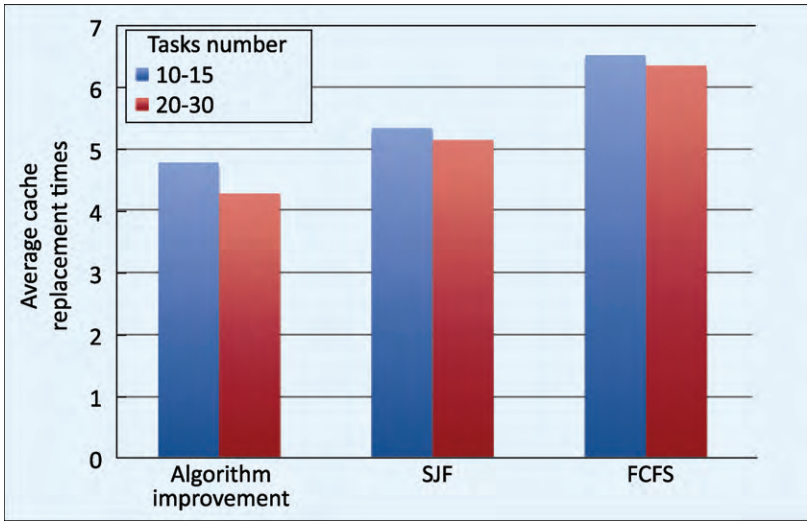

Fig.4 *Performance comparison with different quantity of tasks*

Table III *The range of parameters in the experiment*

Parameters	Range
Contexts number related to every task	5-10
Refresh period of every context	30-300 s
Transmission cost of every context	0-3 s

mainly concern with the transmission between reasoned and context sensors. We will compare this algorithm improvement with two classic real-time scheduling algorithms: FCFS and SJF in the main index that data transmission times between reason unit and receiver unit. Parameters related to the system will be ge1nerated randomly according to normal distribution.

4.2 Comparison of performance with different algorithms

Each time of experiments, the FCFS, SJF and the algorithm improvement will schedule a tasks queue separately. In order to simplify the experiment, we assume that new task will not be inserted into the queue in every scheduling process. Every low-level context tasks request for will be stored in the cache. If the cache is full, replace the context of lowest priority with the new one.

The simulation demonstrates that when the cache size is small, the performance of the algorithm improvement is nearly the same to the SJF and FCFS. As shown in the Table IV, in average the algorithm improvement consumes 2%-5% less cache replacement times than that of the SJF and 14%- 20% less than that of FCFS. The main reason with small memory size cannot afford enough long time to store each context, and the improvement will not work completely. In this situation, SJF performs better than FCFS, because SJF is advanced in cache replacement by calculating the priority of the context.

Figure 3 shows that algorithm improvement performing better in large cache environment. When cache volume change from 10 to 20, the replacement times of cache reduced by 22%-24% with algorithm improvement, while the other two algorithms have little improvement. The reason is that the performance of SJF and FCFS do not mainly rely on the cache hits.

Another main analysis is the effect of the quantity of the tasks in the request queue. We assume that all tasks will be executed before deadline, and get the performances of these algorithms showed in Figure 4. As the tasks number in the queue increasing, the algorithm improvement reduces the replacement times of cache.

The experiment results indicate that this scheduling algorithm improvement performance is better than the classic real-time scheduling algorithms in large cache and mass tasks system. In this environment, the algorithm improvement can obviously reduce the replace-

ment times of cache. In other word, context in the smart space can be more effective within its refreshing period with the lower transmissions cost.

V. Conclusion and Future Work

This paper discussed a context-based model of smart home system. Context reasoning needs an effective strategy to manage reasoning tasks in pervasive environment. Based on context correlation, we proposed a context priority-scheduling algorithm in order to improve the average utilization of low-level context in the pervasive computing environment. Simulation experiments show that this algorithm can reduce the times of cache replacement and the times of data transmission. In order to further reduce the cost of context transmission, our next research work is to optimise the algorithm of cache replacement and make coordination with the scheduling algorithm proposed in this paper.

Acknowledgement

This work was partially supported by the National Natural Science Foundation of China under Grant No. 61103115; the Hunan Provincial Natural Science Foundation of China under Grant No. 11JJ4058; and the Scientific Research Fund of Hunan Provincial Education Department under Grant No. 11A041.

References

[1] LIN Xin, LI Shanping, YANG Zhaohui. Freshness-Aware Real-Time Scheduling Algorithm for Context Reasoning[J]. Journal of Electronics and Information Technology, 2009, 31(5): 1185-1188.

[2] BACCOUCHE L, ELEUCH H. RT-DBP: A Multi-Criteria Priority Assignment Scheme For Real-Time Tasks Scheduling[J]. Applied Mathematics and Information Sciences, 2012, 6(2): 383-388.

[3] ZHOU Benhai, QIAO Jianzhong, LIN S K. Research on Parallel Real-Time Scheduling Algorithm of Hybrid Parameter Tasks on Multi-Core Platform[J]. Applied Mathematics and Information Sciences, 2011, 5(SI): 211-217.

[4] NOAH S W. A System Architecture for Context-Aware Mobile Computing[D]. New York: Columbia University, 1995.

[5] INTILLE S S, KENT L, EMMANUEL M, *et al*. Using a Live-in Laboratory for Ubiquitous Computing Research[C]// Proceedings of the 4th International Conference on Pervasive Computing . May 7-10, 2006, Dublin, Ireland, 2006: 349-365.

[6] WANG Jingxiao, ZHENG Hong. Research on Context-Aware Scheduling Algorithm Based on Correlation in Smart Home Environment [C]// Proceedings of the 26th IEEE International Parallel & Distributed Processing Symposium. May 21-25, 2012, Shanghai, China, 2012: 2312-2315.

[7] ZHANG D G, ZHANG X D. A New Service-Aware Computing Approach for Mobile Application with Uncertainty[J]. Applied Mathematics and Information Sciences, 2012, 6(1): 9-21.

[8] RANGANATHAN A, CAMPBELL R H. An infrastructure for context-awareness based on first order logic[J]. Personal & Ubiquitous Computing, 2003, 7(6):353–364.

[9] PARK J S, MOON M, HWANG S, *et al*. CASS: A Context-Aware Simulation System for Smart Home[C]// Proceedings of the 5th ACIS International Conference on Software Engineering Research, Management and Applications 2007 (SERA 2007): August 20-22, 2007. Busan, Korea, 2007: 461-467.

[10] FU Xiufen, LV Zhande, HUANG Weiqiang. Research on Context-Aware Scheduling Algorithms in Pervasive Computing Environments [C]// Proceedings of 2010 5th International Conference on Pervasive Computing and Applications (ICPCA): December 1-3, 2010. Maribor, Yugoslavia, 2010: 277-282.

[11] JURMU M, PERTTUNEN M, RIEKKI J. Lease-Based Resource Management in Smart Spaces [C]// Proceedings of the Fifth Annual IEEE International Conference on Pervasive Computing and Communications Workshops 2007 (PerCom Workshops'07): March 19-23, 2007. White Plains, NY, USA, 2007: 622-625.

[12] MENG Xianfu, DONG Feng. Dynamic Scheduling Strategy of Dependent Tasks in Peer to Peer Network Environment[J]. Computer Integrated Manufacturing Systems (CIMS), 2011, 17(9): 1929-1937.

[13] ZHOU Huan, LI Jing, FENG Yulin. A Low-Cost Automatic Data Hoarding Algorithm for Mobile Environment[J]. Chinese Journal of Software, 2002, 13(10): 1962-1968.

[14] WANG Xiaohang, DONG J S, CHIN C Y, *et al*. Semantic Space: An Infrastructure for Smart Space[J]. IEEE Pervasive Computing, 2004, 3(3): 32-39.

Biographies

ZHENG Hong, Associate Professor with the Department of Computer Science and Engineering, East China University of Science and Technology, China. She received her Ph.D. degree in computer software and theory from Chinese Academy of Sciences, China in 2003. She is a visiting scholar with the Department of Computer Science and Engineering, University of California, USA. Her research interests include pervasive computing, system modelling and analysis, context-aware. Email: zhenghong@ecust.edu.cn

PAN Li, Associate Professor with the Department of Information and Communication Engineering, Hunan Institute of Science and Technology, China. He received his Ph.D. degree in computer applied technology from Tongji University, China in 2009. His research interests are in petri nets, workflows and computational intelligence. Email: panli.hnist@gmail.com

WANG Jingxiao, received his M.S. degree in computer software and theory from East China University of Science and Technology, China in 2013. His research interests include context-aware and scheduling algorithm.