Thesis for the Degree of Master of Computer Engineering

Design and Implementation of a Partitioning Scheme in Sensor Network Nodes

Dae Hoon Kim

Department of Computer Engineering
Graduate School
Kyung Hee University
Suwon, Korea

December, 2013

Design and Implementation of a Partitioning Scheme in Sensor Network Nodes

Dae Hoon Kim

Department of Computer Engineering
Graduate School
Kyung Hee University
Suwon, Korea

December, 2013

Design and Implementation of a Partitioning Scheme in Sensor Network Nodes

by
Dae Hoon Kim

Supervised by
Prof. Jinsung Cho, Ph.D

Submitted to the Department of Chemistry
and the Faculty of the Graduate School of
Kyung Hee University in partial fulfillment
of the requirement for the degree of
Master of Computer Engineering

Dissertation Committee :

Prof. Intae Ryoo, Ph.D.....

Prof. Eui_Nam Huh, Ph.D.....

Prof. Jinsung Cho, Ph.D.....

Acknowledgements

It has been 3 years form I came into MESL lab of KHU. Now, my graduation time has came unexpectedly. Here I will thank all the persons who gave so many helps during this 3 years to let me grow up form a simple and immature guy.

Firstly, sincerely thank for my professor Jinsung Cho who bring up me selflessly. During this 3 years, he let me know not only the posture how to be a researcher but also many conditions required in the life. When I come upon the stage, I will live with all what professor has said. At the same time, I will thank for professor Intae Ryoo and professor Eui_Nam Huh who review my paper prudentially even in busy schedule.

No matter when and where, I will still remember the helps from lab members both in joy and in sorrow. First, thanks for Phd Choongyong Shin who listen to my sharing and give me praise liberally, Phd BeomSeok Kim who give continuous good news all the time, and also for Phd Zilong Jin who is affectionate and with so much advantages I can learn. And thanks for sangbae shin who select a postive way and give me help even just a little, Byoung Seon Kim who can create evething, ByungSik Yoo who with angel's heart, a!! SeonWoo Yoo. Seung Jun Oh who can control people with his word, active and pleasant Weidong Su who can accept all my jokes. Yoon Jung Han who is the only one share the same age with me, Sunghyun Kim who work hard by himself silently, Kun Ryun Cho who with lively personality and also love sport as me, Fanhua Kong who work hard more in the same time. Jong-hyun Choi who with fourth dimensional personality, and special thank for my love Jieun Lee who sit beside of me, work hard and also seriously help me even when tied and ill. Also thanks for Jehyun Yoo the new guy who will go USA and back for joining our lab.

Finally, I will also thanks for the fiends form other lab who gave me helps, such as YoungRok Shin, Kijung Kim, Dongha Kim.

last but not least, I want unfeignedly say thanks to my mother who take me to this beautiful world, without her raise and teach I can't get today's achievement. mom! thank you and I love you.

Design and Implementation of a Partitioning Scheme in Sensor Network Nodes

by

Dae Hoon Kim

submitted to the Department of Computer Engineering
on December 9, 2013, in partial fulfillment of the
requirements for the degree of
Master of Computer Engineering

Abstract

Due to rapid development of embedded system technologies, miniaturization of low-power sensor nodes becomes possible. Based on these improvements of sensor nodes, several sensor applications can run on a single sensor hardware and all of the applications in sensor network should be guaranteed the real-time requirements and independence. To satisfy these requirements, the virtualization concept was proposed. However, the virtualization cannot be applied to sensor nodes because it requires powerful computing resources. To solve this problem, the partitioning scheme which is categorized into temporal partitioning and spatial partitioning has been proposed. In this paper, we design and implement temporal partitioning scheme for sensor nodes and validate that our implementation guarantees minimum execution time of each partition.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In last few years, embedded system is dramatically developed and it is used in various areas such as airplane, air conditioner, television, and sensors. Especially, development of sensor devices makes it possible that sensor network is used in various area such as military, u-healthcare, and monitoring system.

In general, a single application runs on a single hardware because sensor nodes have limited hardware resources. In recent years, however, sensor devices are developed rapidly and it makes possible that various applications run on a single sensor device. Based on these improvements of sensor nodes, many sensor applications can run on a single sensor hardware, and all the sensor applications should provide the reliability and independence at the same time. For example, airplane system should operate properly because the whole airplane system may cause serious problems if one of applications makes error.

To guarantee these requirements, the virtualization scheme has been proposed. The virtualization supports various OS (Operating System) on each virtual machine in a single hardware. However, the virtualization scheme requires powerful computing resources. As mentioned above,

sensor nodes have limited resources and it is not suitable that we directly apply virtual machine to sensor nodes. To handle this problem, the partitioning scheme has been proposed. The partitioning scheme consists of two parts: temporal partitioning and spatial partitioning [1]. Temporal partitioning divides the time resources of all applications and guarantee the minimum execution time. Contrastively, spatial partitioning divides physical memory of applications and it protects memory space of each application using memory protection of CPU. While the partitioning scheme is similar to concept of virtualization, it requires lower hardware resources than virtualization. However, existing spatial partitioning is difficult to adapt to sensor nodes because most CPUs on sensor nodes does not support memory protection.

In this paper, we design the partitioning architecture that guarantee minimum execution time using temporal partitioning with a limited performance of hardware and implement the partitioning on KHIX using ATmega128.

The rest of the paper is organized as follows: Section 2 discusses background and existing studies of partitioning scheme. In Section 3, we describe the architecture of partitioning and its performance is evaluated via implementation in Section 4. Finally, Section 5 concludes this paper.

Chapter 2

Background

2.1 Virtualization

Virtualization provides independence of each virtual machine. Figure 2-1 illustrates concept of virtualization. The virtualization can be categorized into two schemes: full-virtualization and para-virtualization. Full-virtualization has several virtual machines which have different guest OS that does not need to modify. It should be provided virtualization function of CPU. Therefore, guest OS requests to control the hardware to

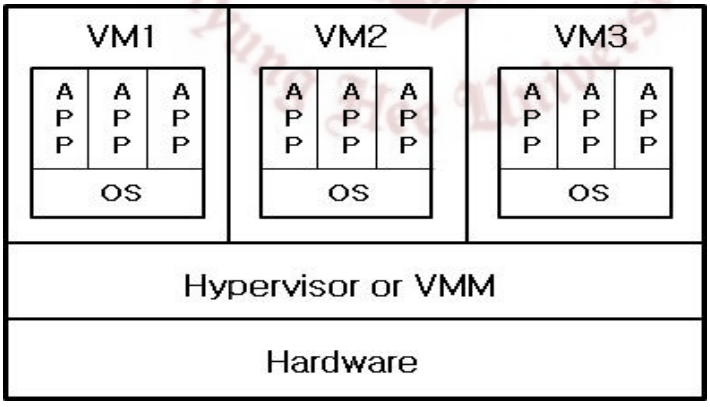

Figure 2-1: Concept of virtualization

CPU and the CPU delivers the request to hypervisor which can control the hardware. The strength of full-virtualization is that guest OS does not need to change. On the other hand, guest OS of para-virtualization requests to control the hardware to hypervisor directly. Therefore, para-virtualization which should modify its guest OSs can provide higher performance than full-virtualization. Meanwhile, these virtualization schemes are available on hardware based on excellent performance.

2.2 Partitioning

While partitioning scheme is similar to virtualization scheme, it provides independent spaces of several applications that are divided into each partition with lower computing resources than virtualization. Therefore, partitioning scheme is more suitable than virtualization to apply sensor network. Partitioning scheme can be categorized into two schemes: temporal partitioning and spatial partitioning. Spatial partitioning means that physical memory of one partition is not affected by the other partitions.

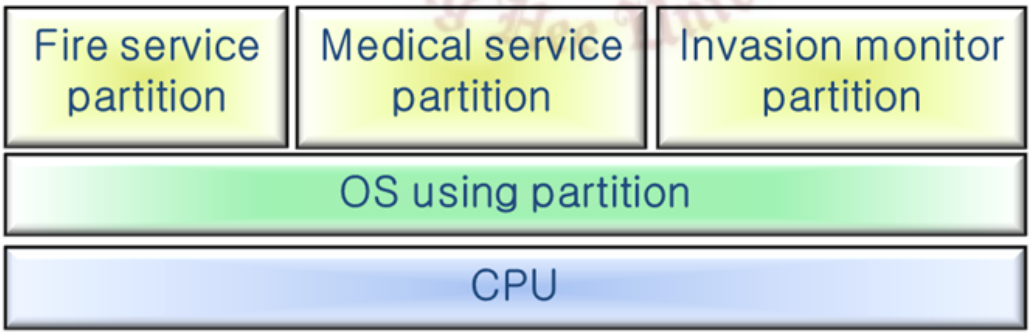

Figure 2-2: Partitioning scheme

On the other hand, temporal partitioning means that each process is allocated time resource to guarantee independence. Figure 2-2 represents the partitioning scheme. This partitioning scheme operates on a single hardware like as a lot of independent hardware. This scheme is used for embedded system such as car and airplane system based on ARINC 653 standard [2].

2.3 ARINC 653

Avionics system is consists of various kind of electronic systems that perform the different work. These electronic systems are integrated into IMA (Integrated Modular Avionics). IMA is performed on a single computing environment because electronic equipment is very heavy weight and variety. In this situation, reliability of each applications and independence are very important. To guarantee this problem, ARINC 653 standard that define the partitioning scheme in avionics system was proposed [2-4]. ARINC 653 standard was written not only partitioning scheme but also whole avionic systems

2.4 KHIX

KHIX (OS developed by Kyung-Hee University) is the OS that can provide the extension and reconfiguration. And it is medical sensor OS providing API based on POSIX [5]. Embedded system of sensor network needs the composition which is suitable to purpose because of many

limitations that are CPU performance, memory and consumption of electronic power. Figure 2-3 shows a block diagram of KHIX. Each function allows easy extension by making component and easy transplant the other process through HAL (Hardware Abstract Lager). Also it allows to develop the program which easily provides the API based on POSIX.

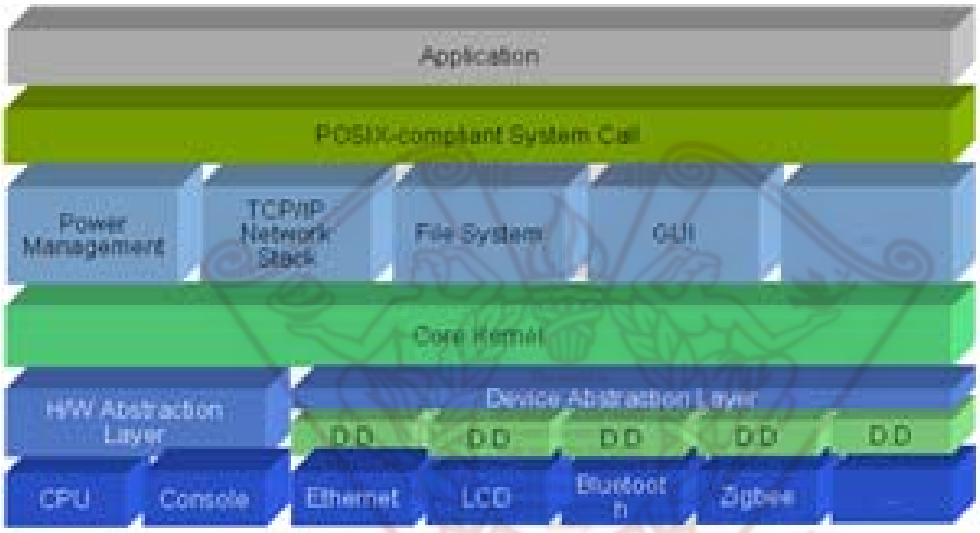

Figure 2-3: Architecture of KHIX

2.5 ATmega128

We design the partitioning scheme in the sensor network. Sensor network requires small capacity of hardware and high-speed microcontroller. The ATmega128 fulfills these requirements. The ATmega128 is manufactured by using ATmel's high-density nonvolatile memory and provide up to 16

MIPS throughput at 16MHz and on-chip 2-cycle multiplier. The device has UART0 and UART1 communication ports for serial communication under UART environment and uses USB ISP cable in order to download programs to the ATmega128 processor. The device has 128kbytes of Flash, 4kbytes SRAM. And it is supported with C compilers [6-7].

2.6 Related work

To provide performance of integrated system and reliability on sensor nodes, independence of each application that runs on a single hardware has become more important. For this reason, there exist studies to guarantee requirements of sensor nodes.

In recent years, study of vehicle mounted infotainment system which simultaneously operates multiple applications tries to apply virtualization and partitioning scheme to provide independence of each application that executes on a single vehicle. The authors of [8] proposed the partitioning scheme that consists of two non-virtual partitions and single virtual partition. They suggest the performance evaluation using various platforms that run on the each partition on a single hardware. In the result, they verify that overload of one partition does not affect the other partition.

COS (Core OS of DECOS project) has the partitioning scheme that consists of temporal partitioning and spatial partitioning [9]. Temporal partitioning has two-level scheduling. First level scheduling divides the several time slots and it allocate the partitions. When system is divided into several partitions, allocate the fixed scheduling with pre-defined

weight. Second level scheduling that does dynamic scheduling with workload of each partition is event or event handler of each partition. Spatial partitioning is to provide memory protection through the MMU (Memory Management Unit) which control the CPU to access the memory. They propose this partitioning scheme that each partition uses the divided resources.

Temperature-aware partitioning embedded system was also proposed. High temperature negatively affects reliability as well the costs of cooling and packaging [10]. This system is divided into two partitions that consist of "cool" task and "hot" task. If "hot" tasks are executed consecutively, the system is negatively affected. Therefore, the system always checks the temperature of board. Then the partition scheduler dynamically determines the executing partition. Experimental results show that task partitioning algorithms can effectively reduce the peak temperature.

These studies assume that they developed with powerful computing resources. However, sensor nodes in sensor network have limited hardware resources and it cannot be applied existing partitioning schemes. To handle this problem, we design the partitioning scheme on the sensor OS which operates on sensor node with ATmega128 that has limited performance of hardware.

Chapter 3

Design of Partitioning on KHIX

3.1 Temporal partitioning

Temporal partitioning requires that it does not affect CPU usage (process resource) of each divided partition. In this paper, we design the independent scheduling scheme of each partition.

As shown in Figure 3-1, KHIX system has various applications on a single node with non-temporal partitioning. Each application has several threads and they have different weight. KHIX system uses the priority-based RR (Round Robin) scheduler which fairly schedules by priority of threads. If a thread has the highest priority and many workloads, it may monopolize the usage of CPU. To solve this problem, we propose partitioning scheme that several threads in divided partition are performed independently. In addition, we design the temporal partitioning using integrated management partition scheduler to guarantee minimum execution time of each application.

Figure 3-2 shows KHIX with proposed temporal partitioning and integrated management partition scheduler. Each partition has different scheduler and consists of several threads. Each partition is allocated the

fixed weight by user, and the partition scheduler divides the time resources depending on weight of partition. By this design of proposed scheme, it can prevent monopoly of CPU and guarantee minimum execution time of each partition(application).

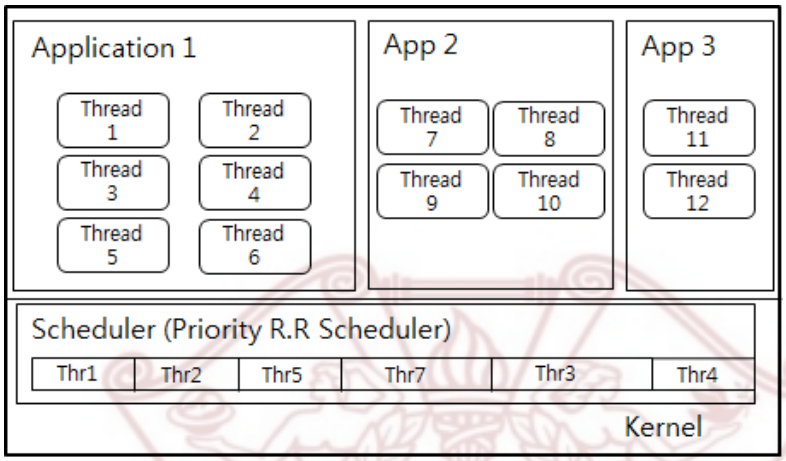

Figure 3-1: KHIX without temporal partitioning

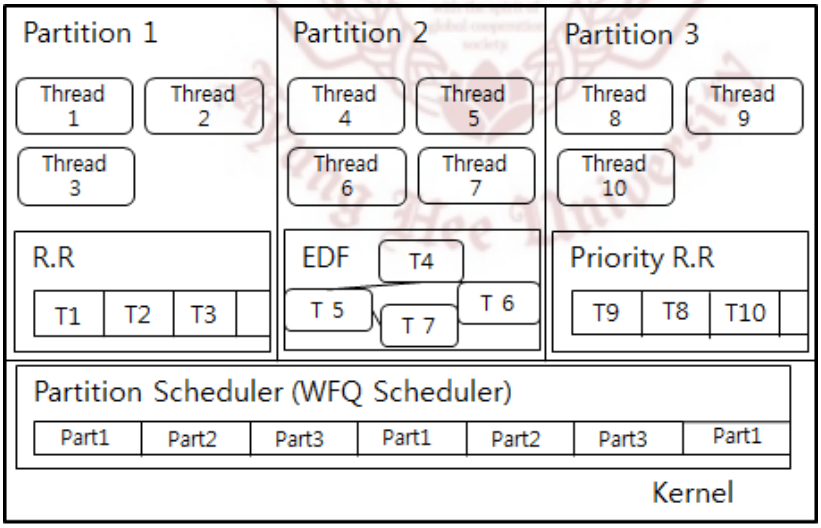

Figure 3-2: KHIX with proposal temporal partitioning

3.2 Partition

One partition consists of several threads. When system is started, the weight of each partition is allocated by user, and scheduling is decided with requirement of partition. Each partition doesn't affect to the other partition. Therefore error of one partition doesn't affect the whole system. For example, Figure 3-2 shows KHIX with proposed temporal partitioning. In case of the partition 1, it uses RR scheduler which fairly allocates the time resources and partition 2 uses EDF (Earliest Deadline First) scheduler which needs to guarantee the dead line of the thread in real time. Scheduler of partition 3 is priority-based RR as mention above. These schedulers are executed independently.

3.3 Partition scheduler

Kernel has partition scheduler which schedules partitions and each partition has its own scheduler. Partition scheduler uses the WFQ (Weight Fair Queue) scheduler that guarantees the execution time with weight of partition. It is similar to RR scheduler which fairly divides the execution time of threads. However, it allocates different time resources to each partition based on weight of partition and it is a scheduler adaptable to dynamic environment. In addition, it can provide prevention of CPU monopoly, guarantee of independence and minimum execution time of each partition. This partition scheduler cannot be applied to hard real time sensor network because this scheme guarantees minimum execution time

of all threads, therefore this scheme is hard to provide deadline of all threads.

At the start of the system, each scheduler initializes variables and registers alarm signal which is periodically called every 10ms. After system initialization time, partition scheduler allocates the time resources to each partition based on weight of partition. Each partition executes its member threads by using its own scheduler. When alarm signal is called, time resource of thread and partition is decreased. Scheduler of partition tries to find available thread in executing partition when time of thread become 0 and time of partition is more than 0. If it finds the available thread, it loads thread to CPU. Otherwise, partition scheduler change to next partition and allocates the time resources using weight of partition. If any partition is not available, CPU executes the idle thread. Figure 3-3 shows sequence diagram when thread is expired.

First, partition scheduler check time of thread. When thread is expired, partition scheduler request schedule to in-partition scheduler. After find available thread, context switching is happened. Figure 3-4 shows sequence diagram when partition is expired. Partition scheduler check time of partition. If partition is expired, partition scheduler try to find available partition. After find available partition, partition scheuler give the time and request schedule to that partition.

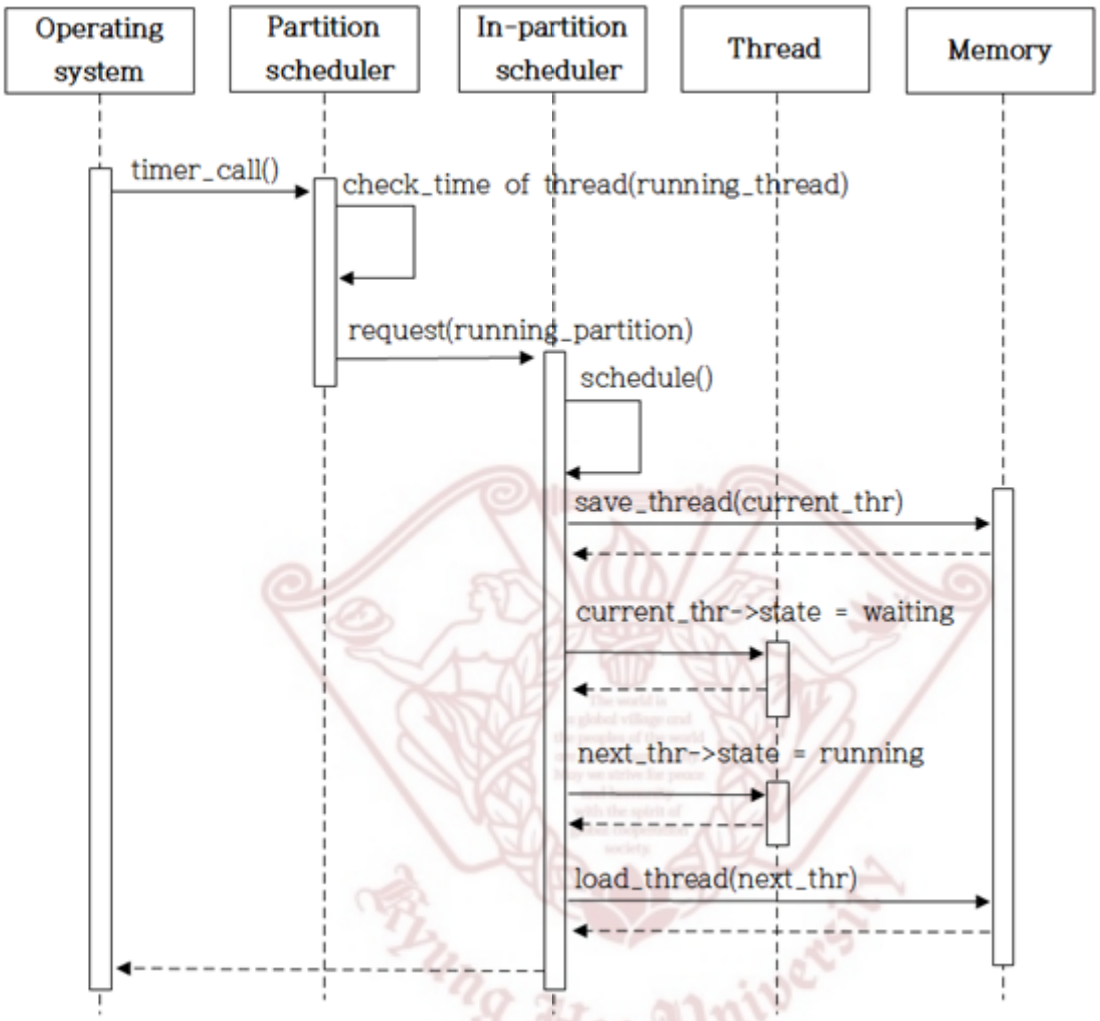

Figure 3-3: Sequence diagram (thread is expired)

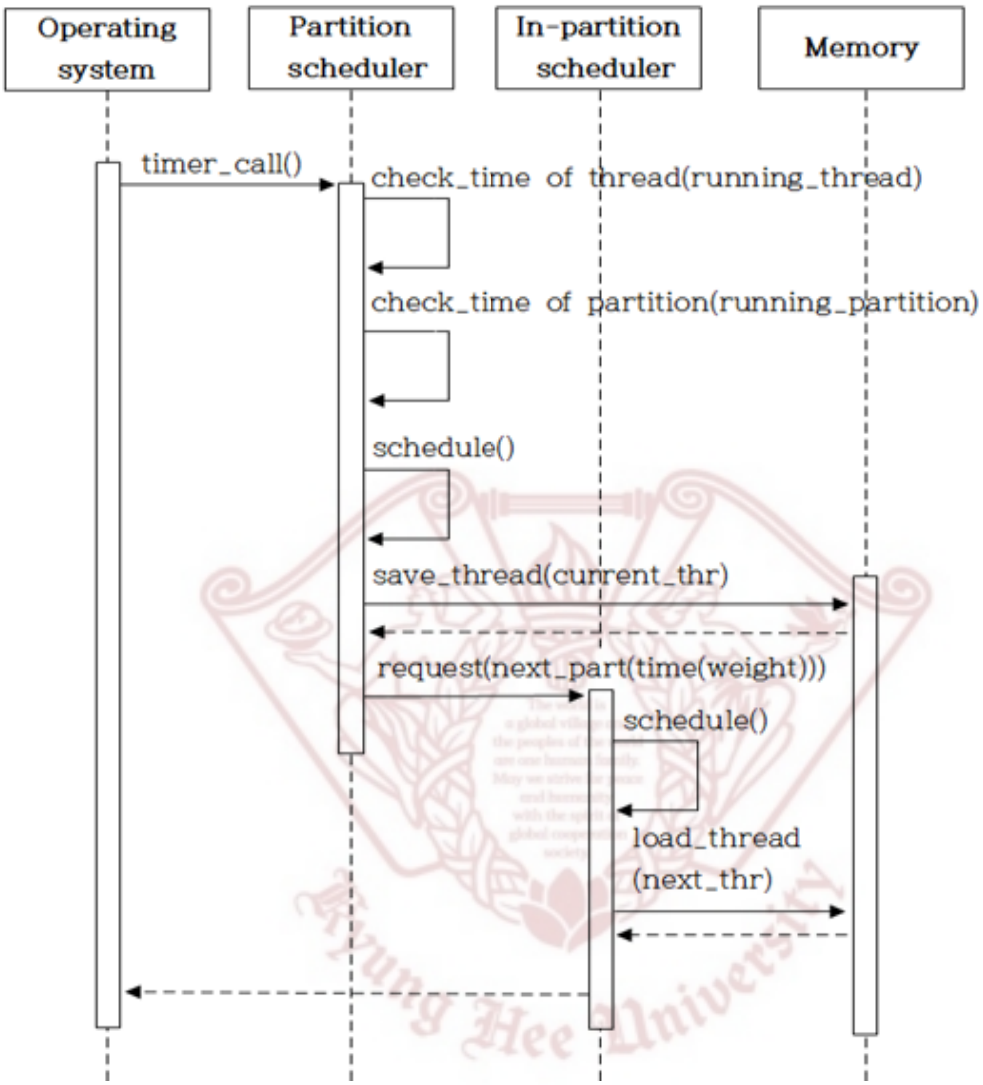

Figure 3-4: Sequence diagram (partition is expired)

Chapter 4

Implementation

4.1 Implementation on ATmega128

We implemented proposed temporal partitioning scheme using partition scheduler. Figure 4-1 shows Zigbex-Ⅱ node which is infrastructure to develop proposed scheme based on ATmega128.

We perform experiment which verifies the guaranteeing minimum execution time using temporal partitioning. The experiment was proceeded during 1sec (1000ms) and time quantum of OS is 20ms. Detail environment of experiment is shown in Table 4.1.

In experiment, five threads have different priority such as High, Middle,

Figure 4-1: Zigbex-II node with ATmega128

Table 4.1 Experiment environment

Application (Partition)	Thread No.	Priority of thread	CPU usage (weight)
Application 1 (Partition 1)	thread 1	High	30%
	thread 2	Low	
Application 2 (Partition 2)	thread 3	Middle	30%
	thread 4	High	
Application 3 (Partition 3)	thread 5	Low	20%

and Low. These priorities and usage of CPU can be modified by users, and we set the data as shown Table 4.1. We compare the performance of KHIX with proposed scheme with KHIX with priority-based RR scheduler. The experiment validates that guarantee the minimum execution time when error occur.

4.2 Validation of partitioning

Figure 4-2 is results of experiment on KHIX using priority R.R. All threads take the time resources of 80% usage of CPU. We assume that the thread 3 occurs the infinite loop error and the thread 1 and the thread 4 have higher priority than thread 3. Therefore, the thread 1 and 4 do not affected by error of the threads 3. However, the thread 2 and the thread

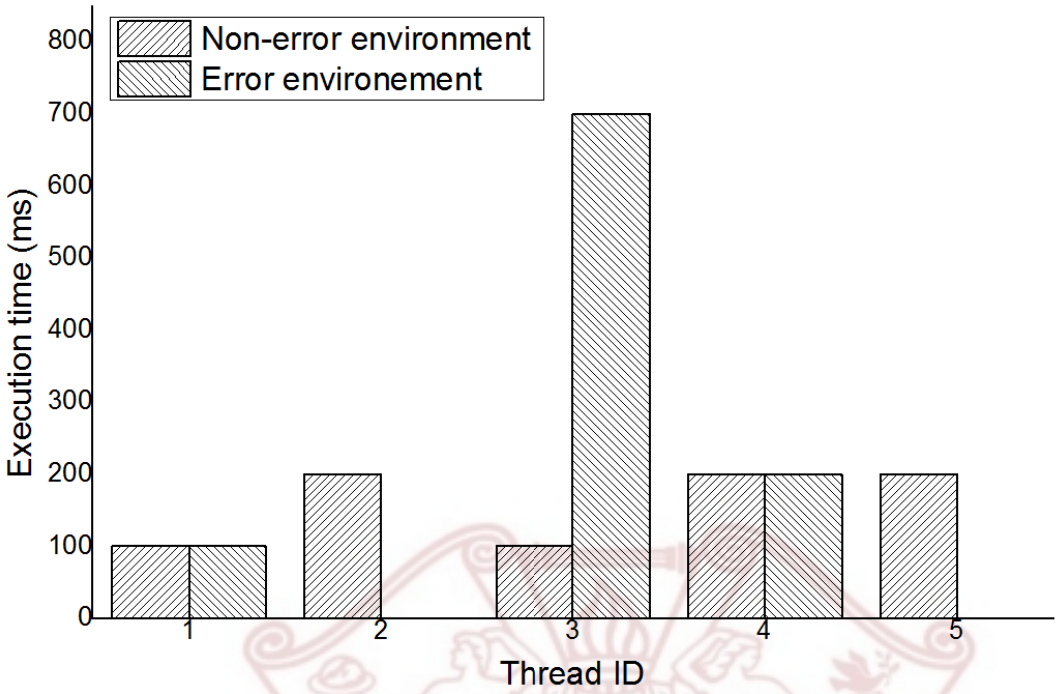

Figure 4-2: Execution time of threads using priority R.R

5 have lower priority than the thread 3. It cause problem with minimum execution time of threads because infinite loop error always make state of thread 3 'Ready' or 'Execute'. As the result, threads which have lower priority than the thread 3 cannot guarantee the minimum execution time.

Figure 4-3 illustrates result of execution time on KHIX with proposed temporal partitioning scheme. As shown the result of non-error environment, all threads are fairly distributed. Meanwhile, thread 2 and thread 5 which have lower priority than thread 3 are guaranteed the minimum execution time when thread 3 occurs error. The thread 3 executes during about 180ms which is longer than thread 3 of non-error environment because the thread 3 additionally use 20% of CPU resource

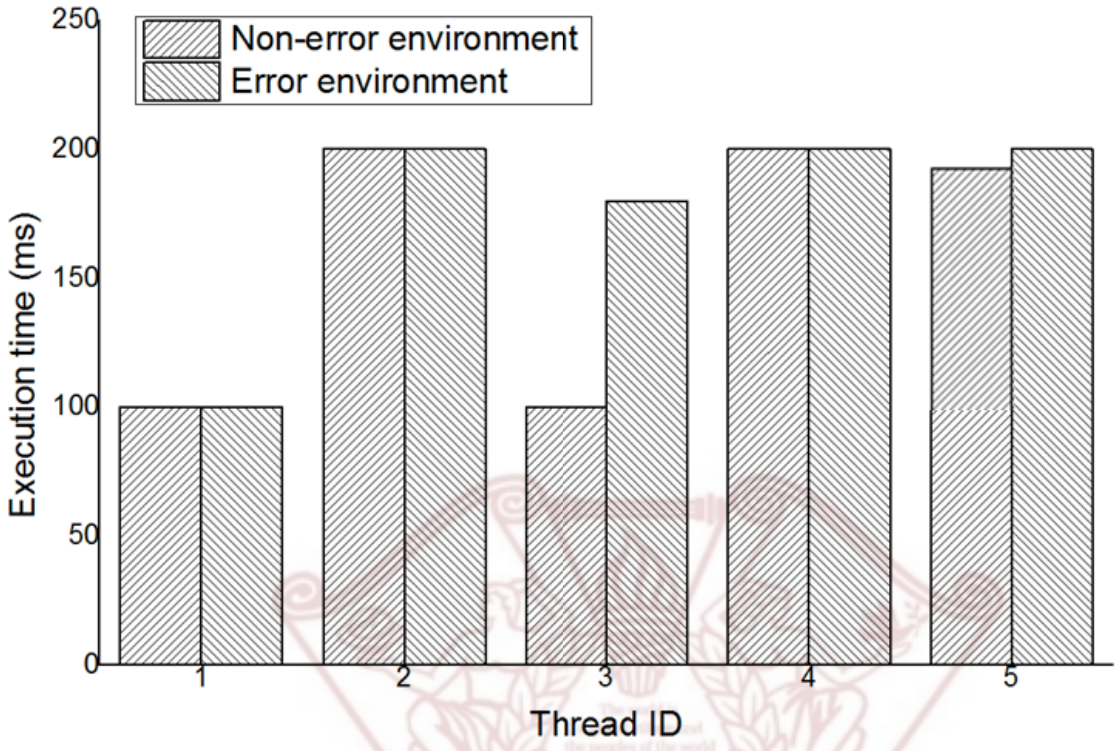

Figure 4-3: Execution time of threads using partitioning

when thread 3 has infinite loop error. As the result, proposed temporal partitioning scheme can guarantee the minimum execution time of each thread.

4.3 Performance Analysis

Proposed partitioning scheme provide that guarantee the minumum execution time of each partition. however, partitioning scheme has overhead more than KHIX system. So, we analyse the overhead of KHIX and partitioning system. Table 4.2 shows variables of overhead. Overall overhead consists of context switching, scheduling and interrupt. In Eq. 4.2, overhead of context switching is sum of overhead of save, load and execute.

$$\triangle all = \triangle con + \triangle sche + \triangle timer \tag{4.1}$$

$$\triangle con = \triangle Tsave + \triangle Tload + \triangle Texe \tag{4.2}$$

$$\triangle exe(one) = \frac{Q}{T}\triangle timer \tag{4.3}$$

where Q is the time quantum; T is the time to tick count; $\triangle timer$ is the overhead of execute. $\triangle exe(one)$ is execution overhead of one thread until it is expired.

$$\triangle exe(cycle) = N(\triangle exe(one)) + (N-1)(\triangle con + \triangle sche) \tag{4.4}$$

In Eq. 4.4, $\triangle exe(cycle)$ is execution overhead when all threads are called one time. $\triangle exe(one)$ is called N times and overhead of context switching and scheduling is called between two threads.

Table 4.2 Variables of overhead

Symbol	Explanation
N	Number of threads
T	Time to tick count
Q	Time quantum
P	Number of partitions
L	Lifetime of threads
$\triangle Tsave$	Overhead to save the thread to memory
$\triangle Tload$	Overhead to load the thread to memory
$\triangle Texe$	Overhead to execute the thread
$\triangle timer$	Overhead to count a tick in a periodic time interrupt
$\triangle sche$	Overhead to choose a thread
$\triangle partsche$	Overhead to choose a partition

$$L(cycle) = \sum_{k=0}^{N} L(exe(Threadk)) \tag{4.5}$$

$$L(all) = \sum_{k=0}^{N} L(Threadk) \tag{4.6}$$

$$\triangle exe(all_{non}) = \frac{L(all)}{L(cycle)} \triangle exe(cycle) \tag{4.7}$$

$L(cycle)$ is lifetime of all threads that are called one times and $L(all)$ is lifetime of all threads. So, $\triangle exe(cycle)$ is called $\frac{L(all)}{L(cycle)}$ times which is overhead of non-partitioning system.

Overhead of partitioning system is as follows:

$$\triangle exe(all_{part}) = \frac{L(all)}{L(cycle)}(\triangle exe(cycle) + (P-1)(\triangle partsche)) \quad (4.8)$$

Best case and worst case of difference of two systems is

$$\triangle diff_{best} = \frac{L(all)}{L(cycle)}((P-1)(\triangle partsche)) \quad (4.9)$$

$$\triangle diff_{worst} = \frac{L(all)}{L(cycle)}((P-1)(\triangle partsche + \triangle con)) \quad (4.10)$$

Best case of overheadof scheduling and context swithing are about 0.12 μs and worst case is 0.3 μs. These values are too small to effect to system.

In addition, proposed partitioning scheme is developed on the KHIX system. In sensor network which has limited resources of sensor nodes, code size is very important. The size of KHIX system using priority-based RR scheduler is 2,057 bytes which takes 50.2% of SRAM and 5,498 bytes which takes 4.2% of flash memory. Similarly, the size of KHIX system with proposed temporal partitioning scheme is 2,390 bytes which takes 58.3% of SRAM and 7,918 bytes which takes 6.0% of flash memory. This result means that the proposed temporal partitioning scheme has capacity of multi-application execution and it is useful to apply sensor network to real world.

Chapter 5

Conclusion

In this paper, we propose the temporal partitioning scheme to prevent the CPU monopoly, guarantee of independence and minimum execution time of each partition. To validate performance of proposed scheme, we also implement proposed scheme on ATmega128 board and measure execution time of each thread. Results of experiment shows that proposed scheme can guarantee minimum execution time of all thread in the system. Moreover, system which is applied proposed scheme has low capacity and it can be used in general purposed sensor nodes. As a result, proposed partitioning system is suitable to sensor network in real world.

Bibliography

[1] B. Leiner, M. Schlager, R. Obermaisser, and B. Huber, "A Comparison of Partitioning Operating System for Integrated Systems," In Proc of S AFECOMP, pp. 342-355, 2007.

[2] S. H.VanderLeest, "ARINC 653 HYPERVISOR," In Proc of Digital Avion ics Systems Conference, pp. 5.E.2-1 - 5.E.2-20, 2010

[3] S. Han, H. Jin, "Virtualization-based ARINC 653 Partitioning for Avioni cs Software," In Proc of KCC, 2011

[4] S. Han, H. Jin, "Kernel-Level ARINC 653 Partitioning for Linux," In Pr oc of ACM Symposium on Applied Computing, pp. 1632-1637, 2012

[1] Y. Baek, J. Cho "KHIX : A Scalable and Reconfigurable Embedded Sy stem Operating" in KCC, 2007

[5] M. Kang, S. Chom, J. Kim, U. Chong, "Implementation of Non-Stringed Guitar Using ATMEGA128," In Proc of IFOST, pp. 585-588, Oct. 200 7

[6] X. K. Pham, D. Q. A. Vo, N. H. Nguyen, T. P. Cao, "PID-Neural Contr oller based on AVR Atmega 128," In Proc of ICARCV, pp 1573-1576, Dec. 2008.

[7] S. Han, J. Seok, H. Jin, "A Partitioning Scheduling Scheme to Support Efficient Mixed Partitoning," In Proc of KIISE, 2013

[8] J. Craveiro, J. Rufino, F. Singhoff "Architecture, Mechanisms and Scheduling Analysis Tool for Multicore Time-and Space-Partitioned Systems," In Proc of ACM SIGBED Review. Vol. 8, No. 3, pp 23-27, Sep. 2011

[9] Z. Wang, S. Ranka, P. Mischra, "Temperature-aware Task Partitioning for Real-Time Scheduling in Embedded Systems,", In Proc of VLSID, DOI = 10.1109/VLSID.2012.64