Real-Time Digital Signal Processing of Phased Array Radars

Chin-Fu Kuo, Tei-Wei Kuo, *Senior Member*, *IEEE*, and Cheng Chang

Abstract—With the advance of hardware and software technology, modern phased array radars are now built with commercial-off-the-shelf (COTS) components, and it opens up a new era in real-time resource scheduling of digital signal processing. This paper targets the essential issues in building a component-oriented Signal Processor (SP), which is one of the two major modules in modern phased array radars. We propose a simple but effective task allocation policy and a real-time scheduling algorithm to address the design objectives of SP's. We are able to bound the number of processing units needed for a component-oriented SP in the design time, while everything was done empirically in the past. A series of experiments was done to demonstrate the strength of our methodology.

Index Terms—Phased array radar, real-time scheduling, digital signal processing, component-oriented signal processor, system capacity estimation.

1 Introduction

A phased array radar must search and track suspicious targets in its surveillance space in a real-time fashion. There are two major modules in a phased array radar: Radar Control Computer (RCC) and Signal Processor (SP). RCC is responsible for scheduling radar beam transmissions for searching and tracking based on targets' status and search types. On the other hand, SP must process returned signals in a real-time fashion. Although the design of a phased array radar must meet various real-time requirements, it is often designed with non-real-time resource scheduling mechanisms, such as FIFO scheduling [2]. The common reason in doing so is mainly because of hardware constraints and insufficient knowledge of real-time technology. As a result, many resources are wasted with a very limited guarantee on system performance.

Real-time scheduling problems have been analyzed for different architectural assumptions. The priority-driven scheduling discipline, e.g., [21], [22], [28], often schedules real-time processes based on their priorities in an online fashion, whereas the time-driven scheduling discipline [10], [11] usually predetermines the execution times of processes before running time. Liu and Layland [21] first introduced the concept of achievable utilization factor to provide a quick test for deciding the schedulability of a set of independent periodic processes and proposed the well-known *rate monotonic* (RM) and *earliest deadline first* (EDF) scheduling algorithms. Dhall and Liu [5], [6] later showed that RM and EDF are not optimal for multiprocessor environments as the achievable utilization factor may be arbitrarily close to 1 when the number of processors is less than the number of processes. Mok [22] proved that real-time scheduling with mutual exclusion is, in general, NP-hard. Sha et al. [28] then proposed the famous *priority ceiling protocol* (PCP) in which processes can inherit the higher priority of a process they block. Lin et al. [20] proposed the imprecise computation model. Han and Lin [9] considered distance constraints in process executions. Kuo and Mok [18] explored scheduling for adaptive system workloads. Kang et al. [15] and Natale and Stankovic [24] explored the scheduling problems with end-to-end deadlines. Bernat and Burns [3] and Koren and Shasha [17] considered the skippings of process executions in consecutive cycles. Mok and Chen [23] considered the scheduling of multi-frame tasks whose execution times vary regularly in consecutive periods. Recently, Jeffay and Goddard [14], Kuo et al. [19], Spuri et al. [29], Stoica et al. [30], and Waldspuger and Weihl [32] proposed rate-based scheduling, in which the schedulability of each process is enforced by a guaranteed CPU service rate, independent of the demands of other processes.

The development of component-oriented SPs is strongly influenced by the Rapid prototyping of Application Specific Signal Processors (RASSP) program lead by the Department of Defense, United States of America [27]. The RASSP program formalized an engineering process for developing SPs to reduce the total product development time and cost by a factor of four. While a number of researchers have proposed highly efficient real-time scheduling algorithms, e.g., [14], [15], [21], [22], [28], little work is done for radar scheduling. The task models and the work presented in [1], [7], [12], [14], [15] are among the few closely related to dwell scheduling at the RCC level. Note that, although researchers have explored the scheduling of SDF graphs for the SAR benchmark and other related signal processing applications in the embedded systems and signal processing literature, the graph processing models are not directly related to real-time scheduling in SPs, where SDF is the dataflow language used in the RASSP program and more related to RCC-level workloads. The work presented in this paper is one of the

- *C.-F. Kuo and T.-W. Kuo are with the Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan 106, ROC. E-mail: {d89005, ktw}@csie.ntu.edu.tw.*
- *C. Chang is with the System Development Center, Chung Shan Institute of Science and Technology, TaoYuan, Taiwan 325, ROC. E-mail: cchang60@ms37.hinet.net.*

Manuscript received 30 July 2002; revised 22 Dec. 2002; accepted 3 Jan. 2003. For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number 117034.

first attempts in building a truly real-time SP. The traditional scheduling approaches for signal processing in SPs are merely FIFO-like mechanisms which may let the signal processing seriously suffer from the priority inversion problem and underutilize the processing power of processing units. With virtually everything done and verified empirically in the traditional approaches, the system capacity of a traditional SP is usually hard to estimate or estimated very conservatively.

This paper addresses two major issues in the design of SPs: 1) the schedulability of critical tasks under the minimum operation requirements, such as the processing of search signals and 2) the system capacity estimation. In this paper, we propose a simple but effective task allocation policy, which separates periodic and aperiodic workloads and a real-time scheduling algorithm to process digital signals in a real-time fashion. We derive bounds for the number of processing units needed for an SP under performance specifications. Most importantly, we formalize the workload of a typical component-oriented SP for modern phased array radars. We must emphasize that this work represents one of the pioneer works in building next-generation SPs. In the past, virtually everything was done empirically. Inefficient resource allocation mechanisms like FIFO were adopted, and system capacity was very difficult to estimate. With the complexity of the system we are building, we believe that only rigorous theory which lends itself to advanced implementation methods can provide us with the reliability and performance acceptable to users.

The rest of this paper is organized as follows: Section 2 defines the system architecture of a typical SP and illustrates the idea of component-oriented SPs and radars. Section 3 formally defines the workload of a typical SP. Section 4 proposes our task allocation policy and real-time scheduling algorithm. We also propose bounds for the number of processing units needed for an SP under performance specifications. Section 5 provides experimental results to show the strength of our approach. Section 6 is the conclusion.

2 System Architecture

A typical radar transmits a beam of microwave toward a target and waits for the reflection. A target is said to be *detected* if the radar receives the microwave reflected by the target. The direction of the target is the direction of the radar beam, and the distance (called *range*) of that target can be calculated by how long it takes to receive the microwave reflection. One major difference between a traditional radar and a phased array radar is that a traditional radar has only one antenna, but a phased array radar consists of an array of antennae [2]. In a traditional radar, the radar beam is steered by the mechanical radar pedestal, while, in a phased array radar, the radar beam is steered by an electronic Beam Steering Controller (BSC). BSC steers the radar beam by adjusting the phase difference of each antenna during microwave transmissions such that the microwave energy is concentrated toward the desired direction. As a result, the Radar Control Computer (RCC) of the phased array radar can steer the radar beam from one direction to

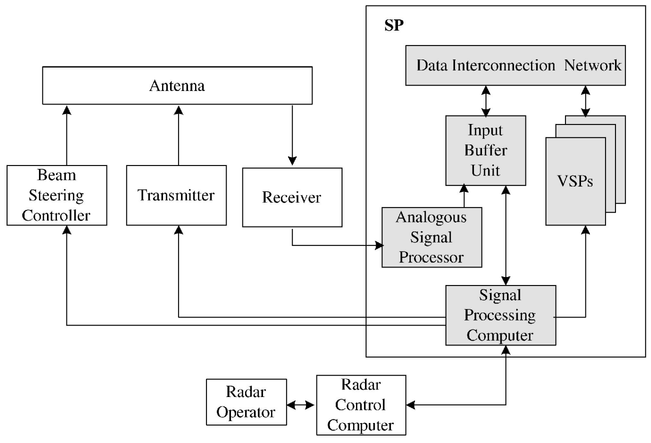


Fig. 1. The hardware architecture of a phased array radar.

another direction in the electronic speed, instead of some mechanical speed for traditional radars [2]. With an advanced signal processor (SP), a phased array radar can be a multifunction radar that supports search, track, missile guidance, etc., simultaneously [2].

A phased array radar consists of several important modules: Radar Control Computer (RCC), Signal Processor (SP), Beam Steering Controller (BSC), Receiver, Antenna, and Transmitter, where an SP consists of an Analog Signal Processor (ASP), a Signal Processing Computer (SPC), and various processing units, as shown in Fig. 1 [2], [12], [25], [31]. RCC schedules dwell transmissions in a real-time fashion by sending SPC commands. When SPC receives commands from RCC for radar beam transmissions, it issues commands to BSC and Transmitter for radar beam transmissions in specified directions. When Antenna and Receiver receive returned signals and pass them to ASP for analog-to-digital signal processing, the digital signals are saved at the input buffer unit (IBU) for later processing. Processing units of an SP includes several types of processors, such as those for data processing, pulse compression, digital signal processing, etc., which are Vector Signal Processors (VSP) in the following paragraphs. Data Interconnection Network (DIN) is for data transmission between processing units and IBU. All processing units, IBU, and DIN are under the command of SPC, where SPC assigns each processing unit a signal processing job to meet the hard deadline of each individual job. Commands in an SP go through another channel, such as VMEbus or Fibre Channel.

With the advance of software and hardware technology, a modern radar SP is no longer a complicated hardware system with everything wired. Instead, an SP can be made of many commercial-off-the-shelf (COTS) components, and the functions of many hardware components are now reimplemented by software algorithms [4]. An SP can be considered as a collection of internal nodes, as shown in Fig. 2, where an internal node (IN) can be an ASP, any COTS board (i.e., VSP) with multiple digital signal processors (DSPs), such as ADSP 21060 DSPs, or memory modules for IBU.

The interconnection fabrics are communication channels for data and command delivery. Network bridges can connect different groups of SP and related subsystems

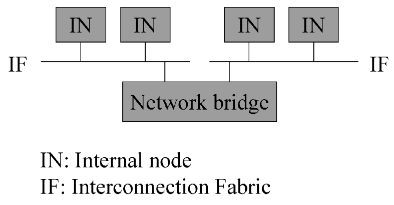


Fig. 2. A component-oriented signal processor.

which work together. Various radar digital signal processing functions, such as pulse doppler processing (PDP), moving target indication (MTI), tracking processing, and monopulse estimation, are implemented by software modules executed on VSPs, where each VSP may also be equipped with some special purpose processing units for time-consuming functions. Most importantly, different combinations of COTS replaceable VSPs may result in different SPs which may fit different radar system requirements. Also, as more powerful COTS DSP boards appear, a new SP can be built rapidly by using them.

3 Workload Characteristics

In phased array radars, RCC schedule radar dwells in units called *scheduling interval* (SI) [2], [7], [13], where the length of the scheduling interval is determined by various factors from the system specs and usually in tens-of-milliseconds. In the beginning of each scheduling interval, SPC receives commands from RCC, where the commands are dwell transmissions and their returned signal processing. SPC, which is in charge of the real-time resource scheduling of the entire SP, must issue commands to BSC and Transmitter for radar beam transmissions in specified directions and prepare to schedule the processing of returned signals in a real-time fashion. In this paper, we will focus on the real-time processing of returned signals, which is the main duty of an SP. For the simplicity of presentation, we will use terms “task,” “job,” and “dwell,” interchangeably.

The incoming workload of an SP in each ith SI can be formally defined as a collection of tasks $T_i = \{\tau_{i,1}, \tau_{i,2}, \cdots, \tau_{i,n_i}\}$, where each task $\tau_{i,j}$ is associated with three parameters $(r_{i,j}, c_{i,j}, d_{i,j})$. $r_{i,j}$, $c_{i,j}$, , and $d_{i,j}$ are the ready time, the processing time, and the deadline of the returned signal for the corresponding dwell, i.e., $\tau_{i,j}$, issued in the ith SI, respectively. Each task $\tau_{i,j}$ is assigned to a VSP to execute (or process) nonpreemptively. In other words, once a task is assigned to execute on a VSP, the VSP is no longer available until the task is done. The processing time $c_{i,j}$ of task $\tau_{i,j}$ includes the actual processing time of the corresponding signal and the data and command communication time.

We are interested in a homogeneous SP environment in which there are N identical VSPs and each internal node is a COTS DSP board. The homogeneity requirements of the environment in SP design is to simplify the design and the maintenance of SP, where functions of many hardware components in SP are now reimplemented by software algorithms over COTS VSP components. The adoption of COTS VSP components and software algorithms could reduce the expense in building SP and also decrease future maintenance cost.

The types of tasks in a phased-array-radar SP can be, in general, classified as search, track, and confirmation tasks [2], [1], [7], [12]: Search tasks are periodic tasks to search moving objects in different directions and elevations. Typical search tasks are Horizontal Search and Volumetric Search [1]. A typical search task must issue and process a fixed number of beams within every specified time interval. Each specified time interval is usually divided into two sections: peak and normal durations. In the peak duration, a larger number of radar beams are issued and processed. Track tasks, which are used to track interesting objects in the surveillance space, can be considered as aperiodic tasks because their periods may change frequently, depending on the target type, target position, target speed, system load, etc. [2], [1]. Search tasks may trigger the creation of track tasks on RCC through confirmation tasks. In other words, when suspicious targets are identified by search tasks on RCC, a confirmation task is created for each suspicious target. If a suspicious target is identified as a real target, the corresponding confirmation task is removed, and a corresponding track task is created [2], [1]. In other words, confirmation tasks are also aperiodic. When a search, confirmation, or track task arrives, it must be processed within a specified deadline. Nonzero ready times will be given to search, confirmation, and track tasks due to delay for radar beam transmission, signal collection, analog-to-digital signal processing, etc. [2]. Note that the precedence constraints of tasks is handled at the RCC level. At the SP level, only independent nonpreemptive tasks are observed.

With similar signal processing procedures, the processing time of each search task can be well-bounded as a constant C_s. Because of the same reason, track tasks (/confirmation tasks) also have a bounded processing time C_t ($/C_c$) [12]. The deadlines of search, confirmation, and track tasks are a multiple of an SI because RCC and SP communicate with each other at a fixed and specified point of each SI [2], [7]. In this paper, we consider a high performance SP which might have many track (/confirmation) tasks whose deadlines are as close as double of SI, and search tasks can be pushed to their extreme such that their deadlines might be close to a threshold value equal to SD times of SI. For example, if the deadline $d_{i,j}$ of task $\tau_{i,j}$ is equal to X times of SI, then $\tau_{i,j}$ must finish its computation before the ending of the $(i+X)$th SI. Furthermore, the search tasks in many phased array radars often have a common big cycle such that all of the search tasks in the system can be considered as a single search multiframe-like task [2], [12], where the execution time of a multiframe task varies regularly in consecutive periods [23]. Such an integrated single search task can be, in general, modeled (and well-bounded) by the following seven parameters:

$$\tau^{search} = \{R, C_s, (SD \cdot SI), Max, Min, K, P\},$$

where, in the first K periods, the search task will request Max identical nonpreemptive jobs, and each of them is size C_s. Starting from the $(K+1)$ period until the Pth period, the search task will request Min nonpreemptive jobs ($Min < Max$), and each of them is size C_s. The search task

TABLE 1
SP Workload Characteristics

Tasks	Timing Parameters	Periodic
Search	$\{R, C_s, (SD \cdot SI), Max, Min, K, P\}$	Yes
Track	$(r_{i,j}, C_t, d_{i,j})$	No
Confirmation	$(r_{i,j}, C_c, d_{i,j})$	No

repeats for each P periods, where each P period is called a *big cycle*. The relative deadline of each nonpreemptive job is $(SD \cdot SI)$ (from the beginning of its issuing SI). When there are x nonpreemptive search jobs $J_1, J_2, \cdots, J_x$ issued in a period, the ready time of the ith job is set as $(R \cdot i)$. It is because there is only one transmitter, receiver, and ASP such that signals are returned one after another. The first K periods of the search task are called the *peak duration*. The workload of an SP is summarized in Table 1.

4 Real-Time Scheduling of Digital Signal Processing—A Case Study

4.1 Overview

The FIFO scheduling policy adopted in many traditional SPs may not be suitable to an advanced component-oriented SP because many hardware functions are no longer hardwired. The reimplementation of hardware functions in terms of software modules makes a highly flexible system architecture possible. Constraints on (FIFO) data flow in many traditional SPs can be substantially relaxed. In this paper, we will present our approach in building a modern real-time SP with a performance guarantee.

A phased array radar must guarantee the minimum operation of the system, which is often realized by periodic tasks responsible for searching suspicious targets in the surveillance space [2], [12]. As a result, for each scheduling interval (SI), RCC usually sends SPC a queue of tasks where the jobs of the search task are left in front of other tasks. The traditional FIFO scheduling policy unavoidably suffers from serious priority inversion problems for the jobs of the search task, where priority inversion is a phenomenon in which a higher priority task is blocked by a lower priority task. It is because track tasks which are not processed in the previous SI might be scheduled before the new jobs of the search task according to the FIFO scheduling policy. If there is a burst of track tasks coming in the previous SIs, jobs of the search task may miss their hard deadlines unless a lot of processing units are deployed in the SP. Furthermore, the mixed scheduling of the search tasks, track tasks, and confirmation tasks also incurs great difficulty in analyzing the timing behavior of the system and in deriving the capability of the SP system with a specified configuration. It is because the arrivals of track and confirmation tasks are highly dynamic. In other words, a hard real-time SP system under the traditional approach is very difficult to verify its performance specifications. With its performance usually being verified empirically, engineers are forced to dump much more processing units than what they actually need.

TABLE 2
Timing Parameters of an Example Search Task

	R	$(SD \cdot SI)$	Max	Min	K	P
Parameters	0	$3SI$	3	1	3	6

The purpose of this section is to first present a simple but effective task allocation policy in a typical component-oriented SP, which shows some useful properties, and then apply the earliest deadline first (EDF) algorithm [21] in a three level priority scheduling framework to schedule SP jobs in a real-time fashion. The rationale behind the separated EDF scheduling is to reflect the semantic importance of different tasks. Note that traditional FIFO and EDF scheduling policy cannot distinguish the importance of different jobs. We will then propose formulas for radar engineers to derive the number of VSPs possibly needed for a radar SP.

4.2 Real-Time Task Allocation and Scheduling Strategy

4.2.1 Task Allocation Policy—A Rule of Thumb

In a component-oriented SP, the processing time C_s of each nonpreemptive job of the search task can be much more than the length of a SI. On the other hand, the processing time of a track or confirmation task is much less than an SI. The relative deadline of track and confirmation tasks can be as close as $2SI$, while the relative deadline of the search task can be close to its big cycle, which is often much more than $2SI$. As a result, the number of VSPs in an SP is often more than the number of jobs issued by the search task in each SI during its peak duration.

The task allocation policy is motivated by the following observation: Suppose that there are three jobs issued by the search task in each SI during its peak duration. Let the system consist of five VSPs and the ready time R of each job of the search task be zero (similar observation can be obtained even if R is nonzero). Assume that each job of the search task needs $C_s = 1\frac{1}{2}SI$ amount of processing time, as shown in Table 2.

Fig. 3 shows the executions of the jobs issued by the search task, where the search task has a higher priority than other tasks, and a priority-driven scheduling policy is adopted. The jobs issued by the search task at the ith SI are marked by "i." Since the relative deadline of track and confirmation tasks can be as close as $2SI$, we observe that the sum of available time for tasks other than the search task within every two consecutive periods varies significantly. That is, the minimum sum of available time for track and confirmation tasks within every two consecutive periods is reduced. For example, the minimum sum of available time for track and confirmation tasks happens at the second and third periods (the total is only $1\frac{1}{2}SI$). On the contrary, if we pack the jobs of the search task as much as we can (on a small number of VSPs, which will be discussed later) and always assign jobs to the VSPs which are available first, then the sum of available time for track and confirmation tasks within every two consecutive periods tends to be

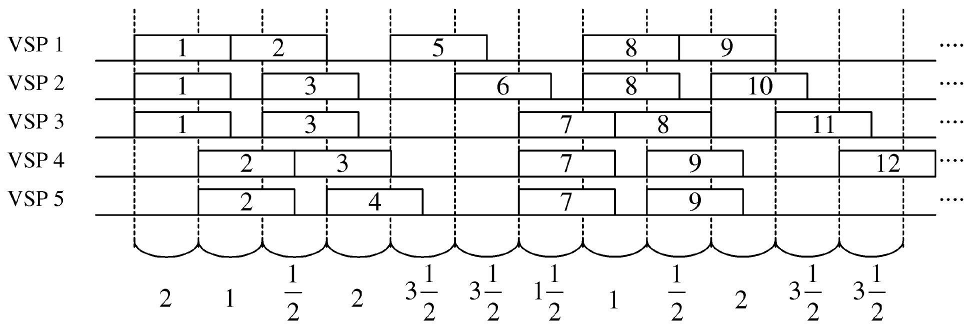


Fig. 3. The execution of the jobs issued by the search task when the jobs may run on all VSPs and $C_s = 1\frac{1}{2}SI$.

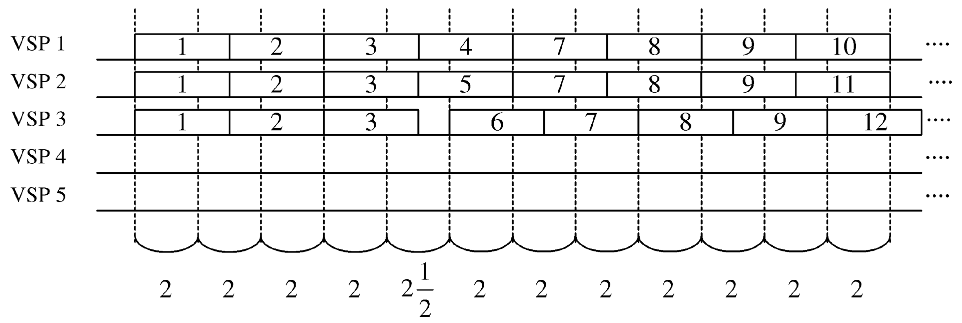


Fig. 4. The execution of the jobs issued by the search task when the jobs only run on the three selected VSPs and $C_s = 1\frac{1}{2}SI$.

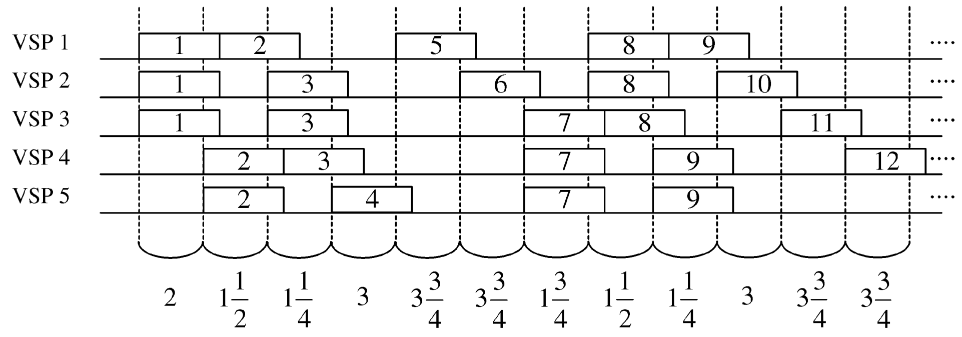


Fig. 5. The execution of the jobs issued by the search task when the jobs may run on all VSPs and $C_s = 1\frac{1}{4}SI$.

more evenly distributed (which is no less than $4SI$), as shown in Fig. 4.

As astute readers may notice, when the processing time C_s of the jobs of the search task is reduced, e.g., $C_s = 1\frac{1}{4}SI$, the situation remains. Fig. 5 shows the executions of the jobs issued by the search task when $C_s = 1\frac{1}{4}SI$. The minimum sum of available time for track and confirmation tasks also happens at the second and third periods (the total is $2\frac{3}{4}SI$) when jobs of the search task are scattered over all VSPs. On the contrary, if we pack the jobs of the search task as much as we can, then the sum of available time for other tasks within every two consecutive periods is more evenly distributed (which is no less than $4SI$), as shown in Fig. 6. It is also obvious that, when the processing time C_s of the jobs of the search task is increased, e.g., $C_s = 2SI$, the minimum sum of available time for track and confirmation tasks within every two consecutive periods is reduced to zero, compared to the situation when $C_s = 1\frac{1}{2}SI$ (a similar figure can be derived from Fig. 3). In other words, critical track and confirmation tasks might suffer from "transient overload" and miss their deadlines when jobs of the search task may run on all VSPs!

When the processing time C_s of the jobs of the search task is under SI, it is clear that the minimum sum of available time for track and confirmation tasks within every two consecutive periods remain the same, regardless of whether jobs of the search task are scattered over all VSPs or not (please see Fig. 7). However, we must point out that, since the processing time of track and confirmation tasks often divides SI, the job packing approach still tends to provide better performance on average because track and confirmation tasks are given entire SIs under the packing approach.

The observation in the previous paragraphs motivates a rule of thumb for task allocation in SPs called the *Job Packing Policy*, where traditionally approaches may have search jobs processed on all VSPs:

Rule 1. *All jobs of the search task should be packed on a collection of VSPs as much as possible.*

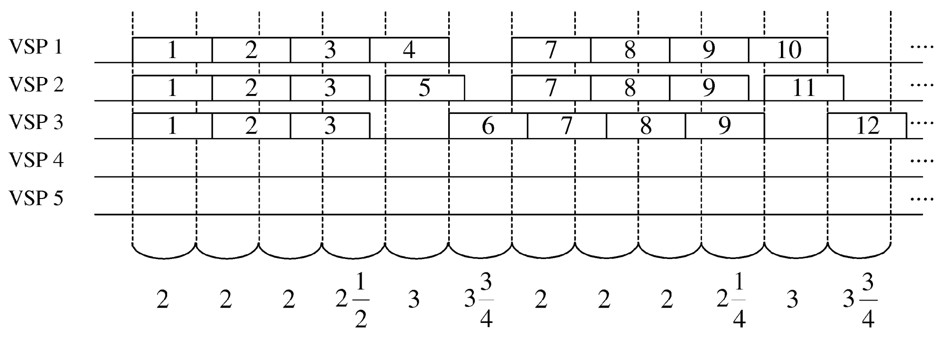


Fig. 6. The execution of the jobs issued by the search task when the jobs only run on the three selected VSPs and $C_s = 1\frac{1}{4}SI$.

In the next section, we shall show that the well-known *earliest deadline first* (EDF) algorithm [21], which assigns a higher priority to any task with a closer deadline, is preferred in scheduling jobs of the search task over multiple VSPs. We must point out that we do not intend to claim that the above packing policy is an optimal solution in general (although some nice properties with the certain real-time scheduling algorithm can be proven later). The observation in previous paragraphs does suggest that the packing approach usually provides a more even distribution in the amount of available processing time for track and confirmation tasks. We should also emphasize that the search task is the only periodic task at the SP level. With the workload separation of the search task and other tasks, the Job Packing Policy has an advantage in estimating and deriving the capacity of the system. As astute readers may notice, the above policy remains good when the ready time parameter R of the search task is not zero, where R is usually relatively small, compared to SI. Most importantly, the job packing approach provides a better way of estimating the system capability in servicing search and track tasks, which will be elaborated in later sections. For the rest of this paper, we assume that jobs of the search task are allocated by the Job Packing Policy.

4.2.2 A Three-Level Deadline-Driven Scheduling Policy

In many phased array radars, periodic search tasks, which are responsible for searching suspicious targets in the surveillance space, usually have higher priorities than track and any other tasks at both RCC and SP levels. The minimum operation requirements often focus on the schedulability of the search task. The priorities of all SP tasks can be, in general, classified in the way that the search task has the highest priority and track tasks have the lowest priority [12].

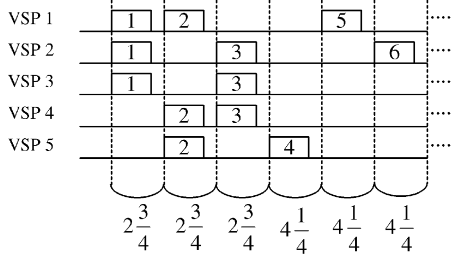


Fig. 7. The execution of the jobs issued by the search task when the jobs may run on all VSPs and $C_s = \frac{3}{4}SI$.

In this section, we shall show, as follows, that the well-known *earliest deadline first* (EDF) algorithm [21], which assigns a higher priority to any task with a closer deadline, is optimal in scheduling tasks (or jobs) of the same type in a nonpreemptive fashion over multiple VSPs. EDF is applied under a greedy policy in which a job (or task) is always assigned to the VSP which is assigned to the task type and is available first. We call such policy *Leveled EDF* (LEDF). The VSP assigned to a task type is called a selected VSP for the task type. Note that track and confirmation tasks can utilize the available processing time of the search-task-selected VSPs left by the search task. We can show the following properties of LEDF:

Theorem 1. *Leveled EDF is optimal in scheduling jobs of the search task over the selected VSPs in a nonpreemptive fashion.*

Proof. When there is any selected VSP available, we can always move the execution of any ready job of the search task forward (without affecting the feasibility of the schedule) because every search job has the same processing time C_s. If any two jobs of the search task in a feasible schedule are out of the deadline order, we can always swap their assigned processing time and VSPs without affecting the feasibility of the schedule. □

Corollary 1. *The Job Packing Policy with Leveled EDF is optimal in the sense that if a task allocation and scheduling policy can schedule jobs of the search task on N VSPs, then the Job Packing Policy with Leveled EDF can schedule jobs of the search task on N VSPs.*

Proof. The correctness of this corollary directly follows from the optimality of the Leveled EDF algorithm (please see Theorem 1). □

Theorem 2. *Leveled EDF is optimal in scheduling confirmation tasks on the available processing time (left in scheduling jobs of the search task) of all VSPs in a nonpreemptive fashion.*

Proof. Because every confirmation task has the same processing time C_c, the theorem can be proven in the same way as Theorem 1. □

Theroem 3. *Leveled EDF is optimal in scheduling track tasks on the available processing time (left in scheduling jobs of the search task and confirmation tasks) of all VSPs in a nonpreemptive fashion.*

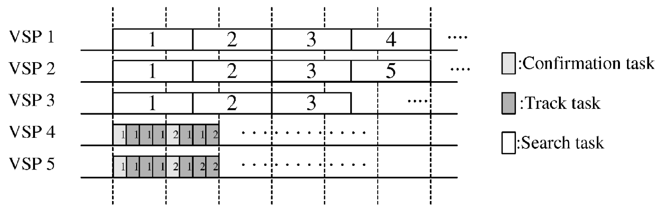


Fig. 8. A Leveled EDF schedule when the jobs of the search task only run on selected VSPs.

Proof. Because every track task has the same processing time C_t, the theorem can be proven in the same way as Theorem 1. □

Example 1: A Leveled EDF schedule. Let an SP system consist of five VSPs, the peak duration of the search task have three jobs, and the ready time R of each job of the search task be zero. Suppose that each job of the search task needs $C_s = 1\frac{1}{2}SI$ amount of processing time, as shown in Table 2. During the peak duration, the search task issues three jobs per SI. Outside the peak duration, the search task issues one job per SI. Let the processing time of a confirmation task and a track task be both equal to $C_c = C_t = \frac{1}{4}SI$.

Suppose that nine track tasks and two confirmation tasks occur at the first SI, and three track tasks and two confirmation tasks occur at the second SI. Let their relative deadlines be $2SI$. Fig. 8 shows the execution of the SP scheduled by the Leveled EDF algorithm when jobs of the search task only execute on the first three VSPs. All tasks and jobs meet their deadline. For comparison, Fig. 9 shows the execution of the SP scheduled by the Leveled EDF algorithm when jobs of the search task may execute on all VSPs. It was shown that one track task issued in the first SI misses its deadline at the end of the second SI. Obviously, it is due to the even distribution of available processing time left by the search task.

Note that, although Leveled EDF is optimal in scheduling each individual type of tasks (/jobs) on a multi-VSP SP, it is, in general, not an optimal scheduling algorithm in the mixed scheduling of search jobs, track tasks, and confirmation tasks. Note that Dhall [5] showed that EDF is not optimal for multiprocessor environments, and Mok [22] showed that real-time scheduling with mutual exclusion is, in general, NP-hard. We must also point out that a simple EDF policy may not be suitable in SP scheduling because the highly dynamic nature of the arrival patterns of track and confirmation tasks may result in possible violations of the radar minimum operation, which is often on the schedulability guarantee of the search task. As astute readers may point out, when an admission control mechanism is provided to manage the workload of confirmation and track tasks, the schedulability of the search task might still be guaranteed, although the system capacity estimation might become complicated. However, we must emphasize that Leveled EDF is a simple and intuitive algorithm which results in an easy estimation of the system capacity and good performance (to be illustrated later).

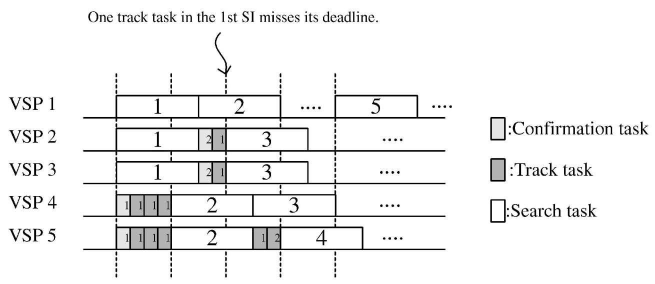


Fig. 9. A Leveled EDF schedule when the jobs of the search task may run on all VSPs.

4.3 System Capacity Estimation

The purpose of this section is to derive the bounds on the number of VSPs needed to schedule digital signal processing under the Job Packing Policy and Leveled EDF scheduling algorithm to satisfy the minimum operation requirements.

4.3.1 *An Upper Bound for the Number of VSPs for the Search Task*

Let N be the number of internal nodes assigned to the search task $\tau^{search} = \{R, C_s, (SD \cdot SI), Max, Min, K, P\}$.

Theorem 4. *When $C_s \leq SI$, the number of VSPs needed for the search task is less than or equal to Max.*

Proof. The correctness of this theorem follows from the fact that, given Max VSPs, every job J_i of the search task (within or outside its peak duration) can be processed immediately after it is ready and will be finished before the corresponding job J_i in the next SI. Since there does not exist any SP which can do better than that, we conclude that the maximum number of VSPs needed for the search task is no more than Max. □

We must point out that Theorem 4 only provides an intuitive idea on the maximum number of VSPs needed for the search task. When $C_s = SI$ and $K = P$, the number of VSPs needed for the search task is exactly Max. When $C_s < SI$, the number of VSPs needed for the search task is, in fact, a simple variation of the well-known bin packing problem [8], [26]. Since C_s in many systems is often larger than SI, we shall focus on the case when $C_s > SI$. We refer interested readers to [8], [26] for solutions of the bin packing problem.

Suppose that the number of VSPs reserved for the search task in an SP is N. For the rest of this section, we shall consider a practical case in which $SI \leq C_s \leq 2SI$. Let C_s be equal to $SI + (X \cdot R)$, where X is any positive integer such that $SI \leq C_s \leq 2SI$. That is, $(X \cdot R) \leq SI$ and $N < 2Max$. Let $J_{i,j}$ denote the jth job of the search task issued at the ith SI. Note that the ready time of the jth job is $(j \cdot R)$ from the beginning of its issuing SI, and $(Max \cdot R) \leq SI$ because jobs (including the Maxth job) issued in an SI must be ready inside the SI.

Lemma 1. *When $K = P$, $J_{i,j}$ is allocated to the $(([(i-1) \cdot Max + j - 1] \bmod N) + 1)$th VSP.*

Proof. It follows from the definitions of the Job Packing Policy and the Leveled EDF algorithm and the fact that

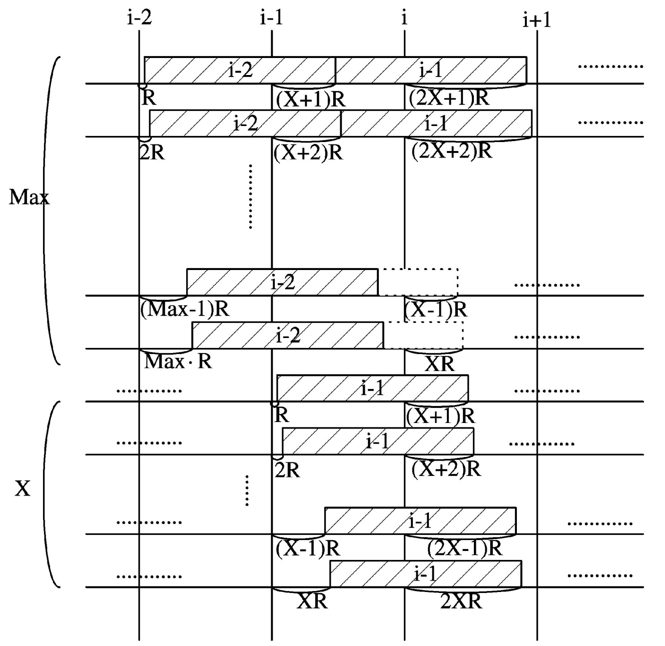


Fig. 10. A Leveled EDF schedule when jobs of the search task run on $(Max + X)$ VSPs.

there are $[(i-1)\cdot Max + j - 1]$ jobs in front of job $J_{i,j}$. Note that $C_s > SI$. □

Lemma 2. *When $K = P$ and the number of VSPs located for the search task is $(Max + X)$, every job $J_{i,j}$ of the search task will finish no later than $(SI + (2Max + X - j)\cdot R)$ from the beginning of its issuing SI.*

Proof. Without losing the generality, we might renumber the VSPs such that $J_{i-2,j}$ is processed at the jth VSP. A repeated job execution pattern in SIs can be observed in Fig. 10 (because of the periodicity of job allocation on the $(Max + X)$ VSPs) that, for $1 \le j \le (Max - X)$, $J_{(i-2),j}$ issued at the $(i-2)$th SI must finish no later than the time $(((i-1)\cdot SI) + ((X+j)\cdot R))$. In other words, the first $(Max - X)$ VSPs will be ready for the last $(Max - X)$ jobs in the $(i-1)$th SI. An observation from Fig. 10 shows that Job $J_{(i-2),Max}$ issued at the $(i-2)$th SI must finish no later than the time $(((i-1)\cdot SI) + (Max + X)\cdot R) \le ((i \cdot SI) + (X \cdot R))$ because $(Max \cdot R) \le SI$. Therefore, we conclude that, for $(Max - X + 1) \le j \le Max$, $J_{(i-2),j}$ issued at the $(i-2)$th SI will finish no later than the time $((i\cdot SI) + ((X - (Max - j))\cdot R))$. Therefore, within every ith SI, there are always Max VSPs in which at least one VSP is available immediately when a job comes in. Therefore, any job $J_{i,j}$ will not start later than $(j \cdot R)$ from the beginning of its issuing SI (with the situation repeated in every SI, as shown in Fig. 10). Since the latest job which can start is the Maxth job in each SI and each job needs $C_s = SI + (X \cdot R)$, every job $J_{i,j}$ of the search task will finish no later than $(SI + (Max + X + (Max - j))\cdot R)$ from the beginning of its issuing SI. □

Given the fact that $C_s > SI$, the relative deadline of jobs of the search task is no less than $2SI$. We can show the following theorem:

Theorem 5. *When $K = P$, the minimum number of VSPs needed for the search task is no more than $(Max + X)$.*

Proof. Since the jth job in each SI cannot finish until $(SI + (2Max + X - j)\cdot R)$ from the beginning of its issuing SI in any possible system, we conclude that $(Max + X)$ VSPs can schedule the search job. □

Note that the bound $(Max + X)$ in Theorem 5 is pretty tight when $K = P$ because every job fits in the right spot without causing any job issued in the next SI any delay in processing. If there is any bad delay, there can be a propagation of delays such that, eventually, some search job will miss its deadline, regardless of how far the deadline is.

Corollary 2. *When $K \le P$, the minimum number of VSPs needed for the search task is no more than $(Max + X)$.*

Proof. When $K \le P$, the number of VSPs available to immediately process jobs (whenever they are ready) outside the peak duration of an SI will be more than Min. Therefore, it can been shown with the same argument in Lemma 2 that every job $J_{i,j}$ of the search task will finish no later than $(SI + (Min + X)\cdot R)$ from the beginning of its issuing SI, where $Max > Min$. Therefore, the minimum number of VSPs needed for the search task is no more than $(Max + X)$. □

4.3.2 A Lower Bound for the Number of VSPs for the Search Task

Theorem 5 and Corollary 2 provide a good intuition for the understanding of job packing on VSPs, and an upper bound for the number of VSPs needed for the search task is provided. We can show the following theorems to derive a lower bound so that engineers may reduce the search range of the number of VSPs needed for the search task.

Theorem 6. *When $P = K$, the number of VSPs needed for the search task is no less than N if $\lfloor \frac{P \cdot Max}{N} \rfloor \cdot C_s \ge (P \cdot SI)$.*

Proof. When the number of VSPs allocated for the search task is N and $P = K$, the minimum of the processing time needed on each VSP is $\lfloor \frac{P \cdot Max}{N} \rfloor \cdot C_s$. If the needed processing time cannot be completed within each P periods, then the number of VSPs needed for the search task is more than N. Obviously, if $\lfloor \frac{P \cdot Max}{N} \rfloor \cdot C_s \ge (P \cdot SI)$, then even the most lightly-loaded VSP cannot afford the needed processing time. □

Theorem 7. *When $P > K$, the number of VSPs needed for the search task is no less than N if*

$$\left(\left\lfloor \frac{K \cdot Max}{N} \right\rfloor + \left\lfloor \frac{(P-K)\cdot Min}{N} \right\rfloor \right) \cdot C_s \ge (P \cdot SI).$$

Proof. The correctness of this theorem can be proven in a way similar to that of Theorem 6. □

4.3.3 Remark: Confirmation and Track Tasks

The purpose of this section is to consider the number of VSPs needed for confirmation and track tasks. There are usually two phases to consider in SP [2]: In the *normal phase*, the system operates as explained in Sections 2 and 3. An SP has a workload mixed of the search task, confirmation tasks, and track tasks. When the system is becoming fully loaded,

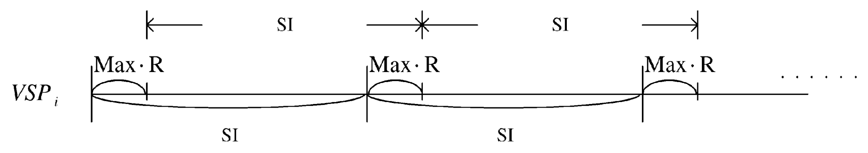


Fig. 11. The availability of VSP processing time in each SI.

the number of confirmation tasks decreases very quickly. It is because a large number of identified targets exist in the surveillance space of the phased array radar and RCC runs out of resources and cannot afford to trigger any more confirmation task. We called such a fully loaded phase the *fully loaded phase*. Note that, even during the fully loaded phase, the workload of the search task usually remains similar because the search and related information in the surveillance space is still needed for various reasons.

As explained in Sections 2 and 3, SPC receives commands from RCC for beam transmissions and signal processing. Due to the hardware constraints, the signals of all tasks are returned serially, and a task is ready only when its signal is returned by ASP. In other words, for all tasks in an SI, confirmation and track tasks are not ready until all jobs in the search task are ready because the search task has a higher priority than confirmation and track tasks at RCC and SPC levels. In the worst case (i.e., the peak duration), the ready times of all incoming confirmation and track tasks are after $(Max \cdot R)$. Note that VSPs for confirmation and track tasks can still process other confirmation and track tasks pending in the previous SIs.

The *full capacity* of an SP with N identical VSPs (during a fully loaded phase) can be estimated easily, given M out of N VSPs being reserved for the search task. Because the processing time C_t of a track task usually can divide SI, the (minimum) full capacity of an SP in target tracking can be defined as $\frac{(N-M)\cdot SI}{C_t}$ per SI or $\frac{(N-M)\cdot 2SI}{C_t}$ per two consecutive SIs or anything similar. Note that, although the processing of incoming track tasks might be delayed by the ready times of jobs of the search task, there is still one full SI amount of time to process track tasks in each SI, as shown in Fig. 11.

The capacity estimation of an SP with N identical VSPs during a normal phase is more difficult. It is more like a bin packing problem with two item sizes, i.e., C_c and C_t [8], [16], [26]. Since the processing time, i.e., C_c and C_t, respectively, of confirmation and track tasks is relatively small, compared to SI, the capacity estimation of an SP can be done by using the maximum of C_c and C_t, and the formula similar to those for the full loaded phase can be derived, e.g., $\frac{(N-M)\cdot SI}{\max\{C_t,C_c\}}$ per SI.

5 Simulation Experiments

5.1 Data Set and Measurement

The experiments described in this section are meant to assess the capability of the Job Packing policy and the Leveled EDF algorithm in scheduling digital signal processing. We have implemented a simulation model for a component-oriented phased array radar SP. We compare the performance of the traditional FIFO-like scheduling mechanism (FIFO), the Leveled FIFO algorithm with and without the Job Packing policy (abbreviated as LFIFO-JP and LFIFO), the earliest deadline first algorithm (EDF), and the Leveled EDF algorithm with and without the Job Packing policy (abbreviated as LEDF-JP and LEDF) based on typical SP workload characteristics.

The primary performance metric is the number of VSPs, referred to as $VSP\ Number$. The smaller the $VSP\ Number$, the better the algorithm is. Note that a smaller number of VSPs means a better system performance, and one VSP is capable of processing a large number of track tasks.

The test data sets were generated based on typical SP workload characteristics of a multifunction phased array radar for air defense frigates [12]. The functions of the radar included Horizon Search, Volumetric Search, Cued Search, track confirmation, air target track, weapon track, and normal track. Horizon Search, Volumetric Search, and Cued Search contributed to search tasks with a peak duration of six SIs, where the sum of the peak duration and the normal duration was equal to 128 SIs, and each SI was equal to 31.25 (i.e., $1,000/32$) ms. The existence of the peak duration was mainly due to the additional search jobs from Cued Search and the fact that the number of search jobs could not divide the number of SIs in an interval. Note that each instance of Cued Search was a sequence of five instances of Horizon search in the same direction. As a result, we set the ratio of search jobs between the peak duration and the normal duration as 0.25 and 0.5 (because the search jobs of the Volumetric Search also contributed to the workloads of search jobs; otherwise, the ratio should be $1/6$).

The deadlines of search jobs and track confirmation tasks were set based on the data update rates in [12]. The task deadlines of air target track, weapon track, and normal track were determined based on the considerations of the minimum time between the ready time of the input data at SP and the ready time of the results at RCC (i.e., 2SI for track tasks) and the mean update rates of track tasks in [12] (i.e., 30 SIs). The processing times of search jobs, confirmation tasks, and track tasks were set based on the ratios of the real execution times of the corresponding algorithms (such as pulse Doppler processing, moving target indication, tracking processing, and monopulse estimation) in [2], where the processing time of the search tasks was set as 1.5 or 2 SIs to explore the impacts of the processing time on the number of VSPs in Section 4.3.1. The ready times of search jobs, track tasks, and confirmation tasks were set based on the number of dwells in one SI [12]. The ready times of track tasks were smaller than those of search jobs and confirmation tasks to reflect the lengths of their dwell lengths. The experiments simulated task sets with a number of tracks between 400 to $4,000$, which represented a range of 4 to 40

TABLE 3
Parameters of Simulation Experiments

parameters	value
The number of search jobs per SI during the peak duration	6
The big cycle of the search task	128 SI
The ratio of the peak duration in a big cycle	0.2 or 0.33
The ratio of job numbers in the peak and normal duration	0.25 or 0.5
The deadline of each search job	6 SI
The deadline of each confirmation task	6 SI
The deadline of each track task	$(2, 30)$ SI
The processing time of a search job C_s	1.5 or 2 SI
The processing time of a confirmation task C_c	0.25 or 0.33 SI
The processing time of a track task C_t	0.125 or 0.25 SI
The ready time of a search job R_s	0.1 SI
The ready time of a confirmation task R_c	0.1 SI
The ready time of a track task R_t	0.05 SI
The arrival pattern of each track task	Poisson distribution (mean=100 SI's, var = 100 SI)
The arrival pattern of each confirmation task	Poisson distribution (mean=25 SI, var = 25 SI)
Simulation time	40,000 SI

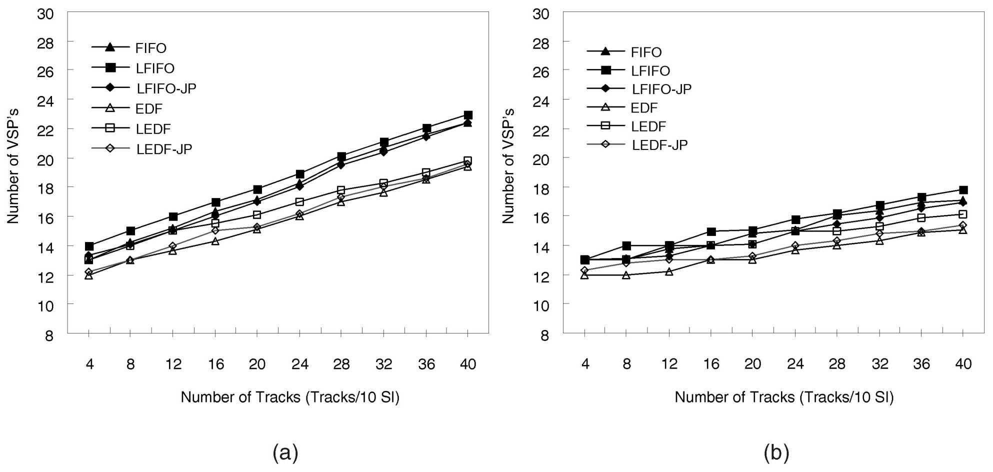


Fig. 12. The ratio of the peak duration in a big cycle = 0.2, the ratio of peak and normal workload = 0.5, $C_s = 2SI$.

tracks per 10 SIs, where the arrival pattern of a track task had a Poisson distribution with a mean equal to 100 and a variance equal to 100. The ratio of track tasks and confirmation tasks was about 4. Each task set was simulated for $40,000$ SI. Ten task sets per track workload were tested, and their results were averaged.

The simulation experiments were repeated for each scheduling algorithm for the minimum number of VSPs under each given workload. If a scheduling algorithm could not schedule a given workload under a specific number of VSPs, then the number of VSPs was increased by one, and the same simulation was rerun for the workload. The procedure was repeated until the given workload was schedulable on the specified number of VSPs. The parameters are summarized in Table 3.

5.2 Experimental Results: System Capacity

We considered workloads with and without confirmation tasks. Workloads without confirmation tasks were for a fully loaded phase, where no new target tracking could be afforded. Workloads with confirmation tasks were for a normal phase. Experimental results in Figs. 12, 13, and 14 considered a fully loaded phase. The rest of the figures considered a normal phase.

Figs. 12a and 12b show the number of needed VSPs (for FIFO, LFIFO, LFIFO-JP, EDF, LEDF, and LEDF-JP), when the processing time C_t of a track task was equal to $0.25SI$ and $0.125SI$, respectively. When the number of tracks per 10 SIs increased, the performance difference between real-time scheduling algorithms, such as LEDF-JP and EDF, and traditional scheduling policy FIFO increased. The performance of LEDF-JP and EDF were similar, and they were better than LEDF (which did not consider job packing), especially when the workload ratio between search jobs and tracking tasks was high (i.e., when the number of tracking tasks was small). LEDF-JP also outperformed LFIFO and LFIFO-JP. In Fig. 12, we could find out that LFIFO had the worst performance among all algorithms; LFIFO always

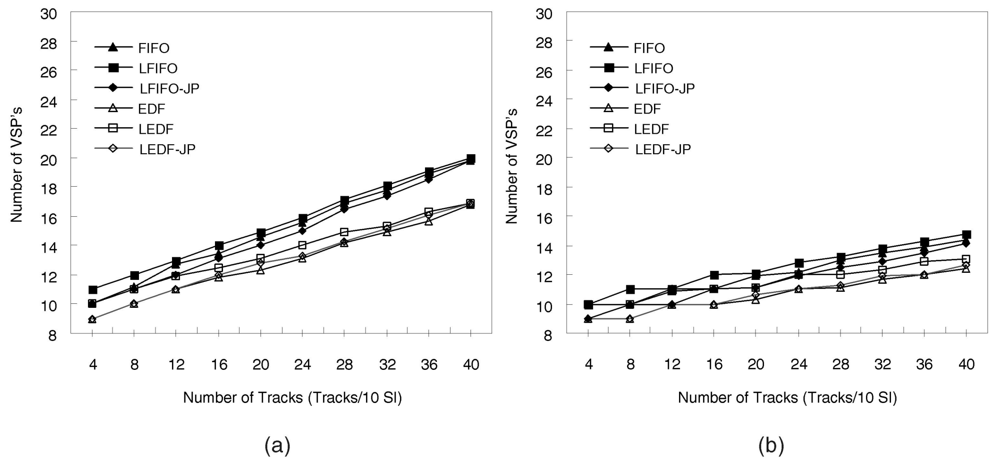


Fig. 13. The ratio of the peak duration in a big cycle = 0.2, the ratio of peak and normal workload = 0.5, $C_s = 1.5SI$.

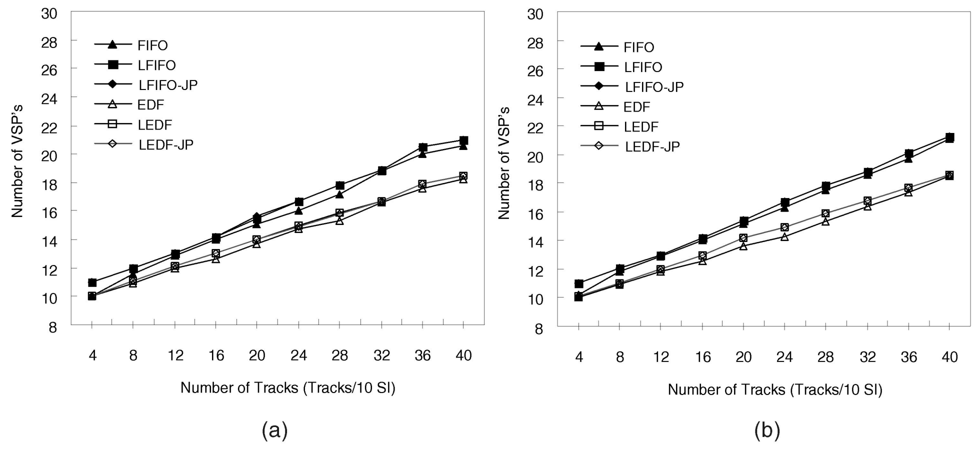


Fig. 14. The ratio of the peak duration in a big cycle = 0.33, $C_s = 1.5SI$, and $C_t = 0.25SI$, where $R_{P/N}$ denotes the ratio of peak and normal workload.

needed more VSPs than other algorithms. Note that one additional VSP could much improve the capacity of the system in terms of the number of track and confirmation tasks. We must also emphasize that EDF does not consider the semantic importance of tasks in digital signal processing and might suffer from the burst of the arrivals of track and confirmation tasks. The results shown in Fig. 13 were similar to those in Fig. 12, except that the processing time of a search job $C_s = 1.5SI$. Because C_s was smaller in Fig. 13, the number of VSPs needed for each scheduling algorithm was smaller.

Figs. 14a and 14b show the number of needed VSPs (for FIFO, LFIFO, LFIFO-JP, EDF, LEDF, and LEDF-JP) when the ratios of peak and normal workload were equal to 0.5 and 0.25, respectively. The performances of LEDF-JP and LEDF were almost the same, and LFIFO-JP had the same performance as LFIFO. The main difference of the simulation parameters between Fig. 14 and the previous two figures was the length of the peak duration. The results in Fig. 14 show that, when the peak duration was longer in a big cycle, the advantage of the Job Packing policy was reduced. It was because a very large portion of VSPs would be reserved for the search task under the Job Packing policy, and track tasks only used a small number of VSPs. As a result, less improvement could be noticed. Experiments for Fig. 15 were as the same as those for Fig. 14, except that confirmation tasks were considered. The results were similar.

When confirmation tasks were considered, Fig. 16 shows the number of needed VSPs (for FIFO, LFIFO, LFIFO-JP, EDF, LEDF, and LEDF-JP), when the processing time C_t of a track task was equal to $0.25SI$ and $0.125SI$, respectively. Note that the experiments in Figs. 13 and 16 only differed in the consideration of confirmation tasks. Obviously, when a normal phase was considered, the advantage of the Job Packing policy was shown. Real-time scheduling algorithms still greatly outperformed the traditional FIFO-like algorithm. The performance of LEDF-JP was close to that of EDF. Note that, although the improvement on the number of VSPs was small, the increased number of tracking tasks was large. Compared to results in Fig. 16, Fig. 17 had a larger C_s and C_c. The performance differences of simulated algorithms were similar. The experiments in Figs. 18 and 16 only differed in the processing time of a search job C_s. The performance differences of simulated algorithms were similar, except that more VSPs were needed for algorithms when C_s was larger.

We conclude that real-time scheduling algorithms did greatly improve the performance of SP, compared to the

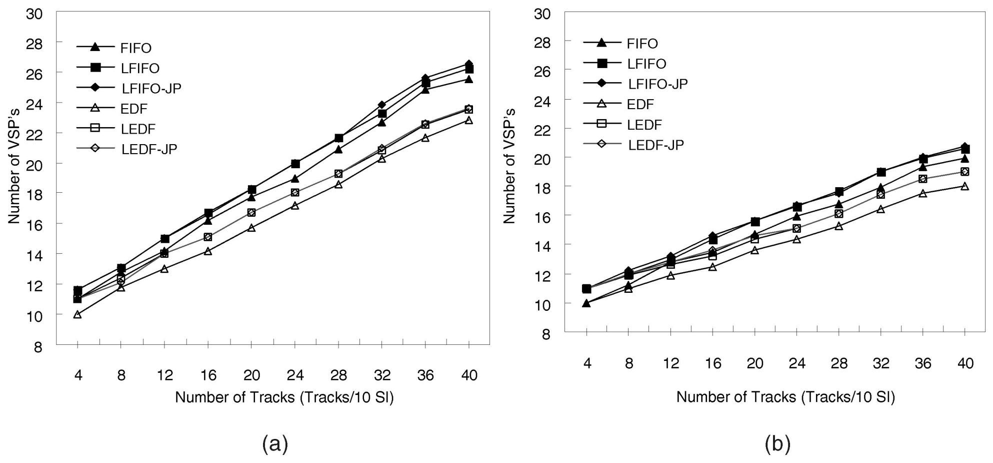


Fig. 15. The ratio of the peak duration in a big cycle = 0.33, the ratio of peak and normal workload = 0.5, $C_s = 1.5SI$, and $C_c = 0.33SI$.

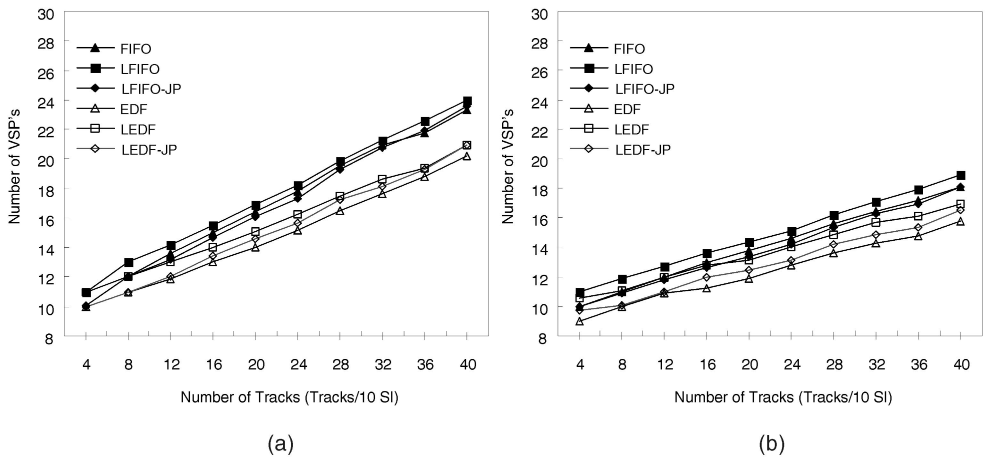


Fig. 16. The ratio of the peak duration in a big cycle = 0.2, the ratio of peak and normal workload = 0.5, $C_s = 1.5SI$, and $C_c = 0.25SI$.

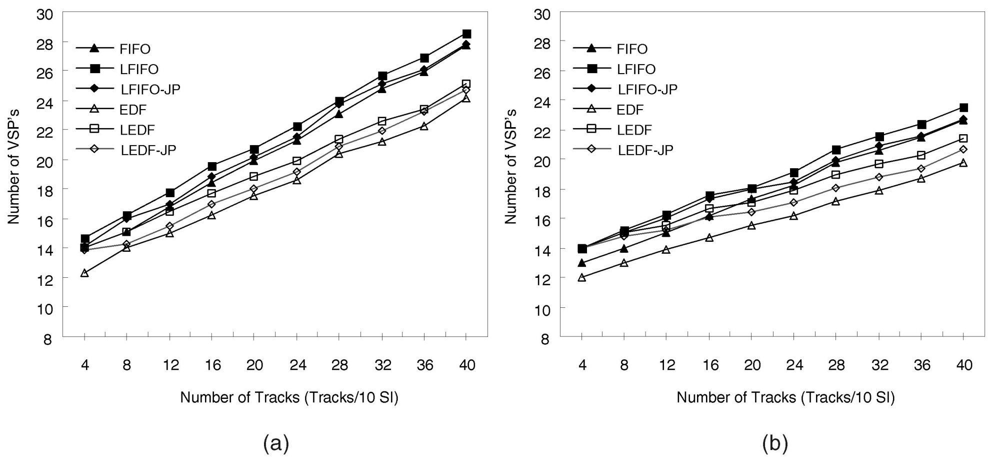


Fig. 17. The ratio of the peak duration in a big cycle = 0.2, the ratio of peak and normal workload = 0.5, $C_s = 2SI$, $C_c = 0.33SI$.

traditional FIFO-like scheduling algorithm. The Job Packing Policy improved the SP performance significantly. Note that, although the improvement on the number of VSPs was small, the increased number of tracking tasks was large. It was shown that LEDF-JP provided a reasonable performance, compared to EDF, where EDF may not be suitable in real-time SP scheduling because it could not guarantee the radar minimum operation requirements.

6 Conclusion

This work is one of the first attempts in building a truly real-time SP for the next-generation phased array radars. We target the two most important issues in the design of a typical component-oriented SP: the schedulability of critical tasks and the system capacity estimation. We first formalize the workload of a component-oriented SP for modern

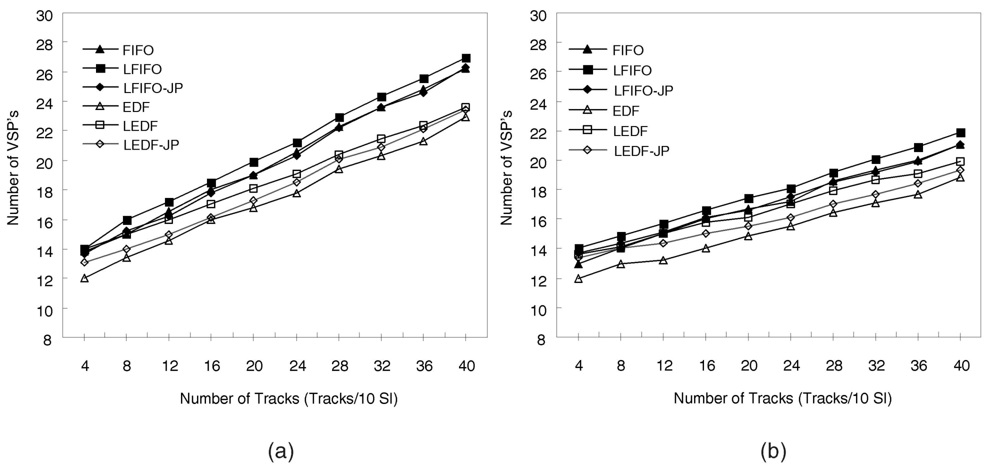


Fig. 18. The ratio of the peak duration in a big cycle = 0.2, the ratio of peak and normal workload = 0.5, $C_s = 2SI$, $C_c = 0.25SI$.

phased array radars. We then propose a simple but effective task allocation policy which separates periodic and aperiodic workloads such that the number of processing units needed by an SP can be better bounded. We present a three-level EDF-based scheduling algorithm to reflect the semantic importance of tasks and to process digital signals in a real-time fashion. We also derive bounds for the number of processing units needed for an SP under performance specifications. We must emphasize that virtually everything was done empirically in the past. Although researchers have started considering real-time scheduling at the Radar Control Computer (RCC), e.g., [1], [7], [12], [14], [15], little work has been done in real-time scheduling of digital signal processing at the SP level. Inefficient resource allocation mechanisms like FIFO were adopted in the traditional SP design, and the system capacity was very difficult to estimate or estimated very conservatively.

This work is one of the pioneer works in building the next-generation SPs, especially with real-time technology. The strength of our methodology is verified by a series of computer simulations for which we have very encouraging results. Our proposed methodology has a performance close to the powerful EDF algorithm and, at the same time, follows the semantic importance of radar tasks and provides bounds on the number of VSPs needed for an SP under performance specifications. We must point out that a simple EDF policy may not be suitable in SP scheduling because the highly dynamic nature of track and confirmation tasks may result in possible violations of the radar minimum operation requirements (which is often on the schedulability guarantee of the search task). EDF with an extra admission control mechanism on confirmation and track tasks might also complicate the system capacity estimation.

For future work, we will tune up our methodology for radar systems of different scales and goals. We will start designing a multi-SP radar system and propose methodology in integrating task scheduling at RCC and SP levels. We shall also explore mechanisms for overload management.

Acknowledgments

The authors would like to thank Dr. Tai-Chung Wang at III and Dr. Yu-Lin Su at the Chung Shan Institute of Science and Technology, Taiwan, for their help in this work. This work was supported in part by a research grant from the Chung Shan Institute of Science and Technology and a research grant from the National Science Council under Grant NSC89-2213-E-194-011. This paper is an extended version of a conference paper which appeared in the Proceedings of the IEEE 21st Real-Time Systems Symposium, 2000.

References

[1] A. Barbato and P. Giustiniani, "An Improved Scheduling Algorithm for a Naval Phased Array Radar," *Proc. Radar 92 Int'l Conf.*, pp. 42-45, 1992.
[2] R.A. Baugh, *Computer Control of Modern Radars.* RCAM&SR-Moorestown Library 1973.
[3] G. Bernat and A. Burns, "Combining (n, m)-Hard Deadlines and Priority Scheduling," *Proc. IEEE 18th Real-Time Systems Symp.*, pp. 46-57, Dec. 1997.
[4] C. Chang and T.-C. Wang, "Use Object-Oriented Paradigm to Design a Programmable Radar Digital Signal Processor," *Proc. Third Workshop Object-Oriented Technology and Applications (OOTA '97),* Sept. 1997.
[5] S.K. Dhall, "Scheduling Periodic-Time-Critical Jobs on Single Processor and Multiprocessor Computing Systems," PhD thesis, Univ. of Illinois, Urbana, 1977.
[6] S.K. Dhall and C.L. Liu, "On a Real-Time Scheduling Problem," *Operations Research*, vol. 26, no. 1, pp. 127-140, 1978.
[7] R. Filippi and S. Pardini, "An Example of Resources Management in a Multifunctional Rotating Phased Array Radar," *Real-Time Management of Adaptive Radar Systems, IEE Colloquium*, pp. 2/1-2/3, 1990.
[8] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness.* San Francisco: W.H. Freeman & Company, 1979.
[9] C.-C. Han and K.J. Lin, "Scheduling Distance-Constrained Real-Time Tasks," *Proc. IEEE 13th Real-Time Systems Symp.*, pp. 300-308, Dec. 1992.
[10] C.-W. Hsueh and K.-J. Lin, "An Optimal Pinwheel Scheduler Using the Single-Number Reduction Technique," *Proc. IEEE 17th Real-Time Systems Symp.*, pp. 196-205, Dec. 1996.
[11] C.-W. Hsueh and K.-J. Lin, "On-Line Schedulers for Pinwheel Tasks Using the Time-Driven Approach," *Proc. 10th Euromicro Real-Time Systems Conf.*, pp. 180-187, June 1998.

[12] A.G. Huizing and A.A.F. Bloemen, "An Efficient Scheduling Algorithm for a Multifunction Radar," *Proc. IEEE Int'l Radar Conf.*, pp. 359-364, 1996.

[13] A. Izquierdo-Fuente and J.R. Casar-Corredera, "Approach to Multifunction Radar Scheduling Simulation," *Proc. IEEE Telesystems Conf.*, pp. 67-70, 1995.

[14] K. Jeffay and S. Goddard, "A Theory of Rate-Based Execution," *Proc. IEEE 20th Real-Time Systems Symp.*, pp. 304-314, Dec. 1999.

[15] D.-I. Kang, R. Gerber, and M. Saksena, "Performance-Based Design of Distributed Real-Time Systems," *Proc. IEEE Third Real-Time Technology and Applications Symp.*, pp. 2-13, June 1997.

[16] R.M. Karp, "Reducibility Among Combinatorial Problems," *Complexity of Computer Computations*, pp. 85-103, 1972.

[17] G. Koren and D. Shasha, "Skip-Over: Algorithms and Complexity for Overloaded Systems that Allow Skips," *Proc. IEEE 16th Real-Time Systems Symp.*, pp. 110-117, Dec. 1995.

[18] T.-W. Kuo and A.K. Mok, "Load Adjustment in Adaptive Real-Time Systems," *Proc. IEEE 12th Real-Time Systems Symp.*, pp. 160-171, Dec. 1991.

[19] T.-W. Kuo, W.-R. Yang, and K.-J. Lin, "Egps: A Class of Real-Time Scheduling Algorithms Based on Processor Sharing," *Proc. 10th Euromicro Workshop Real-Time Systems*, pp. 27-34, June 1998.

[20] K.J. Lin, S. Natarajan, and J.W.-S. Liu, "Imprecise Results: Utilizing Partial Computations in Real-Time Systems," *Proc. IEEE Eighth Real-Time Systems Symp.*, pp. 210-217, Dec. 1987.

[21] C.L. Liu and J.W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment," *J. ACM*, vol. 20, no. 1, pp. 46-61, Jan. 1973.

[22] A.K. Mok, "Fundamental Design Problems for the Hard Real-Time Environment," PhD thesis, MIT, Cambridge, Mass., 1983.

[23] A.K. Mok and D. Chen, "A Multiframe Model for Real-Time Tasks," *IEEE Trans. Software Eng.*, vol. 23, no. 10, pp. 635-645, Oct. 1997.

[24] M.D. Natale and J.A. Stankovic, "Dynamic End-to-End Guarantees in Distributed Real-Time Systems," *Proc. IEEE 15th Real-Time Systems Symp.*, pp. 216-227, Dec. 1994.

[25] R.L. Nevin and F.W. Schatz, "An/Apg-67 Multimode Radar Development," *Proc. IEEE Int'l Radar Conf.*, pp. 1-8, 1985.

[26] C. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity.* Prentice-Hall, 1982.

[27] "Rapid Prototyping of Application Specific Signal Processors (RASSP)," http://www.eda.org/rassp, 2002.

[28] L. Sha, R. Rajkumar, and J.P. Lehoczky, "Priority Inheritance Protocols: An Approach to Real-Time Synchronization," *IEEE Trans. Computers*, vol. 39, no. 9, pp. 1175-1185, Sept. 1990.

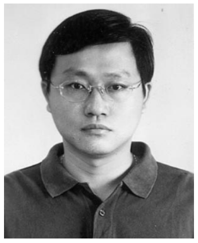
[29] M. Spuri, G. Buttazzo, and Sensini, "Scheduling Aperiodic Tasks in Dynamic Scheduling Environment," *Proc. IEEE 16th Real-Time Systems Symp.*, Dec. 1995.

[30] I. Stoica, H. Abdel-Wahab, and K. Jeffay, "A Proportional Share Resource Allocation Algorithm for Real-Time, Time-Shared Systems," *Proc. IEEE Real-Time Systems Symp.*, pp. 288-299, Dec. 1996.

[31] D. Stromberg and P. Grahn, "Scheduling of Tasks in Phased Array Radar," *Proc. IEEE Int'l Radar Conf.*, pp. 318-321, 1996.

[32] C.A. Waldspurger and W.E. Weihl, "Stride Scheduling Deterministic Proportional Share Resource Management," Technical Report, Technical Memoranduum, MIT/LCS/TM-528, Laboratory for CS, MIT, July 1995.

Chin-Fu Kuo received the BS and MS degrees from the Department of Computer Science and Information Engineering, National Chung Cheng University, in Chiayi, Taiwan, Republic of China, in 1998 and 2000, respectively. He is currently a PhD student in the Department of Computer Science and Information Engineering, National Taiwan University, in Taipei, Taiwan, Republic of China. His research interests include real-time systems and the indexing and querying of databases.



Tei-Wei Kuo received the 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, Republic of China. He was an associate professor in the Department of Computer Science and Information Engineering of the National Chung Cheng University, Taiwan, Republic of China, from August 1994 to July 2000. His research interests include real-time databases, real-time process scheduling, real-time operating systems, and embedded systems. He was the program cochair of the IEEE Seventh Real-Time Technology and Applications Symposium, 2001, and has been an associate editor of the *Journal of Real-Time Systems* since 1998. He has consulted for government and industry on problems in various real-time systems design. Dr. Kuo is a senior member of the IEEE and a member of the Computer Society.

Cheng Chang received the BS degree in applied mathematics from Chung Cheng Institute of Technology in TaoYuan, Taiwan, in 1982. He received the MS degree in computer and decision science from National Tsing Hwa University in Hsing Chu, Taiwan, in 1987. He received the PhD degree in computer science from the University of Illinois at Urbana-Champaign in 1996. He is currently an associated scientist in the System Development Center of the Chung Shan Institute of Science and Technology, TaoYuan, Taiwan, Republic of China, and an assistant professor in the Department of Business Administration of the National Central University, TaoYuan, Taiwan, Republic of China. His research interests include real-time systems, software engineering, and project management.