Dual Face Phased Array Radar Scheduling with Multiple Constraints

Qiuhua Cao and John A. Stankovic
Computer Science Department
University of Virginia
{qhua,stankovic}@cs.virginia.edu

ABSTRACT
Tasks in many real-time applications can be scheduled by variations of rate monotonic or earliest deadline first algorithms. When this is possible, it is satisfying to have formal analysis and performance bounds underlying the use of these algorithms. However, in many applications the simultaneous set of constraints that must be satisfied makes these traditional solutions unsuitable. Practical solutions for these more complicated applications are important. In this paper we develop a novel integrated scheduling and allocation heuristic for a dual face phased array radar system. The realistic features of the radar system that must be simultaneously addressed include timeliness (worst case execution time, period, deadline), semantic importance, and physical constraints such as beam selection and frequency harmonics. The heuristic function we develop provides a very flexible way to incorporate these requirements into one single equation. Since scheduling high semantic importance tasks is paramount, we use the highest semantic importance tasks' success ratio as the major performance metric. Based on simulation results, we show that our static heuristic algorithm can schedule more than 91% of the highest semantic importance tasks at high frequency conflict degree even at heavy workloads. The result is 50% better than EDF and 31% better than an importance (IMP) based static priority scheduling algorithm where IMP is similar to various current approaches. For the online scheduling algorithm, our heuristic algorithm is 30% better than EDF and 20% better than IMP in terms of highest semantic importance tasks' success ratio at heavy workloads.

Categories and Subject Descriptors
C.3 [**Special-Purpose and Application-Based Systems**]: Real Time and Embedded System. D.4 [**Operating System**]: Processor Management – *scheduling*. J.7 [**Computers in Other Systems**]: Real Time.

General Terms
Performance, Management, Design, Experimentations

Keywords
Dual Phased Array Radars Systems, Real Time Systems, Scheduling, Resource Allocations, Heuristic Algorithms, Performance.

1. INTRODUCTION
Modern phased array radars are very sophisticated systems subject to an extremely difficult set of real-time resource requirements and constraints. For example, the dual face phased array radar in the ship-based application presented here has separate transmit and receive arrays. For the transmit array there are 64 beams. Beams may be scheduled individually, or in groups, or as subarrays. The beams are also separated according to function. The beams are classified according to their frequencies and polarizations. The S-band beams' frequencies are in the range [950MHz, 10000MHz]. The frequencies of L-band beams are in the range [1000MHz, 5000MHz]. 16 beams have S-band vertical polarization, 16 beams have L-band dual polarization, 32 beams have L-band vertical polarization and 2 of them have special purpose filters. The physical locations of S-band and L-band beams in the phased array radars are given in Figure 1 and Figure 2. The low band transmit aperture is composed of 8 by 8 subarrays. Each array is capable of supporting a single transmit beam. There are four types of subarrays, each with slightly different capabilities. The receiver aperture is also an array of 8 by 8 subarrays, each supporting reception in both the L-band and S-band. Each subarray has three beam formers, each attached to a separate receiver.

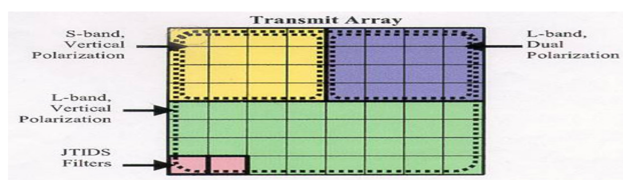

Figure 1: Transmit Array

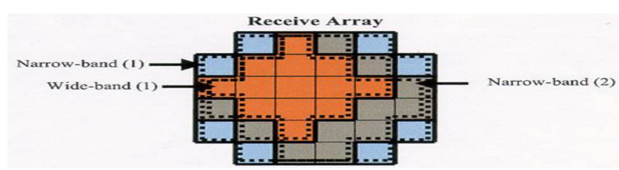

Figure 2: Receiver Array

Similar to a bistatic radar system [6], the dual face phased array radar system can perform the transmit function at one site and the receive function at another. It needs a communication link between the transmitter and the receiver. The timing signals and the transmitter's waveforms and frequencies are transferred over the link. The radar range and bearing of a target can be deduced from measurements of the time taken for the transmitted pulse to travel from the transmitter to the target and on to the receiver. The ambiguity of which received signal corresponds to which transmitted signal is solved by the minimum time interval requirements between two sequentially transmitted signals. The critical advantage of this dual face phased array system is that there is no need to schedule the receiver when scheduling the transmitter. In other words, the more traditional radar scheduling problem of scheduling dwells (both transmitter and receiver tasks) is not necessary. For more detailed information, please refer to [6]. In this paper, we focus on the scheduling problem on the transmitter side.

The scheduling problem for the dual face phased array radars is to allocate (groups of) beams and schedule radar tasks so that as many surveillance and tracking functions can occur and meet their deadlines as possible. Each radar task may require multiple beams to work together. This is a non-trivial and non-obvious problem because all the radar tasks are subject to multiple constraints. The constraints include execution time, period, deadline, energy (over heating during a period of time), frequency harmonics, and semantic importance (defined by the target's threat level) for each task as well as resource contention. The radar task must complete by its deadline at every period in order to keep track of the target. Losing track of the target leads to expensive overhead of searching for the target in a large space before the system can track the same target again. For this reason, it is costly to lose high semantic importance tasks. This requires the scheduling algorithm to meet as many high semantic importance tasks as possible under the various constraints we defined above.

It is well known that tasks in many real-time applications can be scheduled by variations of rate monotonic or earliest deadline first algorithms. When this is possible, it is satisfying to have formal analysis and performance bounds underlying the use of these algorithms. However, for some important applications the simultaneous set of constraints that must be satisfied make these traditional solutions unsuitable. Practical solutions for these more complicated applications are important even if there is no underlying theory. In this paper we develop a novel integrated scheduling and allocation heuristic for a dual face phased array radar system. The realistic features of the dual face phased array radar system discussed above must be addressed. It is not obvious how to account for all these issues in an effective manner. For example, current solutions use complicated runtime rules that are inefficient and underutilize the system.

The remainder of this paper is organized as follows. Section 2 provides a brief discussion of the state of the art. Section 3 describes the dual face phased array radar scheduling problem and challenges. In Section 4, the task model is presented. The heuristic function solution is presented in Section 5. In Section 6, we show the simulation results. We conclude in Section 7.

2. STATE OF THE ART

In the past three decades, many researchers have developed diverse scheduling algorithms for soft and hard real-time problems. The most frequently referenced scheduling algorithms are rate monotonic [16] and earliest deadline first scheduling [16] strategies. In [16], the authors show that rate monotonic is optimal among all fixed priority assignments. Liu and Layland [16] and Dertouzos [4] prove that EDF is optimal in a preemptive task model for one processor without resource contention. These algorithms, however, do not address the physical constraints such as frequency harmonics and semantic importance of a dual face phased array radar task. Buttazzo and Stankovic [3] propose a robust EDF scheduling algorithm (RED) for sporadic tasks under overloads in a hard real-time environment. RED accepts a task based on the earliest deadline first scheduling policy and rejects a task based on the task's value (i.e. the relative importance of the task with respect to the other tasks in the task set). However, RED only considers CPU resource constraints.

Radar tasks are non-preemptive. It has been shown in [11] and [13] that, in general, when preemption is not allowed and when tasks can have arbitrary arrivals, finding a feasible schedule and minimizing the maximum lateness is NP-hard. Branch-and-Bound algorithms such as [21] and [15] are usually proposed to deal with the case when arrival times are known a priori and when there is a non-preemptive requirement. In the average case, branch and bound performs well, but in the worst case it degrades to exponential complexity, which is not suitable for online scheduling.

In order to limit the search space and reduce the computational complexity of scheduling algorithms, two approaches are often used. One approach is to use additional information to prune the tree such as in Bratley's algorithm [2]. But this algorithm still has complexity of $O(n \cdot n!)$. So it is not suitable for online scheduling. In the second approach, found in [22] and [23], the authors proposed an algorithm to drive the search by a heuristic function H, which actively directs the scheduling to a plausible path. On each level of the search, an H function is applied to each of the tasks that remain to be scheduled. The task with the smallest value determined by the heuristic function H is selected to extend the current schedule. The heuristic function approach is a very flexible mechanism to define and modify a scheduling policy. One novelty of this paper is showing how to extend the H function approach by encoding the radar system constraints into a suitable H function.

Other work such as [5], [9], [12], [17], [18] also use heuristic approaches for related scheduling problems. However, none of the work has addressed the complexity involved with the radar scheduling problem presented in this paper.

There have also been a number of papers that explicitly deal with scheduling radars. Some of the work treats the problem as an abstract scheduling problem without considering all the realistic constraints. For example, Baugh [1] describes a best effort scheduling algorithm based on a radar task's semantic importance for a traditional multi-function phase array radar system. Shih et al. [21] present a scheduling template for dwell tasks with energy constraints. This template based dwell scheduling algorithm divides itself into two phases: one is the design phase, and the other is the scheduling phase. The design phase is conducted offline, and the scheduling phase is performed online. The major design phase uses a branch-and-bound algorithm. Gopalakrishman et al. [10] develop an online template construction to support both adaptive adjustment of the schedule and highly dynamic workloads. However, both [21] and [10] do not consider the frequency harmonics constraint and deal with single face radars.

Lee et al. [15] propose a concept of a scheduling envelope. This scheduling envelope is designed offline. It hides the complexity of the radar dwell scheduling and provides a simple measure for the schedulability check. The authors of [15] improve the system utilization by taking advantage of dwell interleaving.

Q-RAM [19] [20] present a novel analytical approach for satisfying multiple quality-of-service dimensions such as timeliness and reliable data delivery. They have also applied their work [7] [8] to radar scheduling for single face phased array radars. Ghosh et al. [7] [8] are based on the assumption that given a set of tasks, each of which is capable of running at one of several different QoS levels, the algorithms can select a QoS operating point. Since Q-RAM is a general and valuable tool, with suitable inventiveness it probably could be applied to the radar scheduling problem described here. To date, it has been used on a different radar problem from the dual face phased array radars described above. A possible future research problem could be to compare and contrast solutions like Q-RAM (an optimization based solution) with heuristic based solutions for radars. Just as RMA and EDF are suitable in different situations, we expect optimization versus heuristic based solutions to each have problems that they are most suited for.

Kuo et al. [14] propose a rate-based approach to schedule radar dwells, but the reservation ratio is computed based on each task's timeliness property. It does not take physical constraints into consideration for the algorithm design which is a very important aspect of the dual face phased array scheduling problem we solve in this paper.

3. THE CHALLENGES

In a dual face phased array radar system, radar tasks are periodic and non-preemptive. Tasks include both surveillance tasks that sweep across the sky to search for targets and tracking tasks. In order to keep track of a suspicious target, the radar task has to run to completion by a deadline without preemption. The nature of the dual face phased array radar system imposes some additional challenges to the real time scheduling algorithm to provide predictable performance.

The major scheduling challenges are similar to [21]: (1) each radar task has its own semantic importance defined by the level of threat from the tracked target. A higher the threat from the tracked target leads to a higher semantic importance of the corresponding task. Whereas in traditional real-time scheduling algorithms such as rate monotonic [16] and EDF [16], the priority or importance of the task is assigned based on the timeliness properties, here it is based on semantics. In a phased array radar system, the semantic importance of the task is more important than the priority defined by the timeliness property. So when the system is overloaded, we cannot discard the higher semantic importance tasks even when their timeliness requirement is lower. This property of a radar task requires favoring the higher semantic importance tasks. Most of the existing scheduling algorithms do not take the semantic importance into account. Our heuristic algorithm takes the semantic importance into consideration at the algorithm design level. (2) There is a highly dynamic system workload. A tracking task is generated whenever a new suspicious target appears in the surveillance space. But the tracked target's action is not predictable so that the tracking tasks' arrival is not known a priori. This requires a scheduling algorithm with the capability to determine whether a newly generated tracking task should be admitted or declined. It must be performed with low overhead in real-time. By contrast with offline scheduling algorithms proposed for radar systems, our heuristic algorithm can perform an online schedulability check with bounded overhead. (3) Physical constraints such as frequency harmonics must be addressed in the scheduling algorithm besides the execution time, deadline and period as well as resource constraints. In the single face phased array, every radar task is composed by two subtasks. One subtask is the transmit task. The other subtask is the receive subtask which performs signal processing upon receiving the returned signal. The two subtasks need to be executed in order and separated by enough physical time. In our dual face phased array radar system, there are separate transmit and receive radars. The transmitter and receiver can be scheduled separately.

All the challenges described above make it extremely difficult to provide predictable performance by using traditional scheduling algorithms. The contributions of this paper are:

1) A heuristic algorithm is defined such that it takes the most important properties of a dual face phased array radar task such as resources, timeliness, semantic importance and frequency harmonics into consideration.
2) The heuristic algorithm can provide an online schedulability check with bounded overhead.
3) The algorithm also makes resource allocation assignments.
4) The heuristic algorithm significantly improves performance.

4. TASK MODEL

Typically, in a dual face phased array radar system, the antenna of the transmitter sends electromagnetic signals with a predefined frequency. In order to avoid the frequency inference when multiple radar antennas transmit electromagnetic signals simultaneously, the frequencies assigned to the radar beams are subject to frequency harmonics constraints.

When a dual phased array radar antenna sends a signal, it causes heating of the radar panel. But the transmitting panel is susceptible to overheating. The task model in [10] [15] [21] captures the energy constraint due to possible overheating. This paper considers another important physical constraint, i.e., frequency harmonics. As we discussed above, it is costly to lose the high semantic importance tasks in dual face phased array radar systems. In our task model, we take the semantic importance into consideration. Radar tasks are non-preemptive periodic tasks as discussed above. Each transmitting task can have multiple instances. Each instance of a task is called a job as defined in [16]. The j-th instance of task T_i is referred to as job $J_{i,j}$. We define the timeliness, frequency harmonics and semantic importance as follows:

Definition 4.1. Timeliness Constraint: A transmitting task completes in time if and only if every instance (job) of the task meets its deadline.

Definition 4.2. Frequency Harmonics Constraint: In any time interval [t0, t1], if any two overlapping execution array antennas are transmitting electromagnetic signals with frequency f_1 and f_2 respectively, these two frequencies should satisfy equation

(2): $|f_1 - f_2| > f_{hs}$, where f_{hs} is the lowest value needed to avoid the frequency conflict. In this paper, we define f_{hs} as the frequency conflict degree. The frequency conflict degree will be used to calculate the earliest available time of resources required by a task. A task meets the frequency harmonics constraint if and only if every instance (job) of the task can be scheduled under the frequency harmonics constraint.

Definition 4.3. Semantic Importance: Higher threat level tasks have higher semantic importance. The semantic importance of an instance (job) of a task is defined by the task.

The notation used in the rest of the paper is:
c_i: worst case execution time of task T_i
d_i: deadline of task T_i
r_i: release time of task T_i
p_i: period of task T_i
f_i: transmitting frequency of task T_i
si_i: semantic importance of task T_i
pol_i: the type (polarizaton) of beams that task T_i requires
jt_i: the number of special filters required by task T_i
n_i: the number of beams required by task T_i
R_{ij}: the resources required by task T_i which is tuple of (pol_i, jt_i, n_i)

Each radar task may require multiple beams to work together with specified semantic importance, frequency, worst case execution, deadline, period, and resources required. Consequently, a task is defined as a nine tuple TASK ($c_i,d_i,r_i,p_i,f_i,si_i,pol_i,jt_i,n_i$). In this new task model, all of the constraints we addressed above are captured.

In our dual face phased array radar system, a radar task only specifies what type of beams it requires. It does not specify which particular beams to use. In fact, a radar task can be assigned to any beams that have the same type as the task requires. And the frequency of task T_i is also assigned to the beams to allocate this task during the task T_i's execution time.

We develop our heuristic algorithm presented in the following section based on the task model proposed here.

5. HEURISTIC ALGORITHM

5.1 Heuristic Function Definition

In a manner similar to [23] our solution is to define a heuristic function H. But our heuristic function defined here extends [23] in two dimensions: semantic importance and frequency harmonics.

As discussed above, each task can have multiple instances (jobs). The release time of each instance is defined by the period of the task. We use the H function to create a system-wide schedule by choosing the next job to be scheduled based on the output of the H function. If it is feasible to schedule then we set that job in the schedule and update the current resource allocations required by that job. We then iterate until all jobs are scheduled or until the remaining jobs cannot be scheduled. A task is schedulable if all of its jobs are schedulable.

The function H is defined as follows:

(3): $H(J_{i,j}) = \alpha \cdot D_{i,j} + \beta \cdot si_{i,j} + \gamma \cdot EST_{i,j}$

where $J_{i,j}$ is j-th instance (job) of task T_i, $D_{i,j}$ is the deadline of job $J_{i,j}$. $D_{i,j} = r_i + d_i + p_i \times j$, while r_i, d_i, p_i are as defined in section 4. $si_{i,j}$ is the semantic importance of job $J_{i,j}$ and it equals si_i as in section 4. The parameters α,β,γ are tunable parameters which can be adjusted to generate good performance. $EST_{i,j}$ is the earliest start time of job $J_{i,j}$ after it is available. $EST_{i,j}$ is defined as equation 4.

(4): $EST_{i,j} = MAX(EAT(R_{ij}))$.

where $EAT\ (R_{ij})$ is the earliest available time for the set of resources required by job $J_{i,j}$ subject to the frequency harmonics constraint defined by the conflict degree f_{hs} given in definition 4.2.

One reason to choose the H function as a linear function of deadline, semantic importance and earliest start time of a job is to keep the heuristic function computationally simple to make it more suitable for online scheduling.

By taking the frequency harmonics constraint into consideration, we provide two ways to calculate the earliest available time for the set of resources a job requires.

First we introduce how to decide whether a job J_i has any frequency conflict with jobs already scheduled or not. Assume jobs $J_j, J_{j+1}, \ldots, J_{j+k}$ have been scheduled when trying to schedule job J_i. Job J_i has no frequency conflict with jobs $J_j, J_{j+1}, \ldots, J_{j+k}$ if and only if $|f_i - f_p| > f_{hs}\ (p = j, \ldots, j+k)$, while f_{hs} is defined in the above section.

Case 1: Job J_i does not have any frequency conflict with the jobs already scheduled to execute during the same period of time.

Then the earliest available time of resources or beams a job requires is decided by the minimum values of the set of the same type of resources. Using 4 S-band beams as the example, let ∅{3, 5, 10, 8} be the set of the earliest available times of these 4 S-band beams. If job J_1 requires two S-band beams, then the earliest available time of the resources job J_1 is a subset with minimum values {3, 5}. The earliest start time of job J_1 is the maximum of these subsets. In this example, it is 5. And if job J_1 is scheduled, beams with earliest available times 3 and 5 will be assigned to job J_1.

Case 2: Job J_i has any frequency conflict with jobs already scheduled to execute during the same period of time.

Then the earliest available time of resources or beams a job requires is not only decided by the minimum values of the set of the same type of resources, but also by the resources (beams assigned to a job will also be assigned the frequency of that job) that have frequency conflicts with job J_i. We give an example in Figure 3 to explain how to calculate the earliest available time of the resources job J_i requires. In this example we assume the frequency conflict degree f_{hs} equals 0.

In Figure 3, we have six jobs and 3 S-band beams as the example jobs and resources, respectively. Jobs J_1, and J_2 have been scheduled to beam 1. Job J_3 has been scheduled to beam 2. And jobs J_4 and J_5 are allocated to beam 3. Job J_6 requires two beams. If there is no frequency conflict, as in case 1, then the earliest available time of resources job J_6 requires is decided by beam 2 and beam 3 as showed by the gray arrow pointing from job J_6 to these two beams. However, because job J_6 has a frequency conflict with job J_2 which is scheduled on beam 1, the earliest available time of resources needed by job J_6 is decided by all three beams as the three lines pointing from job J_6 in Figure 3.

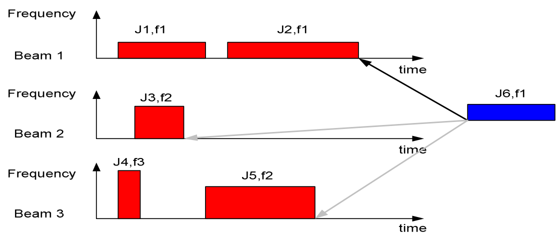

Figure 3: Earliest Available Time Example

The benefit of handling frequency conflicts at the same time as computing the earliest available time of resources a job requires is twofold:

1) Our heuristic algorithm remains a linear time complexity algorithm. The algorithm only requires a frequency conflict check while calculating the resources' earliest available time.
2) When a job has any frequency conflict with jobs already scheduled, it has no impact on the jobs already scheduled.

From the definition in equation (3), we can see that the following is true:

1) $D_{i,j}$ takes the time requirements of a job into account,
2) $si_{i,j}$ takes the semantic importance of a job into consideration, and
3) $EST_{i,j}$ encodes the resource requirements of a job and the frequency harmonic constraints.

We conclude that our heuristic function encapsulates the most significant characteristics of the radar scheduling problem we address in this paper.

5.2 Overview of the Heuristic Function Definition

As Figure 4 demonstrates, the scheduler determines a full feasible schedule for a given set of tasks in the following steps: (1) it generates all the instances (jobs) of each task available, and all the instances of each task available form a job set; (2) it starts with an empty schedule and tries to extend it one job at a time.

At each level, we first calculate the earliest available time for the resources each job requires. At this step, we take the frequency harmonics constraint into consideration. Then we calculate the earliest start time for each job. After that we compute the heuristic value for each task using equation 3. The scheduler tries to schedule a job with the minimum heuristic value in the job set remaining to be scheduled. At any step, if the minimum heuristic value holder cannot be scheduled, it is discarded from the job set. A task is not schedulable if any instance (job) of this task is not schedulable.

```
TASKSET = {T1, T2, … , Tk};
JOBSET =  ;
SCHEDULE =  ;
While(TASKSET    ){
        Calculate all jobs {J1,j, J2,j, … , Ji,j} of each task Tj in TASKSET;
        JOBSET = JOBSET    all the jobs of task Tj;
}
While (JOBSET     )
Do{
        For each job Ji,j in JOBSET
                Calculate Earliest Available Time of each resources required by job Ji,j;
                Calculate Earliest Start Time of job Ji,j;
                Calculate H value of job Ji,j;
        Choose the job Ji,j with minimum H value;
        IF (FEASIBLE(Ji,j)) THEN
                add job Ji,j to SCHEDULE;
                update Earliest Available Time of each resources allocated to job Ji,j;
                remove job Ji,j from JOBSET;
        ELSE
                remove job Ji,j from JOBSET;
                remove all jobs of task Tj from TASKSET;
                remove all jobs of task Tj in SCHEDULE;
                release all resources allocated to all jobs of task Tj;
}
```

Figure 4: Pseudo code for basic heuristic algorithm

5.3 Complexity

In the heuristic search space, the partial schedule of a task set constructs a search tree as shown in Figure 5. The left side of the Figure 5 shows the whole search space which is of exponential complexity. An exhaustive search can find an optimal schedule for a task set if one exists. This is not suitable for our dual face phased array radar scheduling problem to have online schedulability analysis. For this purpose, we only choose one of the vertices at each level of the search tree to expand the schedule to the next level. This enables our algorithm to execute in polynomial time.

At each level of the search tree, an instance (job) of a task which has the minimum heuristic value is chosen to be scheduled. This minimum heuristic value is calculated by using equation 3. The right hand side of Figure 5 shows our heuristic algorithm search path by the gray nodes in the tree. For a set of periodic tasks $\{T_1, T_2, \ldots, T_k\}$, each task T_i (i=1,…,k) may have multiple instances. Assume the scheduling period P equals the least common multiple of all tasks' periods in the task set. Then the number of instances m_i that a task T_i can have during the scheduling period P is P divided by p_i, where p_i is the period of task T_i. Assume $m = \sum_{i=1}^{k} m_i$,

then our heuristic algorithm's complexity is $m \log(m)$. Let the minimum number of m_i (i=1,...,k) be $min(m_i)$ and the maximum number of m_i (i=1,…,k) be $max(m_i)$. We know that both $min(m_i)$ and $max(m_i)$ are finite constants. So equation (5) must be true where n is the number of tasks to be scheduled.

(5): $n \times \min(m_i) \le m \le n \times \max(m_i)$.

From equation (5), we can infer that this algorithm executes in polynomial time.

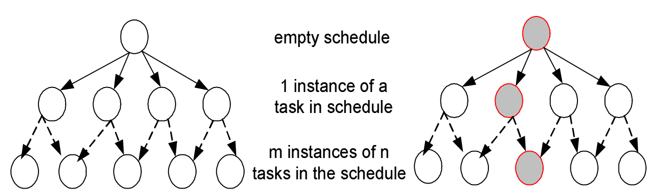

Figure 5: Heuristic Searching Space

6. PERFORMANCE EVALUATIONS

We evaluate our heuristic function through simulations. The resources modeled in the simulation match the ship-based radar system mentioned above, i.e., 16 beams are S-band vertical polarization beams, 16 L-band dual polarization beams, and 32 L-band vertical polarization beams with 2 of these beams having special filters. Each task, either a searching or tracking task, may require multiple beams simultaneously. Every task is non-preemptive and is subject to the frequency harmonics constraint defined in previous sections.

It is quite difficult to compare our solution to other known solutions. No solutions in the literature (as far as we know) address this same scheduling problem. Further, many current solutions are based on a set of ad hoc scheduling rules which are not available for comparison. Solutions using Q-RAM might be appropriate, but a development of a solution using Q-RAM is itself a research problem. The literature [1] does propose using a static priority scheduling algorithm where the priority is set according to semantic importance (we refer to this algorithm as IMP). Hence, we compare to this simple approach. Since using importance alone does not consider deadlines we also compare to EDF. The robust EDF (RED) [3] is a possible comparison to our heuristic algorithm since RED also considers task's value when rejecting a task. However, RED needs an efficient online load calculator to detect the overload conditions. And RED is designed for sporadic tasks. In this paper, we are considering periodic tasks.

We evaluate our algorithm using the total tasks' success ratio and the highest semantic importance tasks' success ratio. The total tasks' success ratio for a task set defines the total number of successfully scheduled tasks divided by the total number of tasks in the task set. The highest semantic importance tasks' success ratio for a task set defines the total number of successfully scheduled highest semantic importance tasks divided by the total number of highest semantic importance tasks in the task set.

We use simulations to (i) set parameters α,β,γ in the heuristic function formula, (ii) evaluate static scheduling and (iii) evaluate online/dynamic scheduling. The simulation results demonstrate that our heuristic scheduling algorithm has the best performance over all the simulations.

6.1 Static Scheduling

The workload of some real-time systems can be considered as the processor utilization factor. For a set of periodic tasks with only CPU as a resource constraint, the utilization factor [16] can be computed by $U = \sum_{i=1}^{n} \frac{c_i}{p_i}$, where c_i, p_i are as defined in section 4. For the radar transmitting tasks that we consider in this paper, each task requires multiple resources such as CPU, beams and each task is also subject to frequency harmonics and semantic importance constraints. Because of these multiple constraints, it is improper to compute workloads by a simple formula. We illustrate the difficulty to quantitatively measure our system workloads using any single resource's utilization in the example below. To simplify the example, we use only CPU, beam 1, and beam 2 as the resources (note we have a total of 64 beams in the system) and task sets S1 and S2 as the example task sets. The utilizations of S1 and S2 are listed in Table 1. Let's look at the utilization of S1 first, if we use CPU utilization to measure the workload, we think it is a light workload, but it is not true because of the heavy load at beam 2. To make things more complicated, it is very difficult to measure a workload that has frequency harmonic constraints. For task set S2, if using any single resource utilization, we can falsely think it is light workload. But it possible because a task can not be scheduled on beam 1 due to a frequency conflict. So we define the light, medium and heavy workloads below for this paper by determining the combination of utilizations on multiple resources through simulations.

Table 1: Resource Utilization

	CPU	Beam 1	Beam 2
Utilization of S1	0.2	0.3	0.9
Utilization of S2	0.2	0.1	0.4

Workload Generator: We evaluate static scheduling based on three types of workloads namely light, medium, and heavy workloads. As discussed in previous sections our task model captures multiple dimensional constraints such as timeliness and resource constraints. We define the workloads from light to heavy by adjusting the worst case execution time (timeliness property) and the number of beams (resource requirements) a task requires. In this paper, light workloads differ from medium and heavy workloads in worst case execution time. And heavy workloads differ from light and medium workloads in the number of beams a task requires. The workload generator creates a task set based on a set of task parameters. A task is represented as a nine tuple TASK ($c_i,d_i,r_i,p_i,f_i,si_i,pol_i,jt_i,n_i$). Here, we try to mimic the real radar tasks' requirements. But we do not have the well-known task specifications to follow. The worst case execution time c_i is uniformly distributed within the range [1, 5] for light workloads, and [1, 10] for medium and heavy workloads. The relative deadline d_i is equal to period p_i. Each task is randomly assigned a period p_i from the set {10, 20, 50, 100}. The release time r_i is set to 0 for the static task sets for simplicity. For dynamic scheduling, the release times of tasks are defined by a Poisson distribution as disscussed in the online scheduling section. The frequency f_i is uniformly distributed within the range [950MHz, 1000MHz] for S-band tasks, and [1000MHz, 5000MHz] for L-band tasks. The semantic importance si_i is uniformly distributed between 1 and 10. So statistically, we have 10% of the tasks as highest semantic importance tasks in a task set. The smaller value has higher semantic importance. The polarization pol_i is uniformly distributed within the range [1, 3] in which value 1 means a task requires vertical polarization beams, value 2 requires dual polarization beams and value 3 requires all polarization beams. The number of special filters jt_i is uniformly distributed within the range [0, 2]. The number of beams a task requires is uniformly distributed between [1, 3] for light and medium workloads. The number of beams a task requires is uniformly distributed within the range [1, 5] for heavy workloads. For each light, medium and heavy workload, we generate 10 sets of tasks and each task set has 200 tasks. Each task may have multiple instances. Each instance of a task differs in its release time according to the task's period. The scheduling period P is 100.

6.1.1 Tuning the Parameters of H

From the definition of the heuristic function, we see that the tunable coefficients α, β, γ are very important to the performance of the algorithm. We determine the values of α, β, γ through simulations.

To find the good α,β,γ values, we first generate a set of 200 tasks for each light, medium and heavy workload. We do not consider the frequency conflict while tuning the parameters. We run our heuristic algorithm on each task set by varying each parameter α, β, γ from 0.1 to 1.0 using steps of 0.2. This exhaustive search of α,β,γ produces good values. The (α,β,γ) values generating the highest semantic importance tasks' success ratio for each workload is chosen for the following performance evaluations. We use (0.7,0.1,0.5), (0.5,0.3,09) and (0.9,0.3,0.7) for light, medium, and heavy workloads, respectively. Figure 6 and Figure 7 give the relationship between α,β,γ values and the total tasks' success ratio, and the highest semantic importance tasks' success ratio at medium workloads while setting α to 0.5. We can see while fixing the α value, the variations of the highest semantic importance tasks' success ratio with the change of β and γ values are within 5%. Similar results exist for the total tasks' success ratio. Figure 8 and Figure 9 give similar results while setting beta to 0.3. This shows that the values chosen are not too sensitive.

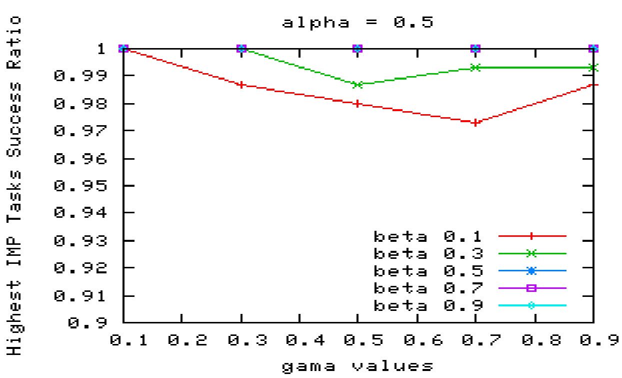

Figure 6: αβγ values

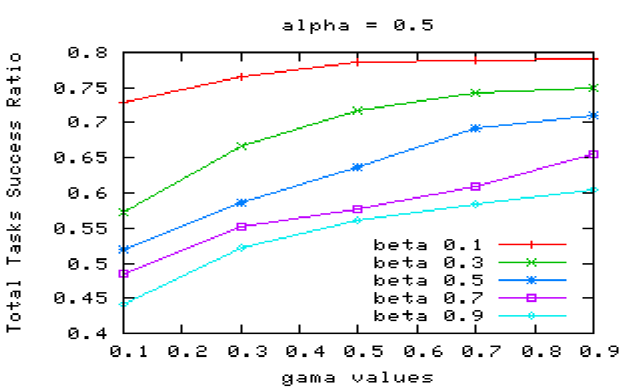

Figure 7: αβγ values

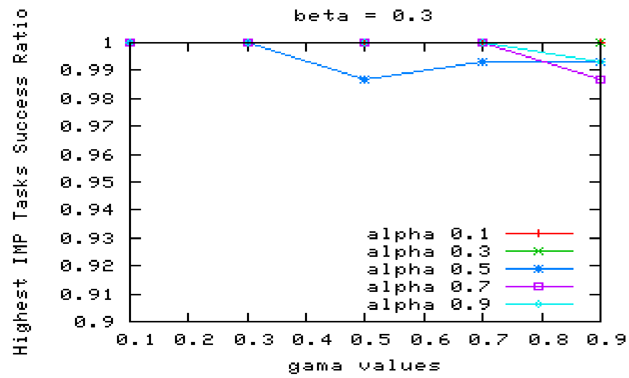

Figure 8: αβγ values

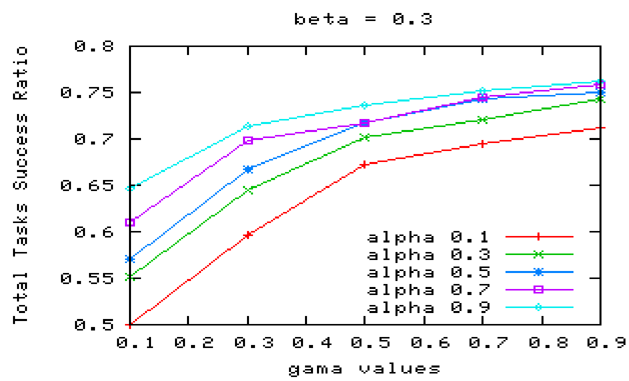

Figure 9: αβγ values

6.1.2 Simulation Results

6.1.2.1 Vary Workloads

In this subsection, we evaluate our heuristic algorithm against EDF and the static priority scheduling algorithm IMP at light, medium and heavy workloads. All the figures are plotted with 95% confidence intervals as the error bar.

In Figure 10 and Figure 11, the frequency conflict degree f_{hs} is set to 3. In the following section, we discuss how the frequency conflict degree affects performance. Figure 10 gives the performance of all three algorithms with respect to the highest semantic importance tasks' success ratio. From Figure 10, we can see that our heuristic algorithm achieves as high as 99%, 94%, and 91% highest semantic importance tasks' success ratios at light, medium and heavy workloads, respectively. EDF results in 72%, 60%, and 41% success ratio at light, medium and heavy workloads, respectively. IMP has 80%, 66%, and 60% success ratio at light, medium and heavy workloads, respectively. These results indicate that our heuristic algorithm is 50% better than EDF and 31% better than IMP at heavy workloads. At medium workloads, our heuristic algorithm is 34% better than EDF, 28% better than IMP. Even at light workloads, our heuristic algorithm performs 27% better than EDF and 19% better than IMP. Figure 11 shows the performance on total tasks' success ratio. From Figure 11, we know that our heuristic algorithm is 10% better than EDF and 45% better than IMP even at heavy workloads in terms of total tasks' success ratio. At light workloads, our heuristic algorithm is over 70% better than IMP and 20% better than EDF.

EDF gives better performance than IMP in terms of the total tasks' success ratio on all three workloads, but it gives worse performance than IMP on highest semantic importance tasks' success ratio. In all three workloads, our heuristic algorithm gives best performance in terms of both highest semantic importance tasks' success ratio and total tasks' success ratio. Non-preempitve EDF is proven to be optimal if no idle times can be inserted in the schedule for a single processor without overload as in [11]. However, the only resource constraint considered in [11] is the CPU resource. Our system has multiple resource constraints such as CPU, 64 beams and frequency harmonic constraints. Our heuristic algorithm by taking the multiple constraints into consideration results in better performance.

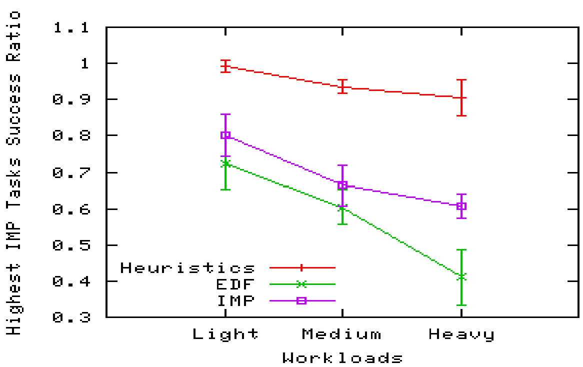

Figure 10: Heuristic Algorithm

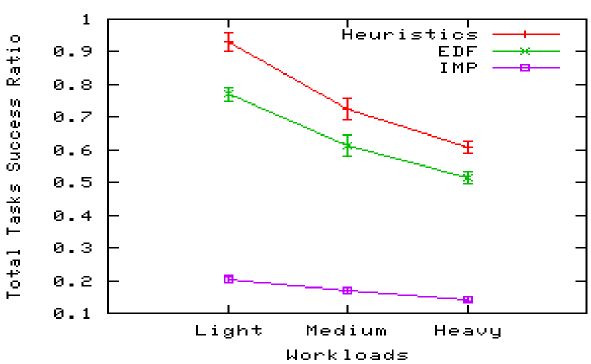

Figure 11: Heuristic Algorithm

6.1.2.2 Vary Frequency Constraints

In this section, we discuss the effect of the frequency harmonics constraint on our heuristic algorithm. The frequency conflict degree f_{hs} is as defined in section 4. We evaluate our heuristic algorithm, EDF and IMP at frequency conflict degree f_{hs} as 0, 1, 3, 5, and 7 on light, medium and heavy workloads. We only present the heavy workloads as the study case, but similar results exist for light and medium workloads. In Figure 12, we plot the highest semantic importance tasks' success ratio against the frequency conflict degree. Figure 13 shows the relationship between the total tasks' success ratio and the frequency conflict degree.

From Figures 12 and 13, we can see that our heuristic algorithm is very robust to the increase of the frequency conflict degree for both success ratios. IMP saturates on both the highest semantic importance tasks' success ratio and the total tasks' success ratio so that it does not change dynamically with the frequency conflict degree. EDF drops both the highest semantic importance tasks' success ratio and the total tasks' success ratio rapidly when the frequency conflict degree increases. Our heuristic algorithm, explicitly addressing frequency harmonics constraints via the heuristic function, gives excellent performance.

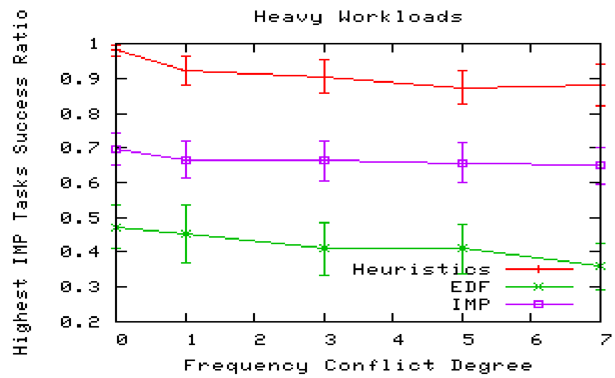

Figure 12: Frequency constraint effect

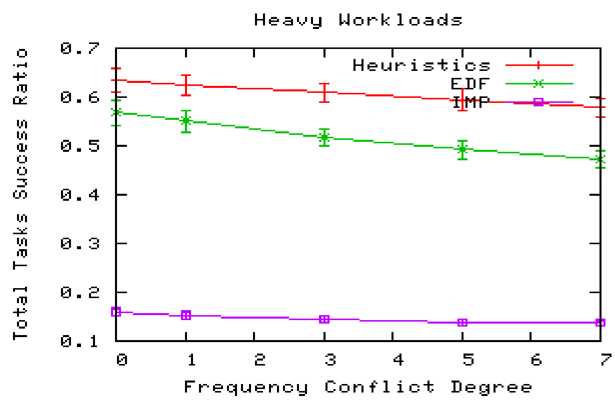

Figure 13: Frequency constraint effect

6.2 Online Scheduling

In this section, we discuss the overhead and performance of our online scheduling heuristic algorithm.

6.2.1 Online Scheduling Algorithm

The basic idea of the online scheduling algorithm is that whenever a new task arrives, the scheduling algorithm reschedules all the tasks which currently are not in execution plus the new arrivals.

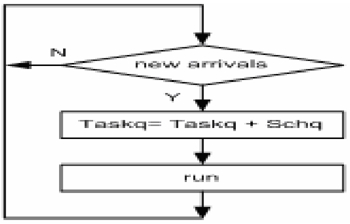

Figure 14: Online Scheduling Control Flow

Figure 14 gives the control flow of the online scheduling algorithm. The Taskq in Figure 14 is the queue to hold the tasks to be scheduled and the new arrivals. The Schq is the queue to hold the tasks already scheduled, but waiting to be executed.

6.2.2 Complexity

By comparing with the static scheduling algorithm, the overhead of the online scheduling comes from the fact that all the tasks in the schedule queue waiting to be executed are to be rescheduled whenever there are new arrivals. Assume tasks arrive according to a Poisson distribution with parameter λ. Then the tasks' inter-arrival time is exponentially distributed with parameter 1/λ. The maximum number of times a task T_i rescheduled is equal to its deadline d_i divided by 1/λ. A task will be deleted from the task queue if it missed its deadline. We assume that the total scheduling period is P. The total number of tasks S arriving during period P is defined by $S = P \times \lambda$. So we can equate the online scheduling to be the static scheduling algorithm with total number of tasks n satisfying equation (6), and (7):

(6): $S \times (1 + \min(d_i /(1/\lambda_i))) \le n$

(7): $n \le S \times (1 + \max(d_i /(1/\lambda_i)))$

As we can see the algorithm still executes in polynomial time.

6.2.3 Simulation Results

We also evaluate our online scheduling algorithm on three types of workloads. Similar to the static scheduling algorithm, each task is represented by a nine tuple TASK $(c_i,d_i,r_i,p_i,f_i,si_i,pol_i,jt_i,n_i)$. Different from the static scheduling workloads generator, the distribution of release times of tasks is the major parameter to vary from light to heavy workloads. The worst case execution time c_i is uniformly distributed in the range [1, 3] for all three workloads. The deadline d_i is equal to period p_i. The period p_i is uniformly randomly chosen from the set {10, 20, 50, 100}. The release time is calculated based on the Poisson distribution with parameter λ, the arrival rate. The parameter λ equals 5 for light workloads, 10 for medium workloads, and 15 for heavy workloads. The frequency f_i is uniformly distributed within the range [950MHz, 1000MHz] for S-band tasks, and [1000MHz, 5000MHz] for L-band tasks. The semantic importance si_i is uniformly distributed between 1 and 10. The polarization pol_i is uniformly distributed within the range [1, 3]. The number of special filters jt_i is uniformly distributed within the range [0, 2]. The number of beams n_i is uniformly distributed in [1, 3] for all three workloads. We use (0.5,0.1,0.7) as the α,β,γ values for the heuristic algorithm for all three workloads. The parameters α,β,γ are tuned through simulations similar to static scheduling. The scheduling period P is 100. For all three workloads, we set the frequency conflict degree f_{hs} to 3. Each task may have multiple instances during the scheduling period.

As Figure 15 shows, our heuristic algorithm can achieve 97%, 93% and 85% in terms of the highest semantic importance tasks' success ratio at light, medium and heavy workloads, respectively. At heavy workloads, our heuristic algorithm is 30% better than EDF and 20% better than IMP in terms of the highest semantic importance tasks' success ratio. As shown in Figure 16, our heuristic algorithm is 10% better than EDF and 20% better than IMP on the total tasks' success ratio for all three workloads. For online scheduling, the results are similar to the static scheduling algorithm for varying frequency conflict degrees. Due to space limits, we do not present the figures here.

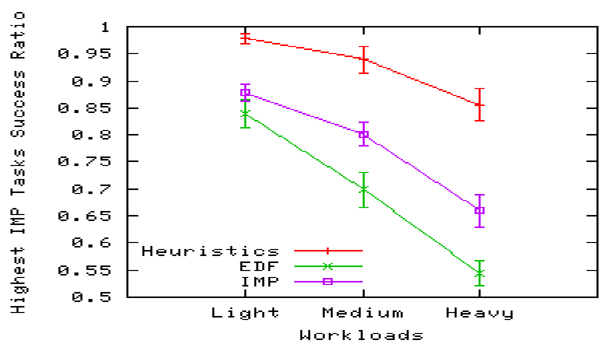

Figure 15: Online Scheduling

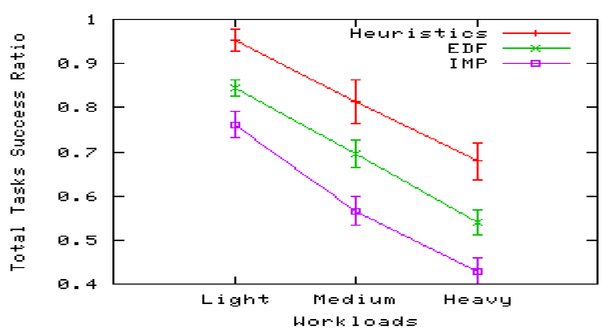

Figure 16: Online Scheduling

7. CONCLUSIONS

Many real-time applications such as dual face phased array radar systems are complex and scheduling solutions based on rate monotonic or EDF are not adequate. To handle the realistic and multi-dimensional constraints of this application a heuristic function solution was developed and evaluated. The heuristic algorithm in this paper takes the timeliness, resource requirements, semantic importance and physical constraints into consideration. It is a very flexible algorithm that incorporates many aspects of a dual face phased array radar task. Based on the simulation results, we show that our heuristic algorithm provides excellent performance. We also discuss the overhead and performance of the use of the algorithm on-line.

8. ACKNOWLEDGEMENTS

This work was supported in part by the MURI award N00014-01-1-0576 from ONR and DARPA.

9. REFERENCES

[1] R. Baugh. Computer Control of Modern Radars. New York: RCA Corporation, 1973.

[2] P. Bratley, M. Florian, and P. Robillard. Scheduling with Earliest Start and Due Date Constraints, Naval Research Quartery, 18(4), 1971.

[3] G. C. Buttazzo and John A. Stankovic. RED: Robust Earliest Deadline Scheduling*, Proceedings of The Third International Workshop on Responsive Computing Systems*, Austin, 1993.

[4] M. L. Dertouzos. Control Robotics: The Procedural Control of Physical Processes, *Information Processing*, 74, 1974.

[5] K. Efe. Heuristic Models for Task Assignment Scheduling in Distributed Systems, *IEEE Computer*, June 1982.

[6] N. Fourikis, Phased Array-Based Systems and Applications, *John Wiley & Sons, Inc*. ISBN 0-471-01212-2, 1996.

[7] S. Ghosh, Ragunathan Rajkumar, Jeffery Hansen and John Lehoczky. Scalable Resource Allocation for Multi-Processor QoS Optimization, *In Proceedings of the 8th International Conference on Distributed Computing Systems (ICDCS 2003),* May. 2003.

[8] S. Ghosh, Ragunnathan Rajkumar, Jeffery Hansen, and John Lehoczky. Integrated Resource Management and Scheduling with Multi-Resource Constraints, *IEEE Real Time Systems Symposium*, 2004.

[9] K. Gopalan, and T. Chiueh. Multi-resource Allocation and Scheduling for Periodic Soft Real-Time Applications. *In Proc. of ACM/SPIE Multimedia Computing and Networking (MMCN2002)*, San Jose, CA, Jan. 2002.

[10] S. Gopalakrishman, M. Caccamo, C. S. Shih, C. G. Lee, and L. Sha, Finite-Horizon Scheduling of Radar Dwells with Online Template Contruction, *IEEE Real Time Systems Symposium*, 2004.

[11] K. Jeffay, D. F. Stanat, and C. U.Martel. On Non-preemptive Scheduling of Periodic and Sporadic Tasks with Varying Execution Priority, *In Proceedings of the IEEE Real-Time Systems Symposium*, pages 129-139, Dec. 1991.

[12] H. H. Johnson and M. S. Madison. Deadline Scheduling for a Real-Time Multiprocessor, NTIS (N76-15843), Springfield, VA, May, 1974.

[13] H. Kise, T. Ibaraki, and H. Mine. A Solvable Case of the One Machine Scheduling Problem with Ready and Due Times, *Operations Research*, 26(1):121-126, 1978.

[14] T. W. Kuo, Y. S. Chao, C. F. Kuo, C. Chang, and Y. L. Su. Real-Time Dwell Scheduling of Component Oriented Phased Array Radars. *In Proceedings of IEEE Real Time Systems Symposium,* Dec. 1992.

[15] C. G. Lee, P. S. Kang, C. S. Shih, L. Sha. Radar Dwell Scheduling Considering Physical Characteristics of Phased Array Antenna, *IEEE Real Time Systems Symposium,* 2003.

[16] C. L. Liu and J. W. Layland. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment, *Journal of the ACM*, 20(1), 1973.

[17] R. P. Ma, Y. S. Lee, and M. Tsuchiya. A Task Allocation Model for Distributed Computing Systems, *IEEE Trans. Computers*, Vol. C-31, Jan. 1982.

[18] R. R. Muntz, and E. G. Coffman. Preemptive Scheduling of Real-Time Tasks on Multiprocessor Systems, *Journal of ACM*, Vol. 17, Apr. 1970.

[19] R. Rajkumar, C. Lee, J. P. Lehoczky and D. P. Siewiorek. Practical Solution for QoS-based Resource Allocation Problems, *IEEE Real Time Systems Symposium*, pages 296-306, 1998.

[20] R. Rajkumar, C. Lee, J. P. Lehoczky and D. P. Siewiorek. A Resource Allocation Model for QoS Management, *IEEE Real Time Systems Symposium*, pages 298-307, 1997.

[21] C. S. Shih, S. Gopalakrishnan, P. Ganti, M. Caccamo, and L. Sha. Template-based Real-Time Dwell Scheduling with Energy Constraints, *IEEE Real-Time and Embedded Technology and Applications Symposium*, 2003

[22] J. Stankovic, and K. Ramamritham. The Design of the Spring Kernel, *In Proceedings of the IEEE Real Time Symposium*, Dec. 1987.

[23] W. Zhao, K. Ramamritham, and J. A. Stankovic. Scheduling Tasks with Resource Requirements in Hard Real-Time Systems, *IEEE Transactions on Software Engineering*, Vol. SE-13, No.5, May 1987.