Bulletin of the College of Engineering, N.T.U., No. 90, February 2004, pp. 75–82
國立臺灣大學「台大工程」學刊　第九十期　民國九十三年二月　第 75–82 頁

REAL-TIME PROCESS SCHEDULING AND OPERATING SYSTEMS
即時行程排程與作業系統

Jun Wu[*]　*Li-Pin Chang*[**]　*Chih-wen Hsueh*[†]　*Tei-Wei Kuo*[‡]
吳 卓 俊[*]　張 立 平[**]　薛 智 文[†]　郭 大 維[‡]

[*]Ph.D. student　[**]Ph.D.　[†]Assistant Professor　[‡] Professor
[*] [†]Dept. of Computer Science and Information Engineering, National Chung Cheng University, Chiayi, Taiwan 62102, R.O.C.
[**] [‡] Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan 10617, R.O.C.

[*]博士班研究生　[**]博士　[†]助理教授　[‡] 教授
[*] [†]國立中正大學資訊工程學系
[**] [‡] 國立台灣大學資訊工程學系

摘　要

即時系統能適時地分派系統資源給應用程式以滿足它們的效能要求，在過去數十年來已被學者們廣泛地研究。在即時系統的基礎研究中，許多有效率的排程演算法與設計良好的許可控制策略，已成為新一代的作業系統在設計資源管理機制的重要基礎。其結果使得現今作業系統的資源管理已經非常靈活且功能強大。本論文將針對即時排程演算法與作業系統，彙整包含理論與實務的各項相關研究成果。

關鍵詞：即時作業系統、即時行程排程、資源管理、可排程性分析。

Abstract

Real-time systems, which could timely deliver applications various system resources to meet their performance specifications, had been studied for many decades. Based on efficient scheduling algorithms and reasonable admission control policies proposed in fundamental researches, the capabilities in resource management for new-generation operating systems implemented are now very flexible and powerful. In this paper, we survey classic and recent results in real-time scheduling algorithms and operating systems to cover both theoretical and practical issues.

Keywords: real-time operating system, real-time process scheduling, resource management, schedulability analysis.

1. INTRODUCTION

For different purposes, real-time processes are modelled by different parameters and the systems are evaluated according to specific criteria. For example, while real-time processes are modelled with timing parameters such as periods, deadlines, and minimum separations, processes whose deadlines can be violated but have hazardous results are called *hard* real-time processes; those which still contribute to the system after their deadlines expire are called *soft* real-time processes. On the other hand, systems aim at certain degree of performance guarantee could usually specify their own quality-of-service (QoS) parameters. For example, an average frame rate could be a QoS specification for a multimedia presenting system.

Researchers had made great success in real-time research community. Since early 1970s, classic scheduling algorithms driven by priorities were extensively studied, and then new approaches such as share-driven algorithms were proposed to complement the shortages of priority-driven scheduling algorithms. Based on the scheduling algorithms over independent processes, resource synchronization protocols were proposed with the objective to properly bound the duration and number of blocking due to resource contentions. The process model was also extended from the basic periodic processes to aperiodic/sporadic processes to handle special time-critical events such as interrupts. With these theoretical results researchers and engineers also started realizing new powerful operating systems to offer system designers more possibilities in the designing phase. Traditional operating systems were also given new capabilities to extend their applications to the field of embedded real-time computing.

In this paper, we survey classic and recent results in real-time scheduling algorithms and operating systems to cover both theoretical and practical issues. The rest of this paper is organized as follows: Section 2 summarizes related work in real-time process scheduling and resource management. Section 3 provides a snapshot of recent developed real-time operating systems. Section 4 is the summary.

2. REAL-TIME PROCESS SCHEDULING

The most common and applicable real-time process model is the *periodic real-time process model*. A periodic real-time process recurrently invokes the same program to perform certain computations. Every invocation of program of a process is referred to as a job of the process. The time among any two subsequent jobs is regular, which is referred to as the *period* of the process. Every job of the process must complete before a specified deadline. In the following section, we consider various scheduling algorithms proposed in different contexts by considering independent processes only.

2.1 Priority-Driven Real-Time Scheduling

Real-time processes could be scheduled according to their criticality so that important processes are always scheduled first other than the less-important ones. Alternatively, a real-time scheduling algorithm could also aim to maximize the system utilization. Priorities could be fixed or dynamic under different scheduling algorithms, depends on their objectives.

Liu and Layland [1] proposed the *rate monotonic scheduling* (RMS) algorithm that assigns priorities to processes that are inversely proportional to their periods. For fixed-priority assignments, they showed that RMS algorithm is optimal in the sense that if a process set is schedulable with any fixed-priority scheduling algorithm, the process set is schedulable with RMS.

Liu and Layland [1] also introduced the concept of *achievable utilization factor* so as to provide a quick test for deciding whether a set of periodic processes can be scheduled by a preemptive fixed-priority scheduler. They derived an expression for the achievable utilization factor as a function of the number of processes in the process set. Kuo and Mok [2] later extended their result by relating the achievable utilization factor to the number of *fundamental frequencies* in a set of periodic processes. Note that the number of fundamental frequencies is usually much less than the number of processes in a system. Based on the transformation of process periods into a special pattern, Han and Tyan [3,4] proposed new polynomial-time schedulability tests. Kuo, *et al.* [5] later exploit the process period pattern and show that their schedulability tests are much more precise than any existing work [1~4].

Excellent dynamic-priority real-time scheduling algorithms were also extensively studied. Liu and Layland [1] proposed an optimal scheduling algorithm, called *earliest deadline first* (EDF) algorithm that schedules the most urgent process first. Another well-known optimal scheduling algorithm is the *least slack first* (LSF) algorithm [6,7] that assigns priorities to processes based on their slacks, which is defined as the maximal amount of time that the process can wait while still meeting its deadline. Different from EDF, the priority of a single job could also dynamically change since the slack of every job decreases as time goes by. Note that most dynamic-priority real-time scheduling algorithms such as EDF and LSF could achieve full processor utilization.

As real-time scheduling algorithms for uniprocessor environments were extensively studied, it is less understood to schedule real-time processes under multiprocessor environments. Most multiprocessor real-time processes scheduling problems are NP-hard. Process allocation techniques are used so as to minimize the number of processors required and to provide a better achievable utilization factor. The process allocation problem is often cast as a bin packing problem or other similar problems. Dhall and Liu [8] classified multiprocessor scheduling algorithms as following two distinct approaches: (1) *partitioned scheduling approaches*: Processes are statically assigned to processors and not allowed to migrate; (2) *global scheduling approaches*: Processes are dynamically assigned to processors and migration is allowed.

Partitioned scheduling approaches has the advantage of reducing the multiprocessor scheduling problem to scheduling problem on individual processors. Partitioned scheduling approaches may seem less powerful than global scheduling approaches due to statically processor assignment, but Dhall and Liu [8] showed that global scheduling using either EDF or RM can result in arbitrarily low processor utilization in multiprocessor systems. Recently, Andersson, *et al.* [9] proposed a global fixed-priority scheduling algorithm for multi-processor real-time processes. They showed that their proposed algorithm successfully schedules any periodic processes with utilization no more than $m^2/(3m-2)$ on m identical processors.

2.2 Share-Driven Real-Time Scheduling

Lin, *et al.* [10~12] proposed time-driven scheduling schemes which are based on the *pinwheel* task model. Their proposed pinwheel scheduling mechanisms not only have good schedulability but also have good jitter control. Based on the notion of *general processor sharing* (GPS), various rate-based scheduling algorithms for periodic and sporadic processes were proposed. The GPS-based scheduling is a work-conserving scheduling mechanism [13~15]. The schedulability of each process in GPS-based systems is guaranteed with an assigned CPU service rate, independent of the demands of other processes. The enforcement of a guaranteed CPU service rate for a process must rely on certain admission control mechanism to manage the total workload of the system [16].

2.3 Real-Time Resources Synchronization

Resources such as memory semaphores usually require exclusive possessions so that the semaphores or the memory protected by the semaphore can be accessed without race conditions. However, to exclusively obtain certain resources real-time processes might be prevented from doing so if another process currently holds the desired resources. Eligible real-time processes is said to encounter a *priority inversion* if it tries to lock a resource which is currently held by one other lower priority process. How to properly control the number and duration of priority inversions is an important issue since uncontrolled priority inversions could severely damage the stability of real-time systems.

Mok [7] proposed a monitor scheduling model in which critical sections of a process is nonpreemptible. Sha, Rajkumar, and Lehoczky [17] proposed the *priority inheritance protocol* (PIP) in which processes can inherit the higher priority of a process they block and the *priority ceiling protocol* (PCP) to handle process synchronization with different sizes of critical sections. The *priority ceiling* of a resource is the priority of the highest priority process which may use the resource. A process's resource request is blocked if its priority is no higher than the priority ceiling of any resource which has been grabbed by another process but has not yet been released.

By consideration of dynamic priority assignments, Chen and Lin [18] proposed a variation of PCP, called *dynamic priority ceiling protocol* (DPCP), while PCP was originally designed for fixed priority assignments. Baker [19] proposed the *stack resource policy* (SRP) that extends PCP for multiple-resource-unit synchronization and may adopt either dynamic or fixed priority assignments. Baker also showed that the maximum number of context switching per process under SRP is no more than two.

2.4 Scheduling Aperiodic Real-Time Processes

Processes in real-time systems may be classified into *periodic process*, *aperiodic process* and *sporadic process*. Aperiodic process invokes its process instance exactly once while periodic process and sporadic process are not. Sporadic process invokes its instances at any time instants with a defined minimum *inter-arrival time* between two consecutive invocations.

Most work on scheduling aperiodic processes can be divided into two types: *slack stealing* [20,21] and *bandwidth preserving*. The concept of slack stealing is executing aperiodic processes by using the available *slack* times of periodic and sporadic processes. The concept of *bandwidth preserving* is creating a periodic (or sporadic) process, called *bandwidth preserving server*, with execution budgets. During the period (or minimum inter-arrival time) of bandwidth preserving server, the execution budgets is a time amount for executing aperiodic processes.

For fixed-priority systems, Lehoczky [22] proposed a bandwidth preserving mechanism, called *deferrable server*, which the execution budget may be suspended and deferred. Note that the execution budget of a deferrable server is consumed when the server executes and replenished to original amount when the server invokes. The major drawback of the deferrable server is it may delay lower priority processes for more time than a periodic process with the same period and execution time of a deferrable server. Spuri and Buttazzo [23] proposed the *sporadic server* to solve this problem. The sporadic server emulates a periodic process when its budget is broken up into chunks to be replenished at different times. Such an approach also improves the schedulability of deferred server.

For deadline-driven systems, Deng, *et al.* [24] proposed a *constant utilization server* (CUS) to emulate a sporadic task that has a constant instantaneous utilization. Spuri and Buttazzo [23] proposed an enhanced CUS, called *total bandwidth server* (TBS), that is allowed to claim the background time not used by periodic processes. Note that TBS may has the starvation problem, Demers, *et al.* [25] proposed a fairness TBS that eliminates the starvation problem, called *weighted fair-queuing* (WFQ) algorithm.

3. REAL-TIME OPERATING SYSTEMS

The primary goal of a real-time operating system is to provide a set of resource reservation methods so that the quality-of-service for applications could be guaranteed. The contributions made by researchers and engineers to real-time operating systems could be categorized into two major forms: The first is to build a new operating system. The other is to add a real-time application interface to a full-functional traditional operating system. It could be accomplished by modifying the kernel source, adding kernel-mode modules, or manipulating the existing system primitives. Recently developed real-time operating systems will be enumerated in the rest of this section.

3.1 System Scheduler Extensions

In 1996, a series of research and system experiments were executed over Windows, Linux, and other third-party operating systems. Kuo, *et al.* started exploring QoS over ordinary operating systems. Their work on Microsoft Windows was to provide Windows NT soft QoS guarantee through the design of a Microsoft Windows NT middleware, called the *computing power regulator* (CPR) [26]. Distinct from the past work, their work made no modifications to Windows and any application programs. They proposed a larger granularity of resource reservation at the application level and showed its feasibility in the implementation and performance evaluation of CPR over realistic workloads.

Adelberg, *et al.* [27] presented a real-time emulation program to build soft real-time scheduling on the top of UNIX. Processes' priorities were adjusted according to their timing constraints. The well-known EDF algorithm and the LSF algorithm were studies for emulation. Childs and Ingram [28] chose to modify the Linux source code by adding a new scheduling class called SCHED_QOS to let applications to specify the amount of CPU time per period. QoS scheduling is supported for CPU and IDE disks. The idea of disk QoS guarantee is as the same as that for CPU. Each disk request is tagged with process ID. The driver records the duration of requests and charges that to processes. The study tried to provide multi-resource QoS guarantee.

3.2 Real-Time Application Interfaces

Well-designed application interfaces could significantly reduce the cost to develop and maintain real-time application systems. Most of the real-time application interfaces are crafted over Linux operating system due to the source code availability. *Real-time application interface* (RTAI) for Linux was proposed by Mantegazza at DIAPM [29] and RT-Linux was proposed by Barabanov and Yodaiken at New Mexico Institute of Technology [30]. Both RTAI and RT-Linux are two successful real-time Linux solutions, where they share very similar system architecture. All hard real-time tasks are scheduled by a tiny kernel under Linux, where Linux is also scheduled by the tiny kernel as a "process". Because all interrupts are intercepted by the tiny kernel, and every scheduling activity is under the control of the tiny kernel (including Linux), hard real-time scheduling is, thus, supported. However, a significant portion of system services must be done at the Linux level unless more implementation is done at the tiny kernel. Such a reality imposes great difficulty on RT-Linux and RTAI in truly QoS support for various real-time or time-sensitive applications. RT-Linux is commercialized under a company called FSMLabs. RTAI is now supported by Lineo.

The research group at the University of California at Irvine [31,32] directly modify the scheduling mechanism of Linux. The goal is to provide a "general-purpose" real-time scheduling framework which can handle time-driven, priority-driven, and even new scheduling schedulers. Users could specify a budget for each real-time process under a priority-driven scheduler or do all kinds of combinations. The research group also proposes a series of monitor tools and tool kits for system/application developments. The work is commercialized under a company called RedSonic. The real-time Linux is called RED-ICE. The major problem in QoS support under RED-ICE also happens in many real-time Linux solutions such as those described below. That is the complicated inter-activities among processes, and the difficulty in managing priority inversion.

3.3 Resource-Specific QoS Support

While it seems if the CPU is properly scheduled among applications then the applications could also obtain desired shares of resources other than CPU (such as I/O devices), experimental results usually leads us to unexpected resource utilizations and scheduling behaviors. The rationale behind is that modern subsystems such as peripheral interfaces already adopted intelligent and powerful co-processors to handle requests. In other words, to correctly guarantee that applications could accomplish their tasks the reservations and releases of the CPU and the other resource should be properly coordinated.

Recently, researchers focus on QoS support for storage systems and bus bandwidth management. Chang, *et al.* [33] propose an energy-efficient real-time storage system over embedded Linux platforms. An USB bandwidth reservation technique is also proposed by Chen *et al.* [34] for a Linux-based surveillance server.

A more precise reservation technique for USB devices is proposed by Huang, *et al.* [35]. In addition to the QoS reservation for CPU time, researchers started exploring the resource allocation problems and their QoS issues over various buses. In particular, the control area network (CAN), which is a serial communication protocol for distributed real-time system, was studied by Hon and Kim [36] in proposing a bandwidth reservation algorithm. In [37], Natale discussed the applicability of the EDF algorithm for the scheduling of CAN messages. They showed that a higher bus utilization was delivered. Kettler, *et al.* [38] developed formal scheduling models for several types of commonly used buses, *e.g.*, PCI and MCA, in PC's and workstations.

4. SUMMARY

In this paper, we give a brief review of related work in real-time process scheduling, resource management, schedulability analysis and recent developed real-time operating systems. We cover the scheduling of periodic, sporadic, and aperiodic processes, and the processes could be scheduled by priority-driven or share-driven techniques. Resource synchronization protocols to bound the time and duration of priority inversions are presented. Based on the theoretical results, new operating systems realized or traditional operating systems empowered with new technologies are now very powerful and flexible. Due to the very distinctive nature of real-time and embedded systems, each system has very specific requirements to meet. The primary objective of this paper is to deliver readers a picture on which tools they could use and which approaches should be avoided. Researchers could also find which topics are well-studied and where is the missing part.

REFERENCES

[1] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard real-time environment," *Journal of the Association for Computing Machinery (JACM)*, Vol. 20, No. 1, Jan. 1973, pp. 46–61.

[2] T.-W. Kuo and A. K. Mok, "Load adjustment in adaptive real-time systems," *Proceedings of the IEEE 12th Real-Time Systems Symposium (RTSS)*, San Antonio, U.S.A., Dec. 1991, pp. 160–170.

[3] C.-C. Han and H.-Y. Tyan, "A better polynomial-time schedulability test for real-time fixed-priority scheduling algorithms," *Proceedings of the IEEE 18th Real-Time Systems Symposium (RTSS)*, San Francisco, CA, Dec. 1997, pp. 36–45.

[4] C.-C. Han, "A better polynomial-time schedulability test for real-time multiframe tasks," *Proceedings of the IEEE 19th Real-Time Systems Symposium (RTSS)*, Madrid, Spain, Dec. 1998, pp. 104–113.

[5] T.-W. Kuo, Y.-H. Liu, and K.J. Lin, "Efficient on-line schedulability tests for priority driven real-time systems," *Proceedings of the IEEE 6th Real-Time Technology and Applications Symposium (RTAS)*, Washington, D.C., U.S.A., June 2000, pp. 4–13.

[6] J. Y. T. Leung and J. Whitehead, "On the complexity of fixed-priority scheduling of periodic real-time tasks," *Performance Evaluation*, Vol. 2, No. 4, Dec. 1982, pp. 237–250.

[7] A. K. Mok, "Fundamental design problems for the hard real-time environment," MIT Ph.D. Dissertation, Cambridge, MA, 1983.

[8] S. K. Dhall and C. L. Liu, "On a real-time scheduling problem," *Operations Research*, Vol. 26, No. 1, 1978, pp. 127–140.

[9] B. Andersson, S. Baruah and J. Jansson, "Static-priority scheduling on multiprocessors," *Proceedings of the 22th IEEE Real-Time Systems Symposium (RTSS)*, Dec. 2001, pp. 193–202.

[10] C.-C. Han, K.-J. Lin and C.-J. Hou, "Distance-constrained scheduling and its applications to real-time systems," *IEEE Transactions on Computer*, Vol. 45, No. 7, July 1996, pp. 814–826.

[11] C.-W. Hsueh and K.-J. Lin, "Optimal pinwheel schedulers using the single number reduction technique," *Proceedings of the IEEE 17th Real-Time Systems Symposium (RTSS)*, Washington, D.C., U.S.A., Dec. 1996, pp. 196–205.

[12] C.-W. Hsueh and K.-J. Lin, "On-line schedulers for pinwheel tasks using the time-driven approach," *Proceedings of the 10th Euromicro Conference on Real-Time Systems*, Berlin, Germany, June 1998, pp. 180–187.

[13] R. Gerber, S. Hong and M. Saksena, "Guaranteeing end-to-end timing constraints by calibrating intermediat processes," *Proceedings of the IEEE 15th Real-Time Systems Symposium (RTSS)*, San Juan, Puerto Rico, Dec. 1994, pp. 192–203.

[14] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the single node case," *IEEE/ACM Transactions on Networking*, Vol. 4, No. 1, Feb. 1993, pp. 344–357.

[15] A. K. Parekh and R. G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the multiple node case," *IEEE/ACM Transactions on Networking*, Vol. 2, No. 2, Apr. 1994, pp. 137–150.

[16] M. Spuri, G. Buttazzo and Sensini, "Scheduling aperiodic tasks in dynamic scheduling environment," *Journal of Real-Time Systems*, Vol. 10, No. 2, 1996, pp. 179–210.

[17] L. Sha, R. Rajkumar and J. P. Lehoczky, "Priority inheritance protocols: an approach to real-time synchronization," *IEEE Transactions on Computers*, Vol. 39, No. 9, Sep. 1990, pp. 1175–1185.

[18] Chen and Lin, "Dynamic priority ceilings: a concurrency control protocol for real-time systems," Technical Report, Dept. of Computer Science, University of Illinois at Urbana-Champaign, UIUCDCS-R-89-1511, 1989.

[19] T. P. Baker, "A stack-based resource allocation policy for real time processes," *Proceedings of the IEEE 11th Real-Time Systems Symposium (RTSS)*, Lake Buena Vista, Florida, Dec. 4-7, 1990, pp. 191–200.

[20] H. Chetto and M. Chetto, "Some results of the earliest deadline scheduling algorithm," *IEEE Transactions on Software Engineering*, Vol. 15, No. 10, Oct. 1989, pp. 1261–1269.

[21] J. P. Lehoczky and S. Ramos-Thuel, "An optimal algorithm for scheduling soft-aperiodic tasks in fixed priority preemptive systems," *Proceedings of the IEEE Real-Time Systems Symposium (RTSS)*, Dec. 1992, pp. 110–123.

[22] J. P. Lehoczky, L. Sha and J. K. Strosnider, "Enhanced aperiodic responsiveness in hard real-time environments," *Proceedings of the IEEE Real-Time Systems Symposium (RTSS)*, 1987, pp. 261–270.

[23] M. Spuri and G. Buttazzo, "Scheduling aperiodic tasks in dynamic priority systems," *Real-Time Systems Journal*, Vol. 10, 1996, pp. 179–210.

[24] Z. Deng and J. W.-S. Liu, "Scheduling real-time applications in an open environment," *Proceedings of the IEEE Real-Time Systems Symposium (RTSS)*, San Francisco, CA, U.S.A., Dec. 1997, pp. 308–319.

[25] A. Demers, S. Keshav and S. Shenker, "Analysis and simulation of a fair queueing algorithm," *Proceedings of ACM SIGCOMM*, 1989, pp. 1–12, and *Journal of Internetworking Research and Experience*, Oct. 1990.

[26] G.-H. Huang, S.-K. Ni and T.-W. Kuo, "The design and implementation of the CPU power regulator for multimedia operating systems," *Proceedings of the Work-In-Progress Session of IEEE 17th Real-Time Systems Symposium (RTSS)*, Washington, D.C., U.S.A., Dec. 1996, pp. 27–30.

[27] B. Adelberg, H. Garcia-Molina and B. Kao, "Emulating soft real-time scheduling using traditional operating systems schedulers," *Proceedings of the IEEE 15th Real-Time Systems Symposium (RTSS)*, San Juan, Puerto Rico, Dec. 1994, pp. 292–298.

[28] S. Childs and D. Ingram, "The Linux-SRT integrated multimedia operating systems: Bring QoS to the desktop," *Proceedings of the IEEE 7th Real-Time Technology and Applications Symposium (RTAS)*, Taipei, Taiwan, R.O.C., June 2001, pp. 135–140.

[29] E. Bianchi, L. Dozio, G. L. Ghiringhelli and P. Mantegazza, "Complex control systems, applications of DIAPM-RTAI at DIAPM," *Realtime Linux Workshop*, Vienna, Austria, 1999.

[30] M. Barabanov and V. Yodaiken, "Introducing real-time Unix," *Linux Journal*, No. 34, Feb. 1997, pp. 19–23.

[31] Y. C. Wang and K. J. Lin, "Enhancing the real-time capability of the linux kernel," *Proceedings of the 5th Real-Time Computing Systems and Applications Symposium (RTCSA)*, Hiroshima, Japan, Oct. 1998, pp. 11–20.

[32] Y.-C. Wang and K. J. Lin, "Implementing a general purpose real-time scheduling framework in the red-linux real-time kernel," *Proceedings of the IEEE Real-Time Systems Symposium (RTSS)*, Phoenix, Arizona, U.S.A., Dec. 1999, pp. 246–255.

[33] L.-P. Chang, T.-W. Kuo and S.-W. Lo, "A dynamic-voltage-adjustment mechanism in reducing the power consumption of flash memory for portable devices," *Proceedings of the IEEE International Conference on Consumer Electronics*, Los Angeles, U.S.A., June 2001.

[34] H.-M. Chen, S.-Y. Zhuo, C.-Y. Huang and T.-W. Kuo, "An USB-based surveillance system over wireless network," *Proceedings of the 7th International Conference on Distributed Multimedia Systems*, Taipei, Taiwan, R.O.C., Sep. 2001.

[35] C.-Y. Huang, L.-P. Chang and T.-W. Kuo, "A cyclic-executive-based QoS guarantee over USB," *Proceedings of the IEEE 9th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, Toronto, Canada, June 2003.

[36] S. H. Hong and W.-H. Kim, "Bandwidth allocaton in CAN protocol," *Proceedings of the IEEE Control Theory and Applications*, Jan. 2000, pp. 37–44.

[37] D. M. Natale, "Scheduling the CAN bus with earliest deadline techniques," *The 21st IEEE Real-Time Systems Symposium*, Nov. 2000.

[38] K. A. Kettler, J. P. Lehoczky and J. K. Strosnider, "Modeling bus scheduling policies for realtime systems," *IEEE Real-Time Systems Symposium*, Dec. 1995.

Jun Wu (吳 卓 俊) received the A.A.S. degree in electronic data processing from the Chung Kuo Institute of Technology in Taipei, Taiwan, R.O.C., in 1993. He received the BSE degree in information engineering from the I-Shou University in Kaohsiung, Taiwan, R.O.C., in 1996. He received the MBA degree in information management from the National Yunlin University of Science and Technology in Yunlin, Taiwan, R.O.C., in 1998. He is currently a PhD student in the Department of Computer Science and Information Engineering of the National Chung Cheng University in Chiayi, Taiwan, R.O.C. His research interests include resource management for real-time systems and query processing for multiprocessor database systems.

Li-Pin Chang (張 立 平) received the BS degree in information engineering from I-Shou University, Kaohsiung, Taiwan, R.O.C., in 1995. He received the MS and PhD degrees in computer science and information engineering from National Taiwan University, Taipei, Taiwan, R.O.C., in 1997 and 2003, respectively. He is currently working on research topics over real-time systems and storage systems for embedded systems.

Chih-wen Hsueh (薛 智 文) received the BS degree in computer science and information engineering from the National Taiwan University. He received the MS degree in computer science from the University of Southern California and the PhD degree in information and computer science from the University of California at Irvine. He is an Assistant Professor in the Department of Computer Science and Information Engineering at the National Chung Cheng University (NCCU), Taiwan. His research interests include real-time systems, real-time operating systems, and real-time embedded systems. After two years of military service, he worked for one year as an research assistant in the Institute of Information Science at the Academia Sinica. Before he joined NCCU, he worked for several startups in the San Francisco area.

Tei-Wei Kuo (郭 大 維) received BSE degree in computer science and information engineering from National Taiwan University in Taipei, Taiwan, in 1986. He received the MS and PhD degrees in computer sciences from the University of Texas at Austin in 1990 and 1994, respectively. He is currently a Professor in the Department of Computer Science and Information Engineering of the National Taiwan University, Taiwan, R.O.C. From August 1994 to July 2000, he was an Associate Professor in the Department of Computer Science and Information Engineering of the National Chung Cheng University, Taiwan, R.O.C. His research interests include real-time process scheduling, real-time operating systems, real-time databases, and embedded systems. He has published more than 60 papers in academic journals and conference proceedings.

Dr. Kuo is an associate editor of the Journal of Real-Time Systems. He was a Program Co-Chair of IEEE 7th Real-Time Technology and Applications Symposium, Taipei, Taiwan, 2001, a Program Co-Chair of the 7th International Conference on Real-Time Computing Systems and Applications, Cheju Island, Korea, 2000, and the Program Chair for Asia and Far East of the 9th International Workshop on Parallel and Distributed Real-Time Systems, San Francisco, U.S.A., in 2001. He has received several research awards in Taiwan, including the Distinguished Research Award from the Taiwan National Science Council in 2003 and the Young Scholar Research Award from the Academia Sinica (Taiwan) in 2001. He has consulted for government and industry on problems in various real-time systems design.

收稿日期 92 年 12 月 31 日、修訂日期 93 年 2 月 11 日、接受日期 93 年 2 月 20 日
Manuscript received December 31, 2003, revised February 11, 2004, accepted February 20, 2004