Real-Time Syst (2006) 33: 47–75
DOI 10.1007/s11241-006-6882-z

Finite-horizon scheduling of radar dwells with online template construction

Sathish Gopalakrishnan · Marco Caccamo · Chi-Sheng Shih · Chang-Gun Lee · Lui Sha

Abstract Timing constraints for radar tasks are usually specified in terms of the minimum and maximum temporal distance between successive radar dwells. We utilize the idea of feasible intervals for dealing with the temporal distance constraints. In order to increase the freedom that the scheduler can offer a high-level resource manager, we introduce a technique for nesting and interleaving dwells online while accounting for the energy constraint that radar systems need to satisfy. Further, in radar systems, the task set changes frequently and we advocate the use of finite horizon scheduling in order to avoid the pessimism inherent in schedulers that assume a task will execute forever. The combination of feasible intervals and online dwell packing allows modular schedule updates whereby portions of a schedule can be altered without affecting the entire schedule, hence reducing the complexity of the scheduler. Through extensive simulations we validate our claims of providing greater scheduling flexibility without compromising on performance when compared with earlier work based on templates constructed offline. We also evaluate the impact of two parameters in our scheduling approach: the template length (or the extent of dwell nesting and interleaving) and the length of the finite horizon.

Keywords Radar dwell scheduling · Real-time scheduling · Energy constraints · Finite horizon

1. Introduction

Multi-function phased array radar systems search and track targets in a specified surveillance region. In order to maintain tracks, the system must revisit the targets within some time

S. Gopalakrishnan · M. Caccamo · L. Sha
Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, USA

C.-S. Shih
Department of Computer Science and Information Engineering, Graduate Institute of Networking and Multimedia, National Taiwan University, Taipei, Taiwan

C.-G. Lee
Department of Electrical and Computer Engineering, The Ohio State University, Columbus, OH, USA

frame. This timing constraint ensures that the radar will illuminate the targets with high probability. On the other hand, the tracks must be scheduled in a manner that does not violate the energy constraint, i.e., the schedule must not lead to overheating of the radar's components.

A (radar) track task is an end-to-end task (Bettati, 1994; Sun, 1997) made up of three subtasks: a control command subtask, a dwell subtask, and a signal processing subtask (Shih et al., 2003b; Baugh, 1973). At the start of a track task, the control command subtask generates the commands for sending and receiving the electromagnetic waves. Based on the generated commands, the dwell subtask sends electromagnetic waves (EMWs) at scheduled instants in time and receives the echoes (reflected EMWs) during an expected receive interval.[1] The received signals are passed to the signal processing subtask that estimates the location of the tracked target. The location estimate is used by the control command subtask to decide the next set of dwell commands and in this fashion the execution sequence of the three subtasks repeats. The track task stops when there is no further need to monitor the target.

Dwell execution is usually the performance bottleneck in a radar system because of the tight constraints (temporal and physical constraints) within which the dwells need to be scheduled. On the other hand, the performance of the signal processing algorithms and control planning can be improved by adding more processors at the back-end. The enormous cost of a radar antenna makes it prohibitive to add extra array elements and hence it is vital that the dwell scheduler maximizes antenna utilization. For this reason, the work presented in this paper focuses on radar antenna scheduling. We introduce novel scheduling techniques and improve radar performance compared with prior work (Shih et al., 2003 a, b, 2004) while substantially reducing the scheduling overhead.

When scheduling dwells, the system must maintain a minimal and maximal temporal distance between any two consecutive dwells of a track task. This requirement can easily be understood by means of an illustration (Fig. 1). The figure shows the trajectory of a tracked target, and the time line is used to indicate time instants at which dwells (for this target) start or finish. Assume that dwell n completes at time a. The *processing window* $(a, b]$ is the time interval during which the signal and control command subtasks execute. Only when the control command subtask completes does the system know the antenna command for dwell $(n + 1)$. Dwell $(n + 1)$, therefore, cannot be scheduled in interval $(a, b]$.[2] The difference $b - a$ is the *minimal temporal distance* between the dwells. In the figure, the coverage of the waves and target location estimates are shown as the shaded and dashed circles, respectively. The uncertainty in target velocity increases with time; consequently, the error in the estimate of the target location also increases. If dwell $(n + 1)$ is completed within time interval $(b, d]$, the target can be found and the error in the estimated location is acceptable. However, if dwell $(n + 1)$ is sent after time d, the beam may miss the target with high probability. The difference $d - a$ is called the *maximal temporal distance* between dwell n and dwell $(n + 1)$ and interval $(b, d]$ is the *illumination window* of dwell $(n + 1)$.

Traditionally, the system either disregards the actual timing constraints (e.g., the best effort scheduling algorithms (Baugh, 1973)) or uses the most stringent values to ensure safety (e.g., using the half length of the maximal temporal distance as the period of periodic tasks.) If dwells of a track task are not executed inside the appropriate time intervals

[1] The start time for the receive interval depends on the estimated distance between the radar and the target being tracked. The duration of the interval depends on the duration of the transmission. The duration of the receive interval can be adjusted to improve the accuracy of target tracking.

[2] Notice that this is equivalent to a deadline decomposition for back-end processing and dwell execution within the end-to-end radar task.

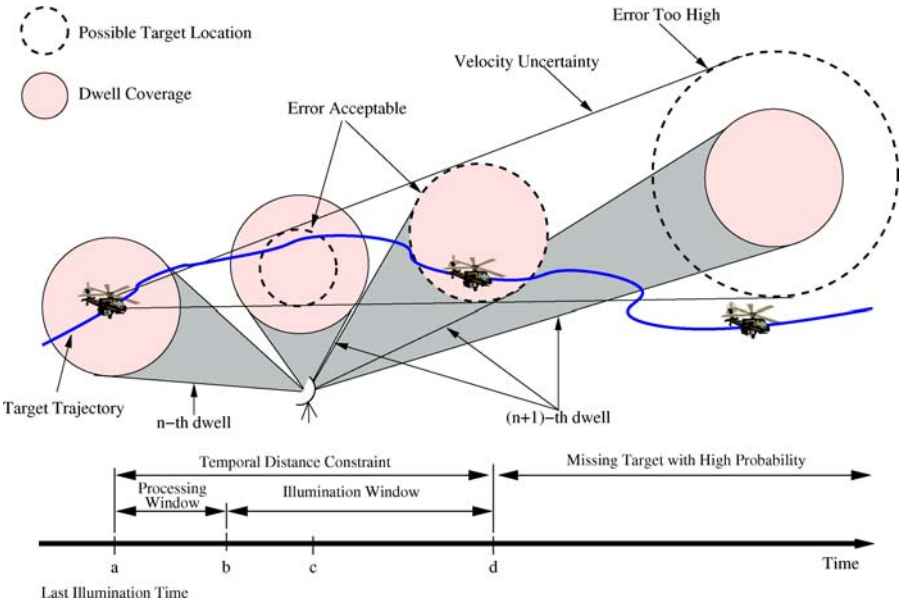


Fig. 1 Temporal distance constraint

then the result is unpredictable timing behavior and/or poor resource utilization (Fig. 1). In addition, the "estimated" execution time of dwell $(n + 1)$ must be known before the control command subtask starts. Without this knowledge, the control command subtask cannot generate the appropriate control commands. In traditional dynamic real-time scheduling algorithms (such as the EDF algorithm), it is hard to predict the time instant at which a job will be scheduled before the job is released. In a radar system, this would require the control command subtask to make conservative estimates of a target's location and the error in the location estimate would be high. Such a requirement necessitates knowing the schedule, not just the schedulability.

Our work is related to research on temporal distance constraints (e.g., Han and Lin 1992; Hsueh et al. 1995; Han et al. 1996; Dong et al. 1998). In prior work, consecutive jobs of the same task need not have a minimum separation. Moreover, the nature of radar dwells (with non-preemptible send and receive phases) poses significant challenges to the schedulability analysis; these issues are not addressed by other research on scheduling with temporal distance constraints. Kuo et al. (2002) have proposed a reservation-based approach for real-time dwell scheduling. This approach allows the system to guarantee the performance requirement when the schedulability condition holds. On the other hand, Kuo et al. assume that the sampling periods of tasks are known and do not consider the energy constraint or the potential improvement in antenna utilization through nesting and interleaving of dwells.

Radar dwells are unlike jobs in traditional real-time systems because they have three phases—send, round-trip delay, and receive. Dwells are non-preemptible during the send and receive phases. Treating a dwell, in its entirety, as a non-preemptible job leads to poor resource utilization because it is possible to schedule other dwells during the round-trip delay phase of a dwell. Shih et al. (2003b) introduced the idea of template-based scheduling. A *template* is a fixed-length partial schedule, constructed by interleaving or nesting multiple dwells under the energy constraints. The overall schedule is a sequence of templates (Fig. 2). Templates contain more than one radar dwell. It is also evident (comparing A1 and A6 in Fig. 2) that nesting and interleaving can improve antenna utilization.

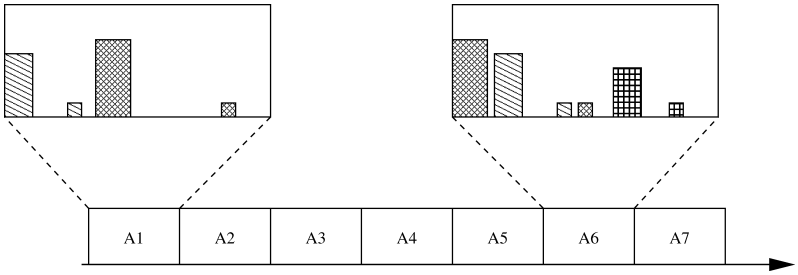


Fig. 2 Template-based schedule

While earlier work on template-based dwell scheduling (Shih et al., 2003 a, b, 2004) has addressed many of the issues in radar scheduling, they have some limitations. Firstly, the radar tasks with temporal distance constraints are conservatively modeled as tasks with harmonic periods (Shih et al., 2003b), resulting in significant under-utilization of resource. Further, the templates are constructed offline, assuming a small set of parameter combinations and a predetermined task set called the *performance requirement* (Shih et al., 2003 a, b, 2004). However, in a more general setting, where a resource manager dynamically assigns parameters to tasks online depending on target dynamics, it is not practical to construct all possible templates offline. Finally, the schedule is constructed for the hyperperiod[3] of the task set on the basis of the *permanent task model* which assumes that each task will be present in the system forever. In reality, the life time of a radar task is governed by the duration for which the related target is in the surveillance space, and the time between mode changes (a target might transition from high-precision to low-precision or vice versa, and such adaptations will alter the task parameters). To accommodate for the dynamics, resource allocations made using the permanent task model are overly conservative, thus making it an inappropriate model in a radar system.

In this paper, we make three main contributions to overcome the above drawbacks. First, we identify, offline, *feasible intervals* for dwells. The key idea behind feasible intervals is that as long as successive jobs of the same task are scheduled to start within their respective feasible intervals, irrespective of the exact start times, they are guaranteed to satisfy the temporal distance constraints. Feasible intervals are computed for each job based on the temporal distance constraints. Second, we propose online template construction, i.e., *online dwell packing*, to support both adaptive adjustment of the schedule and highly dynamic workloads. This way, our dwell scheduler can work with resource managers like Q-RAM (Hansen et al., 2004; Ghosh et al., 2004) that make fine-grained and dynamic adjustments to task parameters. Finally, we utilize the machinery we have developed and propose aggressive task admission using *finite horizon scheduling*. When the task set and associated parameters are subject to frequent change, scheduling for a hyperperiod is meaningless because the schedule might be invalidated long before the end of the hyperperiod. Finite horizon scheduling can reduce the pessimism of the permanent task model while guaranteeing timing and energy constraints for the horizon specified by the radar operator or resource manager. To streamline the design of a finite horizon scheduler, we introduce the idea of *modular schedule update* which provides a framework for simple and flexible scheduling.

The following section describes the task model and defines the terms used here. The section also states the problem of scheduling radar dwells. Section 3 presents the feasible

[3] The hyperperiod is the least common multiple of all task periods in a system.

interval concept and the associated period and deadline assignment algorithm. Section 4 details the online dwell packing technique. Section 5 describes finite horizon scheduling, and section 6 presents an evaluation of our approach. In Section 7, we discuss some of the related work in greater detail. Section 8 summarizes the paper.

2. Formal model and problem statement

Formal model. A *dwell pattern*, called a *dwell* for short and denoted by W, specifies the requirements for the transmission and reception (of electromagnetic waves) for a dwell task. A dwell is characterized by its execution time and power function. The execution of a dwell consists of three sequential phases: *sending*, *round-trip delay*, and *receiving phase*. We denote the time taken for each of these phases by $e_T(W)$, $e_{RT}(W)$ and $e_R(W)$ respectively. Dwells should not be preempted during the send and receive phases; such preemptions are equivalent to dwell failures. The radar system could be idle or execute other dwells during the round-trip delay phase of one dwell. The execution time of a dwell W, denoted by $e(W)$, is equal to $e_T(W) + e_{RT}(W) + e_R(W)$. The amount of power consumed by a dwell is described by a time function. A power function $P(W, t)$ of dwell W for $0 \leq t \leq e(W)$ represents the power consumed by the dwell during the course of its execution.[4] A dwell W is therefore characterized by the pair $(e(W), P(W, t))$.

Thus far, and in our subsequent discussion, we use the terms task and job as they are commonly used in real-time systems literature (Liu and Layland, 1973; Han and Lin, 1992; Sprunt et al., 1989). A *job* is a radar dwell and is an instance of a (dwell) task. A *task* is a sequence of jobs that carry out a particular function and have identical or similar characteristics and timing requirements. In a radar system, a task might track a target or search for targets. Tasks are denoted by T_1, T_2, etc. The j-th instance of task $\mathcal{T}_i$ is referred to as job $\mathcal{J}_{i,j}$. Unless stated otherwise, by task and job we mean a dwell subtask and a job of a dwell subtask, respectively.

Each task, $\mathcal{T}_i$, is associated with a dwell W_i. The timing parameters of a task $\mathcal{T}_i$ are its release time, execution time, minimal temporal distance and maximal temporal distance. The *release time* of task $\mathcal{T}_i$, denoted by r_i, is the instant of time at which the task becomes known to the system. The execution time, denoted by $e_i = e(W_i)$, is the time required to send and receive the waves.

Definition 2.1 (Illumination time). The *illumination time* of a job $\mathcal{J}_{i,j}$ is denoted by $t_{i,j}$ and is the instant of time at which the radar system starts sending the radar beams for the job.

The minimal temporal distance and maximal temporal distance of task $\mathcal{T}_i$, denoted by $\delta_{i,\min}$ and $\delta_{i,\max}$, are the lower and upper bounds for the temporal distance between its jobs.

Definition 2.2 (Minimum temporal distance constraint). If $t_{i,j}$ is the illumination time of dwell job $\mathcal{J}_{i,j}$ and $t_{i,j+1}$ is the illumination time of dwell job $\mathcal{J}_{i,j+1}$ then we must have $\delta_{i,\min} \geq t_{i,j+1} - t_{i,j}, \forall i,\ j \geq 0$, where $\delta_{i,\min}$ is the minimum temporal distance between two successive dwell jobs of the same task.

Definition 2.3 (Maximum temporal distance constraint). If $t_{i,j}$ is the illumination time of dwell job $\mathcal{J}_{i,j}$ and $t_{i,j+1}$ is the illumination time of dwell job $\mathcal{J}_{i,j+1}$ then we must have

[4] As can be expected, during the round-trip delay phase, a dwell consumes no power and the values of the power function are zero.

$\delta_{i,\max} \leq t_{i,j+1} - t_{i,j}, \forall i, j \geq 0$, where $\delta_{i,\max}$ is the maximum temporal distance between two successive dwell jobs of the same task.

An obvious requirement is that the minimum temporal distance be greater than the execution length of the dwell ($\delta_{i,\min} \geq e_i$) and that the maximum temporal distance is greater than the minimum temporal distance ($\delta_{i,\min} \geq \delta_{i,\max}$).

The *illumination window* of job $\mathcal{J}_{i,j}$ is a time interval which starts $\delta_{i,\min}$ units of time after the illumination time of its preceding job $\mathcal{J}_{i,j-1}$ and whose length is the difference $\delta_{i,\max} - \delta_{i,\min}$. For the first job of a task, its illumination window starts $\delta_{i,\min}$ units of time after the release time of the task.

Energy constraint. The energy level of the radar system needs to remain below an energy threshold, E_{TH}. The constant transmission and reception of beams leads to an increase in the temperature of the different electronic components on the antenna (Kirschmann, 1998). The temperature needs to be kept within tolerable bounds else the radar equipment will breakdown. The thermal energy level of a radar system can be modeled by an exponential decay function with a look-back interval (time constant) τ (Baugh, 1973; Raemer, 1996). If $P(t)$ is the thermal power generated by a radar system at time instant t, the energy level of the system at some time instant t^*, $t^* \geq 0$ is $E(t^*) = \int_0^{t^*} P(t) e^{\frac{t-t^*}{\tau}} dt$. The energy constraint can be expressed as $E(t) \leq E_{\mathrm{TH}}, \forall t$. The energy constraint has become an important consideration for modern radars because of the shrinking footprints of antennas; smaller surface areas have decreased the heat dissipation rate and antenna elements are prone to failure because of over-heating.

Timing constraint. The timing constraint of a dwell job is defined as follows.

Definition 2.4 (In-time completion of dwell jobs). A dwell job executes to completion if there is no interruption during its sending and receiving phases. A dwell job completes in time if and only if its illumination time lies within its illumination window.

The non-preemptibility of a dwell job ensures that it is sufficient for a job to start on time if it has to complete on time. A dwell task completes in time if all its jobs complete in time. The following definition states the timing constraint of a dwell task.

Definition 2.5 (In-time completion of dwell tasks). A dwell task completes in time if and only if all of its dwell jobs complete in time.

Problem statement. The dwell scheduling problem is that of determining a schedule in which all dwell tasks in the system meet their timing constraints without violating the energy constraint. A radar system always performs search tasks to locate targets that enter the surveillance space. When new targets are detected, track tasks may be generated (depending on the threat level posed by the targets) to monitor the targets closely.

We are interested in determining if a new task can be accommodated in the antenna schedule. If a new task can be admitted, the schedule is suitably updated. When a new task cannot be added, we assume that the scheduler provides feedback to a resource/QoS manager, which can then determine new parameters for tasks to ensure that all tasks are scheduled, or select tasks that can be dropped. In this work, we deal with the scheduling problem only and leave the resource allocation problem to tools like Q-RAM (Hansen et al., 2004).

The real-time dwell scheduling problem can be stated as follows: at any time instant t when a new task T enters the system, given the current schedule S, the energy threshold E_{TH} over a look-back period τ, can T be added to the schedule without violating the energy constraint or the timing constraints of tasks previously admitted? If yes, generate the new schedule S'.

3. Feasible intervals and period synthesis

Our approach involves identifying feasible intervals for each job. A job meets its timing constraint if it is scheduled to start within its feasible interval. Determining the feasible intervals offline, however, is not trivial. The difficulty can be understood by example (Fig. 3).

Suppose the n-th dwell job starts at time 0 and the minimal and maximal temporal distance are a and $a + b$, respectively. The illumination window of job $(n + 1)$ is the interval $[a, a + b)$ and can be set as its feasible interval, shown as the cross-hatched box in the first time line (Fig. 3). However, computing the feasible interval of job $(n + 2)$ is not trivial. Although job $(n + 1)$ can be scheduled at any time in interval $[a, a + b)$, it is sufficient to consider two cases: job $n + 1$ starts at time a, and job $n + 1$ starts at time $a + b$. This is because the length of the illumination window, which in the example is b, is a constant. When job $(n + 1)$ starts to execute at time a, the illumination window of job $(n + 2)$ is the interval $[2a, 2a + b)$, shown on the second time line (Fig. 3). When job $(n + 1)$ starts at time $a + b$, the illumination window of job $(n + 2)$ is the interval $[2a + b, 2a + 2b)$, shown on the third time line in the figure. The feasible interval of job $n + 2$ must be the intersection of all possible illumination windows (if it is to be known offline). Unfortunately, there is no overlap between the two illumination windows in our example. In other words, if job $(n + 1)$ is allowed to start at any time in its illumination window $[a, a + b)$, the length of the feasible interval of job $(n + 2)$ is zero.

3.1. Period and deadline synthesis

The motivation for the period synthesis phase is to limit the scheduling intervals within which jobs may execute so that jobs can be scheduled anywhere within their restricted interval (without knowledge about earlier and future dwells) and still meet their temporal distance constraints.

Again, job n starts at time 0 and the minimal and maximal temporal distance are a and $a + b$, respectively. Hence, the illumination window of job $(n + 1)$ is $[a, a + b)$.

Rather than allowing job $(n + 1)$ to start at any time in its illumination window, the Period Synthesis Algorithm limits job $(n + 1)$ job to the interval $[a, a + \frac{b}{2})$, shown as the

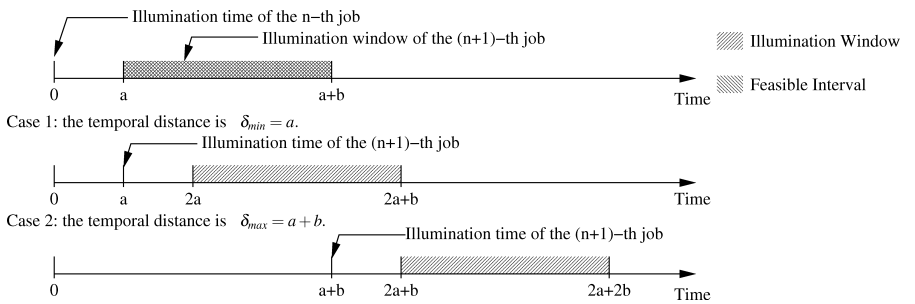


Fig. 3 Illumination windows

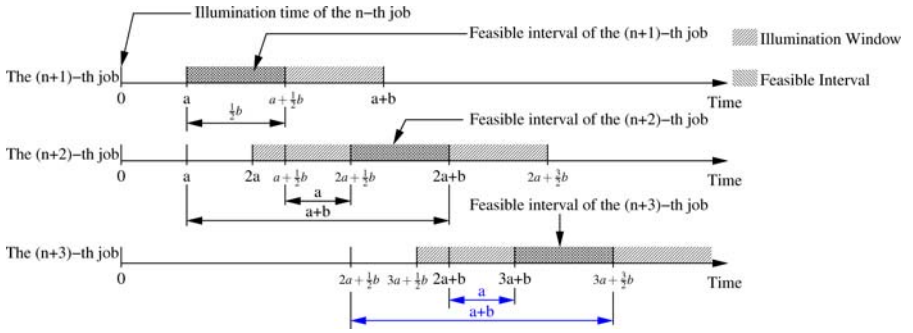


Fig. 4 Synthetic periods

cross-hatched box on the first time line (Fig. 4). Because interval $[a, a + \frac{b}{2})$ completely overlaps with the illumination window of job $(n + 1)$, the interval can be set as the feasible interval for job $(n + 1)$. When job $(n + 1)$ starts at time a, the illumination window of job $(n + 2)$ is $[2a, 2a + b)$; when job $(n + 1)$ starts at time $a + \frac{b}{2}$, the illumination window of job $(n + 2)$ is $[2a + \frac{1}{2}b, 2a + \frac{3}{2}b)$. As shown on the second time line (Fig. 4), the two illumination windows overlap at interval $[2a + \frac{1}{2}b, 2a + b)$, which can be set as the feasible interval of job $(n + 2)$. The figure also shows that the distance of any two time instants in these two feasible intervals are bounded by a and $a + b$. Hence, when job $(n + 1)$ and job $(n + 2)$ start to execute at any time within their corresponding feasible intervals, the temporal distance is bounded by a and $a + b$. Similarly, the feasible interval of job $(n + 3)$ is the interval $[3a + b, 3a + \frac{3}{2}b)$, shown on the third time line (Fig. 4). The feasible intervals have constant length $\frac{b}{2}$ and repeat every $a + \frac{b}{2}$ units of time, which is called the *synthetic period* of the task. Hence, the task can be modeled as a periodic task whose jobs are released with period $a + \frac{b}{2}$ and relative deadline $\frac{b}{2}$. We can now now define a feasible interval formally.

Definition 3.1 (Feasible intervals for radar dwells). If successive dwell jobs $\mathcal{J}_{i,j}$, $\mathcal{J}_{i,j+1}$, $\mathcal{J}_{i,j+2}$ of task $\mathcal{T}_i$ are assigned time intervals $[t_j, t'_j)$, $[t_{j+1}, t'_{j+1})$, $[t_{j+2}, t'_{j+2})$ respectively, the interval $[t_{j+1}, t'_{j+1})$ is a feasible interval for $\mathcal{J}_{i,j+1}$ if and only $\mathcal{J}_{i,j+1}$ will meet its temporal distance constraints when it is scheduled to start within its assigned interval, irrespective of the exact starting time, provided the other jobs also start within their assigned intervals.

Algorithm 1 Period and deadline synthesis

Input task $\mathcal{T}_i = \langle W_i, \delta_{i,\min}, \delta_{i,\max} \rangle$.
Output synthetic period T_i, relative deadline D_i.

1. Set $D_i \leftarrow \frac{\delta_{i,\max} - \delta_{i,\min}}{2}$.
2. Set $T_i \leftarrow \frac{\delta_{i,\max} + \delta_{i,\min}}{2}$.
3. Return D_i and T_i.

We now state a theorem on the optimality of the synthetic period T_i, and relative deadline D_i computed by the Period and Deadline Synthesis Algorithm (Algorithm 1). By optimal, we mean that a periodic interval of length D_i and period T_i is the *largest constant-length periodic interval* that can be assigned to each job.

Theorem 3.2. Given a dwell task $\mathcal{T}_i = \langle W_i, \delta_{i,\min}, \delta_{i,\max} \rangle$, $D_i = \frac{\delta_{i,\max} - \delta_{i,\min}}{2}$ is the length (relative deadline) of the maximal length periodic interval and $T_i = \frac{\delta_{i,\max} + \delta_{i,\min}}{2}$ is the period of the maximal length periodic interval for the task.

Proof: To determine a constant period, T_i, and a relative deadline, D_i, we consider the following simple requirements.

1. The illumination time of a dwell job, $\mathcal{J}_{i,j}$, can be anywhere in the interval $[t, t + D_i)$ where t is the start of a new period for the dwell task. The illumination time of the subsequent dwell job can be anywhere within the interval $[t + T_i, t + T_i + D_i)$ because of the periodicity that we would like to maintain. To satisfy the maximum temporal distance constraint, we must have $\delta_{i,\max} \geq T_i + D_i$. This is the limiting case when a dwell has its illumination time at the start of its period and the succeeding dwell has its illumination at its deadline.
2. Similarly, to satisfy the minimum temporal distance constraint, the following condition must hold: $\delta_{i,\min} \leq T_i - D_i$.

The largest feasible periodic interval is realized when both the conditions indicated are, in fact, satisfied as equalities. Combining solving the simultaneous linear equations, we obtain

$$T_i = \frac{\delta_{i,\max} + \delta_{i,\min}}{2} \text{ and } D_i = \frac{\delta_{i,\max} - \delta_{i,\min}}{2}.$$

□

4. Online dwell packing

A template is the basic scheduling unit in our discussion on dwell scheduling. Within each template, multiple dwells may be packed, in other words, nested or interleaved, so that the actual time that the antenna spends idling is reduced.

Definition 4.1 (Template). A template is a fixed-length partial schedule consisting of one or more dwells. Within a template, dwells maybe nested or interleaved. Templates are non-preemptible, and once a template starts execution, it will continue till completion.

When the scheduler packs dwells in a template, it needs to ensure that one dwell does not interfere with the completion of another dwell. Also, the scheduler should ensure that the energy threshold is not exceeded by over-aggressive packing of dwells. The algorithms presented in this section will be used as subroutines when we compute the overall finite-horizon schedule in Section 5.

The problem of producing the best (in terms of the number of dwells) sequence of dwells to execute over a finite time is reducible to the bin packing problem, and is NP-hard (Garey and Johnson, 1979). In related work (Shih et al., 2003 a, b, 2004), this problem was addressed by constructing a library of templates, offline, using a branch-and-bound algorithm. In all earlier work, the scheduler, given the set of tasks to execute, would build a schedule by selecting a suitable sequence of templates from the library. The limitation of such an approach is the need to know before-hand all possible dwell types (characterized by the power function and the send, wait and receive times); without this information a template library cannot be constructed. In practice, however, the parameters of a dwell can be assigned in a manner that maximizes the system utility (Hansen et al., 2004).

When dwells are synthesized dynamically, it is not possible to build a suitable library of templates.

In our work, we use a heuristic and formulate an online algorithm for packing dwells. Before describing the algorithm, we introduce the idea of *cool-down time* for a radar dwell. The cool-down time for dwell W, $C(W)$, is the duration that the antenna must idle before it can send out the beam for dwell W. The purpose of cool-down time is to allow the radar system to idle sufficiently so that executing W will not violate the energy constraint. It is intuitive that the maximum idling time will be required when the radar system has reached the energy threshold E_{TH} just before dwell W needs to be executed. The exponential function approximates rather closely the energy consumption of a radar (Baugh, 1973; Shih et al., 2003b). Using this function, the energy generated by dwell W is $\int_0^{e(W)} P(W,t)e^{\frac{t-e(W)}{\tau}}dt$. We can then derive the tolerable system energy level at the start of the dwell W as

$$\mathcal{E}(W) = \min\left\{E_{\mathrm{TH}}e^{\frac{x}{\tau}} - \int_0^x P(W,t)e^{\frac{t}{\tau}}dt \mid 0 \le x \le e(W)\right\}. \tag{1}$$

$\mathcal{E}(W)$ is the maximum system energy level such that W can execute without crossing the threshold E_{TH}. Now, if E is the system energy level at time t, we would like to ensure that the system energy level is not greater than $\mathcal{E}(W)$ at time $t + C(W)$, i.e., we allow the system to cool down sufficiently before starting dwell W. With this understanding, we can easily compute $C(W)$ as

$$C(W) = \max\left\{\left\lceil -\tau \ln \frac{\mathcal{E}(W)}{E}\right\rceil, 0\right\}, \tag{2}$$

where E is the system energy level at the instant when dwell W starts execution. (For derivations of 1 and 2, see the appendix.)

We present two algorithms, TEMPLATEPACK (Algorithm 2) and TEMPLATEINSERT (Algorithm 3), which are similar to each other but have slightly different semantics. TEMPLATEPACK takes a set of dwells and an energy level E, and packs as many dwells as possible within the template. The energy level E is the expected system energy level when the created template will start executing. The algorithm considers the energy constraint and tries to nest and interleave dwells to avoid unnecessary idling. However, the algorithm might not be able to schedule all dwells within one template. In our algorithm, and for the rest of this work, we will use the symbol L to represent the length of a template. TEMPLATEINSERT attempts to insert dwell W' into a (possibly non-empty) template, A. We note that the inputs for the two template algorithms are identical to the inputs required even with offline template construction (Shih et al., 2003b). The fundamental difference is that in prior work the dwell packing was not done online but a library lookup was used to retrieve the appropriate template.

Algorithm 2 TEMPLATEPACK

Input set S of dwells, initial energy level E.
Output template A', the set of scheduled dwells S'.

1. Sort S in a non-increasing order of dwell lengths.
2. Initialize $S' \leftarrow \emptyset$ and $p \leftarrow 0$. p indicates the next position in template A' where a new dwell can potentially be started.
3. Select the largest dwell W from S that has not been considered yet; set $p' \leftarrow p + C(W)$. ($C(W)$ depends on E.)
4. If $p' + e(W) \geq L$, dwell W cannot be scheduled in this template; go to step 3.
5. If W cannot start at position p' because of collisions with other previously scheduled dwells, keep incrementing p' by 1 until W can be inserted or $p' \geq L$. If at any point in these iterations, $p' + e(W) \geq L$, W cannot be scheduled; go to step 3 and select a new dwell.
6. If W was successfully inserted:
   (a) Set p' as the start time of W.
   (b) Set $E \leftarrow Ee^{-\frac{(p'-p+e_T(W))}{\tau}} + \int_{t=0}^{e_T(W)} P(W,t)e^{\frac{t-e_T(W)}{\tau}}dt$.
   (c) Set $p \leftarrow p' + e_T(W)$ and $S' \leftarrow S' + \{W\}$.
7. Repeat steps 3 through 6 until all dwells have been examined or $p \geq L$.
8. Return the set S' and template A'.

It is easy to see that the dwell packing algorithm is a variation of the largest-item-first packing policy. The inclusion of some idle time before each dwell prevents energy constraint violations. The choice of the largest-item-first policy is motivated by the fact that larger dwells are harder to insert at a later stage in the algorithm. Furthermore, larger round trip times are often associated with dwells that have greater transmission times—mainly because greater transmission power and duration are required to track targets that are far away. By inserting the larger jobs first, we have a better chance of nesting (or interleaving) smaller jobs. A subtle question that might arise is: why is there no check to ensure that the receive phase of a dwell does not violate the energy constraint when another dwell is nested within or interleaved with it? The reason for this is that echo reception has almost no impact on the system energy level. The energy increase while receiving reflected waves is close to zero (Kirschmann, 1998; Hansen et al., 2004) and (empirically) always less than $E_{\text{TH}}(1 - e^{-\frac{1}{\tau}})$. This small energy increase is sufficient to guarantee that E_{TH} will not be exceeded irrespective of the energy level before the receive phase (as long as it is $\leq E_{\text{TH}}$).

TEMPLATEPACK is bound to terminate when we have exhausted all possible locations for dwell insertion. Since the number of locations is a constant (the length of a template), this routine has a time complexity of $\Theta(1)$. Since TEMPLATEINSERT is a variation of TEMPLATEPACK, it has similar time complexity.

5. Finite horizon dwell scheduling

The centerpiece of our work is finite horizon scheduling. For the remainder of this work, we will consider schedules that span a finite horizon, H. In our approach, jobs are assigned to templates that lie completely within the respective feasible intervals. If a job cannot be

Algorithm 3 TEMPLATEINSERT

Input template A, dwell W', initial energy level E.
Output TRUE or FALSE depending on whether W' was inserted or not.

1. Let S be the set of dwells already present in template A.
2. Set $S \leftarrow S + \{W'\}$.
3. Call TEMPLATEPACK with S and E.
4. If S', the set of scheduled dwells returned by TEMPLATEPACK, is not the same as S, return FALSE, else set $A \leftarrow A'$, where A' is the template returned by TEMPLATEPACK and return TRUE.

scheduled within its feasible interval, we consider it to have missed its deadline. A task is admitted if it can be scheduled without a deadline miss over the finite horizon. A task T admitted at time t is guaranteed not to miss a deadline until time $t + H$. T may miss a deadline after $t + H$ but the scheduler should be able to generate a warning sufficiently ahead of the deadline miss. For this reason, H is chosen to be at least as long as the response time of a radar operator to an imminent deadline miss.

The finite horizon is broken up into a sequence of n templates. When the system is initialized, the n templates are A_0 through A_{n-1}. When template A_0 starts execution, template A_n is constructed using dwells which have feasible intervals that overlap with A_n but have not been scheduled yet; while A_0 is executing, the templates $A_1, \ldots, A_n$ form the horizon. Template A_k spans the time interval $[k \times L, (k + 1) \times L)$.

There are two aspects to finite horizon scheduling: **admission control** and **schedule increment**. We will now discuss both these aspects in detail.

Admission control. Let us assume that a new task T arrives at the admission controller at time t. We admit T if it can meet all its deadlines in the finite horizon that starts at $k \times L$ and ends at $(k + n) \times L$, where k is the smallest integer such that $t < k \times L$. To meet the temporal distance constraints, we first synthesize the period s and relative deadline D for task T. We now treat T as a periodic task with release time $k \times L$. Since A_{k-1} will be executing when T arrives at the admission controller, the first job of task T can be inserted, at the earliest, in template A_k. Admission control is carried out by actually building the schedule for the length of the finite horizon and ensuring that no job of T misses its deadline over the time frame of length H. For each dwell job W of T, the scheduling algorithm sequentially processes each template that is completely within the feasible interval of dwell W (using the TEMPLATEINSERT procedure) until it finds a template in which W can be inserted without having to remove any other dwell or creating an energy constraint violation. If all dwells that have deadlines inside the horizon can be scheduled, task T is accepted; else it is rejected. Dwells that have feasible intervals that cross the horizon boundary may or may not have been scheduled. If such dwells have not been scheduled, then they will be considered during the schedule update phase.

As an example (Fig. 5) consider a task T with parameters $\delta_{\min} = 100$ ms, $\delta_{\max} = 400$ ms. The length of template, L, is 50 ms and the finite horizon window is 850 ms or 17 templates. We schedule T as a periodic task with synthetic period 250 ms and relative deadline 150 ms. From the illustration, we conclude that we can admit task T because all jobs which have deadlines within the finite horizon can be scheduled. The fourth job has a feasible interval that extends beyond the finite horizon; we may not have been able to schedule the job 4 in the first two templates that are within the horizon and inside the feasible interval, but we do not reject the task because we might be able to schedule the job in the next template, when

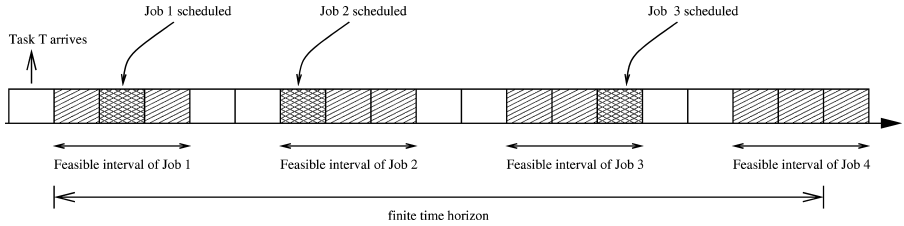


Fig. 5 Admission control

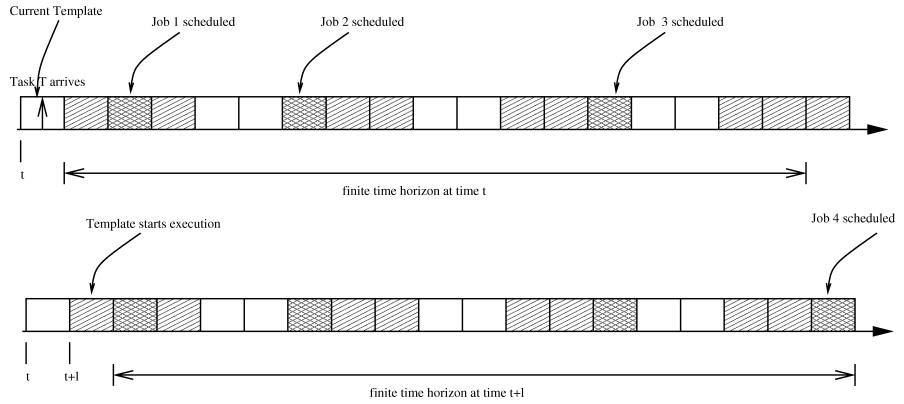


Fig. 6 Schedule increment

it enters the horizon, i.e., on completion of the template currently being executed. Even if we are unable to schedule the job at a later time, the scheduler can generate a warning 850 ms in advance of the deadline miss and a higher-level entity can modify the workload appropriately.

Schedule increment. Our scheduling approach involves building schedules for the duration of the finite horizon. When a template starts executing, the horizon moves by one template and the scheduler looks ahead and starts scheduling dwells in the new template. Our admission control example can be extended (Fig. 6). When the template that began execution at time t completes, the template slated for time $t + L$ starts executing. The finite horizon window slides to the next template, which will initially be empty. In the figure, we highlight the fact that job 4 of task T is scheduled in this template. In fact, jobs belonging to other tasks might have been allocated to the same template if they were waiting to be scheduled.

When we slide the finite horizon window, the main question to be answered is: how do we determine the dwells that need to be scheduled in the new template (which is at the edge of the horizon)? For each task that has been admitted and is still present in the system, we determine if the new template, A_{new}, is within the feasible interval of a job that is yet to be scheduled. We order all the waiting jobs based on EDF priorities. We first select all the jobs that have deadlines such that A_{new} is the last full template that can be used to schedule them[5] and pack them into the template using the TEMPLATEPACK routine. If one or more jobs

[5] By EDF, these will be the jobs with the highest priority. We include jobs that have deadlines at the end of A_{new} or have deadlines that expire in the middle of the next template.

cannot be scheduled, we raise a warning. Then, if there is still some space in the template,[6] we follow EDF priority and attempt to insert (using the TEMPLATEINSERT routine) other jobs, with later deadlines, until there is no space in the template.

Implementing a finite horizon scheduler. A scheduler with a finite horizon of n templates can be implemented using a queue of n elements: each element in the queue is a data structure representing one template. Whenever one template completes and the next template is removed from the queue, a template corresponding to the end of the horizon is computed and added to the queue.

Updating the schedule when a template completes requires knowledge of the dwells that are eligible for scheduling. This information can be obtained by maintaining, for each task, the release time of the next unscheduled dwell. Combining this information with other task characteristics (synthetic period and relative deadline), it is easy to determine if the feasible interval of an unscheduled dwell intersects with the new template that needs to be created.

To avoid exceeding the energy threshold, it is necessary to annotate the schedule with energy levels. However, when exact energy levels are used, the scheduling complexity increases. Whenever some dwell W is inserted in template A_i, we need to verify that the energy constraint will not be violated by any of the future templates.[7] We solve this problem by assuming that the energy level at the start of each template is E_{TH}. This assumption, which might appear pessimistic, is in fact reasonable because the system energy level rises to E_{TH} after a sufficient number of dwells have been executed and tends to remain at that level, especially if a resource manager assigns parameters that maximize the utility (thereby consuming more resources whenever possible). Therefore, when computing cool-down times (according to Eq. (2)), we set $E \leftarrow E_{\mathrm{TH}}$. By doing so we ensure that each template is safe to execute even when the system energy level has reached E_{TH} just before the template starts; the dwell packing algorithms guarantee that the energy level will not exceed E_{TH} during a template's execution.

Modular schedule update. An change to a schedule is *modular* if it occurs only in particular portions of the schedule and can be made without affecting other portions or causing constraint violations. Our scheduling mechanism possesses this property. When new tasks are admitted, the schedule changes only within the templates in which new jobs are inserted. The use of templates provides the modular update property because a dwell is assigned only to a template that lies completely within its feasible interval. As a result, irrespective of the exact start time of the dwell within the template, it will always satisfy its temporal distance constraints. Further, since a job is inserted into a template only if it will not cause the energy level to exceed E_{TH}, and since job insertions assume that the energy level at the start of a template is E_{TH}, job insertions are guaranteed to be safe in terms of the energy constraint. The modular schedule update property is attractive because it reduces the complexity in implementing the scheduler.

Lastly, we remark that the combination of modular updates and feasible intervals creates parallelism in the scheduler, especially when tasks need to be admitted or rejected. Suppose a task T has m jobs with deadlines within the finite horizon. These m jobs can be scheduled in

[6] This is possible even if we raised a warning because we might not have been able to schedule a long job but a short job might be schedulable.

[7] Such tests are required because the energy level which was used when creating the succeeding templates might be significantly different from the energy level that will now propagate along the schedule. This variation might cause an energy violation. It is, of course, true that there will be no violation in template A_i because the subroutines for creating templates and inserting dwells take the energy level into account.

parallel because they will have different feasible intervals; they could be scheduled anywhere within the feasible intervals and meet timing constraints, and the modular update property ensures that each of the m scheduling decisions can be made independently.

6. Performance evaluation

The performance of our scheduling mechanism was evaluated through extensive simulations using the Omnet++ simulator (Varga, 2000). We were interested in the behavior of the online template generation algorithm as well as the impact of the finite horizon guarantee. The parameters for radar dwells were chosen based on the parameters described by Kuo et al. (2002) and similar to the parameters used in prior work (Shih et al., 2003 a,b, 2004).

Through our experiments, we also evaluated the impact of dwell nesting and interleaving (by varying template lengths), the effect of varying the finite horizon and the impact of the energy constraint.

6.1. Experiment setup

Although the online template generation algorithm can handle all possible dwells, for the purpose of comparison we restricted the dwell types to those employed in past work. The workload parameters are listed in Table 1. The three element tuples for execution time and power consumption denote the time and average power consumption for the sending, round-trip delay and receiving phases of a dwell task. The pair $(C_{\min}, C_{\max})$ denotes the minimal and maximum temporal distance between the dwells of a task.

Search tasks were always present in the system. Each search task could potentially generate one confirmation task with probability 0.05. On completion of a confirmation task, track tasks were generated based on a uniform distribution for the results of the confirmation task. A confirmation task could generate a track task with probability p_t, varied from 0.1 to 0.8. The track task could be a high-precision track with probability p_{hpt}, a precision track with probability p_{pt}, or a normal track with probability p_{nt}, such that $p_{hpt} + p_{pt} + p_{nt} = 1$. In our simulations, we set $p_{hpt} = p_{pt} = p_{nt}$. At peak load, with these parameters, about 100 new tasks arrive each minute.

The energy threshold was 250 J with an energy look-back period of 200 ms. We simulated the radar system for a duration of 720 s and the results are averaged over 12 simulation runs. Our simulations were run on a single Pentium 4 2 GHz processor with 512 MB memory.

6.2. Performance metrics

We use three metrics to measure the performance of our algorithms: mean utilization, mean rejection rate, and mean (admission control/scheduling) overhead.

Table 1 Dwell task parameters

Tasks	Importance	Execution time (ms)	consumption (kW)	$(C_{\min}, C_{\max})$ (ms)
High priority search	1	(1, 4, 1)	(5, 0, 0.1)	(600, 930)
Confirmation task	2	(1, 4, 1)	(4, 0, 0.1)	(560, 800)
High-precision track	3	(0.5, 1, 0.5)	(4, 0, 0.1)	(60, 280)
Precision track	4	(1, 2, 1)	(4, 0, 0.1)	(250, 600)
Normal track	5	(1, 2, 1)	(3, 0, 0.1)	(850, 1190)
Low-priority search	6	(0.5, 1, 0.5)	(3, 0, 0.1)	(850, 1700)

Definition 6.1 (Mean utilization, U). The mean utilization is the fraction of time the antenna is either sending or receiving electromagnetic beams.

$$U = \frac{\text{Time antenna is in use}}{\text{Total time}}.$$

The utilization acts an indicator of whether the scheduler is making effective use of the antenna.

Definition 6.2 (Rejection rate, RR). The rejection rate refers to the fraction of tasks that miss deadlines when compared to the total number of tasks encountered during the evaluation period (simulation run). This fraction includes tasks that were initially admitted but were later found to be unschedulable and tasks that were rejected during the admission control phase.

$$RR = \frac{\text{Number of tasks that are rejected or miss a deadline}}{\text{Total number of tasks}}.$$

The rejection rate reflects the ability of the scheduler to satisfy timing requirements. We generate workloads that typically drive the system into overload and non-zero rejection rates cannot be achieved in most scenarios. A lower rejection rate usually leads to higher utilization and reflects better schedulability.

Definition 6.3 (Scheduling overhead, SO). The scheduling overhead is the time spent in adding a task to the schedule or rejecting a task. This overhead includes the initial computation of synthetic period and relative deadline.

$$SO = \frac{\text{Total time consumed by the scheduler for admission control or schedule update}}{\text{Total number of tasks}}.$$

In the case of finite horizon scheduling, because the schedule needs to be incremented after each template completes, we account for this extra scheduling time and amortize this cost over the total number of tasks. The overhead is a measure of the scheduling efficiency; to accommodate rapid changes to the task set, scheduling overhead should be low.

6.3. Impact of template size, or does dwell packing matter?

The first experiment that we describe involved scheduling radar dwells over a finite horizon using three different template lengths. The results (Fig. 7) indicate that template size plays a role in determining the schedulability of the system; this contribution is hard to determine analytically but empirical evidence can guide us in the choice of an appropriate template length.

The template size determines the extent to which dwells can be nested or interleaved (recall the association with bin packing). By reducing the template size, we limit the packing. For the smallest template size (10 ms), the scheduler is often unable to schedule multiple dwells in a template and the antenna utilization is poor (less than 40%). Large template sizes are not good either, and there are two reasons for this. First, we are using an approximation algorithm for dwell packing and slot wastage due to packing increases with template size;

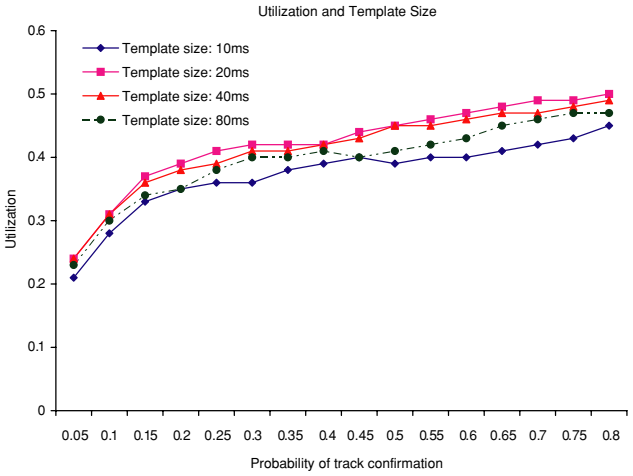


Fig. 7 Impact of template size on utilization

this leads to wasted antenna capacity. Second, templates are non-preemptible and once a template is scheduled, new dwells cannot be inserted. When templates are bigger, new tasks need to wait longer for admission and if a template is not well utilized, performance suffers. Moderate template sizes are attractive because they strike a balance between the extent of packing with non-preemption.

Template size is also influenced by the type of dwell patterns that need to be scheduled. If all tasks are associated with the same dwell pattern, template sizes can be determined accurately because the optimal interleaving can be trivially computed. If the task distribution is known *a priori*, the template size can be chosen after considering a subset of interleavings and nestings that have a high likelihood of occurrence. It is the difficulty of obtaining, before-hand, the task distribution that limits sophisticated offline analysis. Obtaining precise dwell pattern distribution is harder when a resource manager assigns task parameters to improve the quality of surveillance.

6.4. Comparisons with offline dwell packing

To test our online packing heuristic, we replaced the template library in the earlier model (Shih et al., 2004) with the new online technique and assessed the performance.

There was no significant difference in the performance of feasible interval scheduling whether dwells are packed online or not. These results (Fig. 8), which show that there is no significant change in utilization and that the overhead does not increase by more than 30 μs, confirm our belief that it is possible to pack dwells efficiently online. For these experiments, we used a fixed template length of 40 ms. The task rejection rates were consistent with the observed utilizations.

6.5. Evaluation of finite horizon scheduling

To evaluate the performance of finite horizon scheduling, we used a horizon of 15 s and a template length of 40 ms. The duration for which a task remained in the system was drawn from an exponential distribution with mean 30 s. Note that the hyperperiod of the chosen task set (based on the synthetic periods) is 30.6 s. We compared the performance with the feasible interval scheduling approach that generates schedules for a hyperperiod, i.e., provides

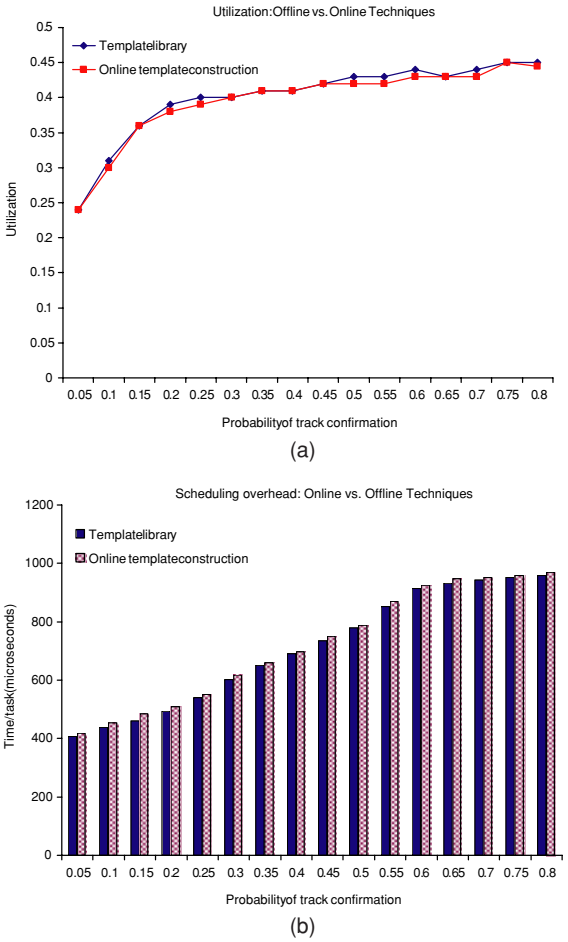


Fig. 8 Performance of online template generation

permanent guarantees (Shih et al., 2004). In these experiments, we were interested only in the improvement that the finite-horizon scheduling provides, therefore we used online dwell packing in both schedulers.

Due to our online dwell packing scheme, our current approach is not limited by a predetermined performance requirement set, which is a set of tasks that the system must be guaranteed of handling. Just for the purpose of comparison, we used a performance requirement set of 45 high priority search tasks, 10 confirmation tasks, 10 high-precision tracks, 10 precision tracks, 5 normal tracks and 20 low-priority search tasks.

In our evaluation of finite horizon scheduling, we observed a gain in antenna utilization (Fig. 9(a)) when finite horizon scheduling is used. This gain is a result of myopic scheduling which does not consider long-term interference between tasks; in schedules that span a hyperperiod, a task may be rejected because it collides with another task near the end of the hyperperiod. In reality, one of the tasks may disappear before the conflict occurs and the rejection becomes unnecessary. Finite horizon scheduling benefits by not looking too far into the future. The increased utilization observed in the experiments aligns with the fact that finite horizon scheduling admits more tasks (Fig. 9(c)). It is important to bear in mind that even small improvements in antenna utilization are significant. With the increased utilization that finite horizon scheduling provides, we are able to schedule more than 80 extra

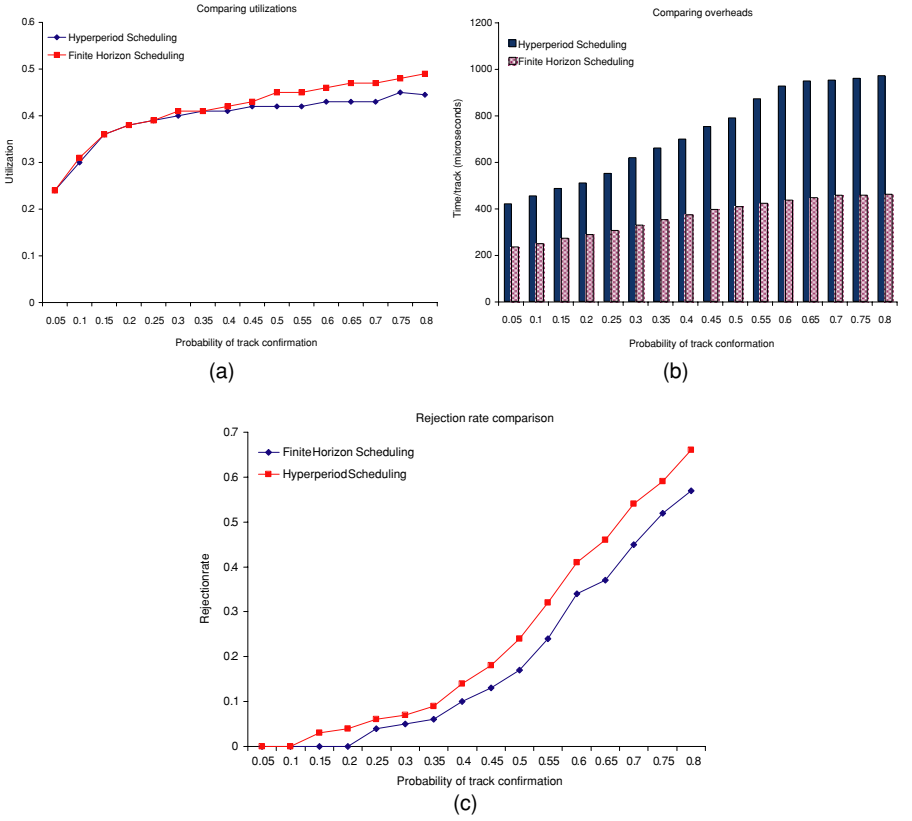


Fig. 9 Performance of finite horizon scheduling

tasks at high load when compared with permanent guarantee scheduling. Note that the simulated system may not be able to achieve 100% utilization. This is because the system has to adhere to the energy threshold, which may limit the maximal utilization of the antenna. For the task set that we chose, 50% utilization is very close to the capacity of the radar system (see Section 6.6 for more details). It is difficult to analytically account for the impact of nesting and interleaving of dwells in combination with the energy constraint; lacking an analytical characterization of dwell packing, we are unable to determine the utilization limit accurately.

The overhead of finite horizon scheduling is substantially lower than permanent guarantee scheduling (Fig. 9(b)). Since the finite horizon is shorter than the hyperperiod, schedule construction takes lesser time. In our experimental task set, the finite horizon and the hyperperiod differ by a factor of 2. In more arbitrary scenarios, this difference may be greater and building a schedule for a hyperperiod might impose an intolerable overhead. As a remark, we add that this scheduling overhead can be reduced by using more processors in parallel.

6.6. Impact of the energy constraint

The energy constraint in a radar system results in reduced antenna utilization. This reduction is unavoidable because the radar needs to cool between dwells. To understand the impact of the energy constraint, we conducted two sets of experiments (Fig. 10). Each set of

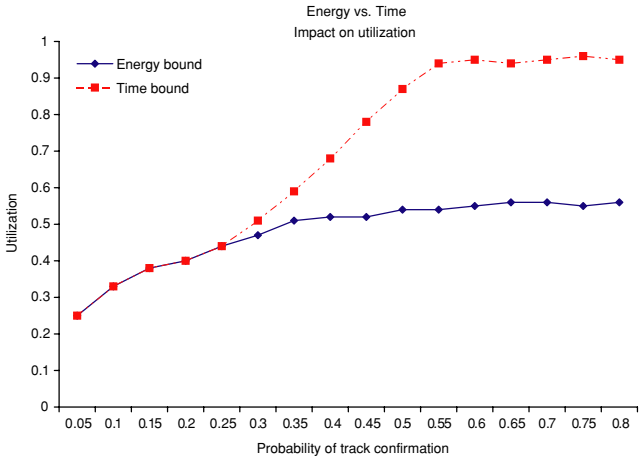


Fig. 10 Effect of time and energy constraints

experiments consisted of 12 simulation runs. The workload generation model was the same as described earlier.

- The first set of experiments was focused on determining the utilization achieved by our scheme in the absence of the energy constraint. For these experiments, the template size was set to 40 ms and the finite horizon was 15 s. These experiments, which we call **time bound** because the timing constraints alone affect the antenna utilization, indicate that the energy constraint is the limiting factor. In the absence of the energy constraint, nearly 95% antenna utilization can be achieved.
- The second set of experiments, the **energy bound** experiments, were performed to evaluate the impact of the energy constraint. For this set of experiments, we retained the energy threshold of 250 J, but we eliminated the receive phase of a dwell so that the results are not affected by the shortcomings of the dwell packing algorithm. Each dwell thus constituted a cool-down time and a transmission time. We did not use finite horizon scheduling, but simply scheduled dwells on a non-preemptive *earliest deadline first* basis. No deadline guarantees were provided; the objective was to simply determine the maximum utilization that could be achieved. We chose EDF because of its optimality when scheduling non-preemptive sporadic tasks (Jeffay et al., 1991). In fact, any non-idling policy would have ensured that the antenna was maximally utilized. These experiments revealed that the antenna utilization saturated at about 55%: the antenna was transmitting beams for 55% of the time when operating under the energy constraint alone.

The bottleneck for dwell scheduling is the energy constraint which limits the antenna usage, even under ideal circumstances—no interleaving, no distance constraints—to about 55%.

In the light of these experiments, antenna utilization of 50% in the presence of time and energy constraints is a satisfactory performance level. Transforming tasks with distance constraints to periodic tasks with relative deadlines less than the periods leads to an efficient scheduling mechanism.

It is true that feasible intervals are a simple mechanism for spacing job release times and deadlines. Constraint programming (Van Hentenryck and Deville, 1991; Baptiste et al., 2001) can also be employed to ensure that tasks will meet their timing requirements, but propagating constraints along the schedule requires more computation and it restricts the

ability to insert dwells into their feasible intervals in parallel. While constraint propagation may result in better system performance, our empirical results suggest that further utilization gains may be marginal at best.

6.7. Choosing the finite horizon

An important issue that needs to be answered with regard to finite horizon scheduling is the choice of the finite horizon. Currently, there is no analytical technique for determining the optimal finite horizon. We ran experiments with four different choices for the finite horizon: 9 s, 15 s, 21 s and 27 s (see Fig. 11). The mean task lifetime is 30 s.

Extremely short-sighted policies (example: finite horizon of 9 s) perform poorly because they admit tasks as soon as there is some capacity but without consideration of what might happen when the horizon expires. These policies are comparable to best-effort policies. They cannot give sufficient guarantees and tasks that were admitted once need to be rejected later resulting in high rejection rates. Long horizons (comparable to the hyperperiod) are also bad choices because they tend to be extremely conservative in dynamic environments. Based on our experiments, finite horizons of length 15 s and 21 s give the best performance. Of these, a horizon of 15 s becomes the natural choice because it imposes a smaller scheduling overhead.

6.8. Finite horizons and mean task lifetimes

In the previous set of experiments, we considered four different horizons while keeping the mean task lifetime fixed. To understand the relationship between the length of the horizon and the task lifetime, we experimented with mean lifetimes of 15 s and 45 s.

When the mean task lifetime is 15 s (Fig. 12(a)), we find that a finite horizon of 10 s or 15 s performs best. With 15 s being the mean lifetime, it is intuitive that a horizon that is equal to the lifetime will perform well. But a horizon of length 10 s performs equally well and would be a better choice because of the lower overhead. A horizon of 5 s is short-sighted and results in more task rejections. A similar trend is observed when the horizon is set to 20

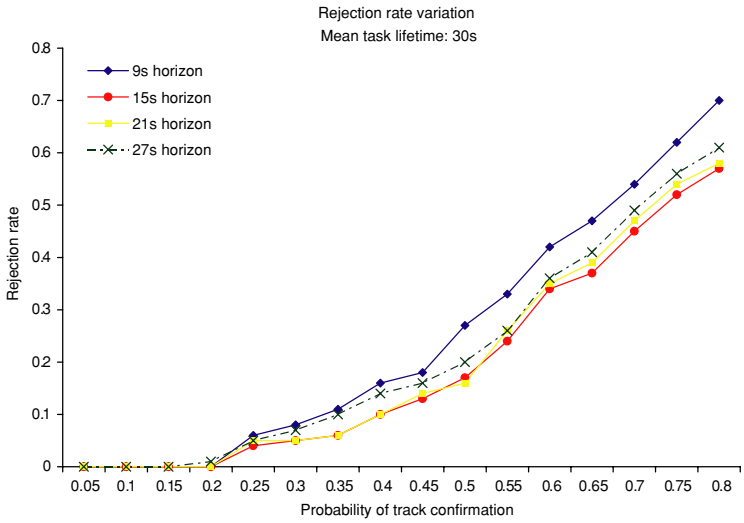
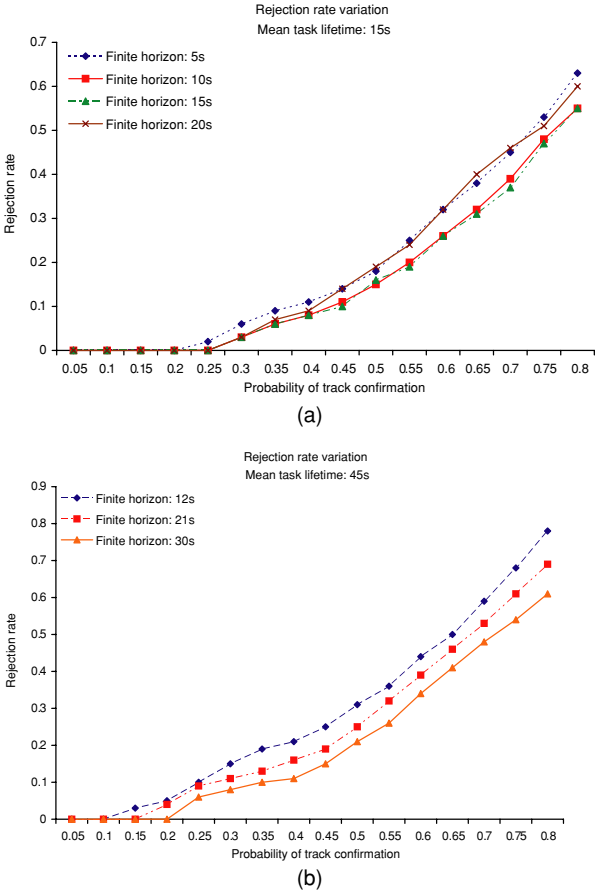


Fig. 11 Finite horizons and rejection rates

Fig. 12 Impact of task lifetimes



s. The conservativeness of the 20 s horizon results in performance that is comparable to, and even marginally better than, a performance using 5 s horizon.

With a mean task lifetime of 45 s (Fig. 12(b)), a horizon of 30 s performs best. 45 s is greater than the hyperperiod (30.6 s) and scheduling is now reduced to hyperperiod scheduling. Any smaller horizon leads to a higher task rejections—many tasks are admitted initially but need to be rejected later because of scheduling conflicts.

From our experiments (Figs. 11 and 12), we draw two inferences:

1. The rejection rate increases with an increase in mean task lifetime. When targets need to be tracked for longer, the number of new targets that can be supported reduces leading to higher rejection rates.
2. The optimal horizon increases with mean task lifetime. However, for good performance (lower rejection rates), the horizon need not equal the expected lifetime. Horizons that are smaller than the expected lifetime perform equally well.

Good choices for the horizon can be obtained from experiments and these choices result in good performance. We conjecture that the horizon should cover the lifetime of a sufficient number of tasks but not necessarily all the tasks. In our experiments, the best horizon appears close to the median task lifetime.

Determining the optimal horizon analytically is challenging because task lifetime is not the only factor affecting the choice. The periodicity of the tasks also plays an important role. When task frequencies are high (periods are low), even a small horizon can capture the demand of the existing taskset and new tasks can be rejected. When the frequency is low, only a few instances of a task may be scheduled within the horizon leading to aggressive task admission, which in turn leads to increased task rejections at a later time. Aggressive task admission is detrimental because multiple tasks may be evicted from the schedule (after the horizon in which a task was accepted expires) by admitting even one extra task, especially if the task has a small period and requires many slots in the schedule.

A scheme that tunes that length of the horizon online on the basis of the observed task lifetime and the rejection rates will yield the best performance. A tradeoff needs to be made between the horizon length and the schedule construction time. When workload is moderate, long schedules can be constructed quickly and a lengthier horizon can be considered. As scheduling time increases, the horizon can be reduced. Another approach would be to use different horizons for different tasks; tasks perceived as more critical may be scheduled over a long horizon and other tasks may be scheduled over a shorter horizon.

6.9. Pessimism in cool-down times

During the template packing algorithm, we assume that the energy level at the beginning of the template is E_{TH} and this might lead to cool-down times that are longer than absolutely required. In general, the energy level may not be E_{TH} but keeping track of the exact energy level to calculate cool-down times requires a highly complex scheduler.

If dwell packing depends on the exact energy level, any change to a template requires a propagation of energy levels to all subsequent templates which, in turn, may change all subsequent templates. Yet, there is some pessimism in isolating templates by assuming that the energy level is E_{TH} before scheduling each template. To understand the effect of this assumption we captured a trace of the energy levels over a simulation run (Fig. 13) with medium workload (probability of track confirmation = 0.6).

The trace, captured from a simulation of 80 s of radar operation, shows that the energy level approaches the threshold of E_{TH} =250 J very early in the simulation run and remains high throughout the run. This observation justifies assuming the energy level is always E_{TH}. There is a short duration for which the energy level is significantly lower than the threshold (Fig. 13(a)) and that is the only operating region when a scheme that uses the current energy level might offer some benefit. But given the short duration for which the radar system functions at an energy level much less than E_{TH}, the overwhelming complexity of using the current energy level at all times negates the potential benefit.

Looking at the energy snapshot of a smaller interval when the energy level is close to E_{TH} (Fig. 13(b)) we see that the energy level fluctuates between 245 J and 250 J. This observation confirms our intuition about approximating the energy level with E_{TH}. The performance loss due to this approximation is insignificant. Other traces (not included in this article) exhibited the same characteristics.

7. Related work

Dong (2001) and Dong et al. (1998) developed a class of template-based scheduling algorithms for tasks with minimum rate guarantees and temporal distance constraints. Though there is no explicit guarantee for maintaining a minimum temporal distance in this body of work, guaranteeing a minimum rate is likely create a minimum temporal distance between instances of the same task. This effect has not been studied. There are also significant

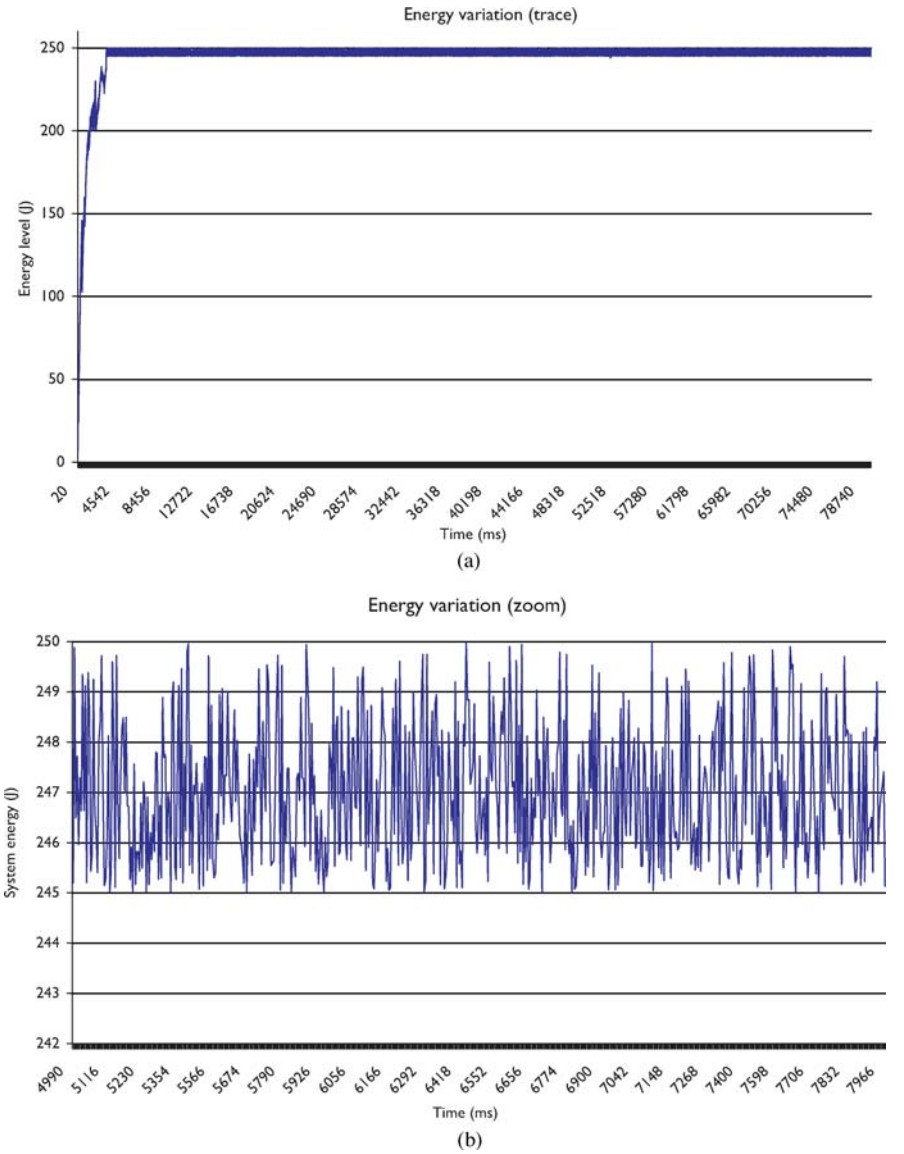


Fig. 13 Energy trace over a simulation run

differences between the model used by Dong and the radar model. In Dong's work, a template is a cyclic executive that is repeatedly scheduled. In our work, a template is only a part of a schedule. Additionally, Dong focuses on message scheduling in real-time systems and is not concerned with task models that are similar to radar dwells, i.e., with send/wait/receive phases and with energy constraints.

(Shih et al., 2003 a, b, 2004) and Gopalakrishnan et al. (2004) introduced the notion of template-based scheduling for radar dwells. This body of work developed techniques for constructing templates (a partial schedule obtained by nesting and interleaving dwells) using a branch and bound technique; this was an offline mechanism that resulted in a template

library. At run-time, when a set of dwells needed to be combined, a library lookup was used to obtain a suitable template (if any). The major difficulty with this approach was the need to know all possible dwell types. This would limit a resource manager from choosing dwell parameters for tasks to minimize system error. Even when dwell types are known, the library would need exponential storage to retain all possible templates. The use of templates, however, simplified dwell packing and verification of the energy constraint.

Prior to Shih et al., Kuo et al. (2002) had proposed a rate-based approach for scheduling radar dwells while offering a minimum performance guarantee. Their work balanced the semantic importance of a task and its scheduling priority via resource reservations. They did not consider other factors that affect radar dwell scheduling: energy constraint and dwell packing.

The nature of radar dwells renders analytical estimates of schedulability hard. Lee et al. (2003) advocated the use of *schedulability envelopes* to abstract the utilization of radar dwells while accounting for dwell packing and energy constraints. Their approach involved an offline regression over the entire schedulable volume to obtain a closed-form expression that was sufficient to guarantee schedulability. Construction of schedulability envelopes can be computationally very expensive (even when run offline) due to the high dimensionality of the schedulability volume. This technique also requires knowledge of possible dwell types—similar to the requirement for a template library.

Q-RAM (Hansen et al., 2004; Ghosh et al., 2004) has been used by Ghosh, Hansen, Rajkumar and Lehoczky to optimize resource allocations for radar tasks using the antenna capacity and energy threshold as constraints. They approximate the utilization bound for checking schedulability by restricting dwell packing to dwells of the same period and creating virtual tasks that need not be packed any further. This approach has been successful despite the pessimism introduced at the dwell packing stage.

No prior work has dealt with the notion of a finite scheduling horizon, which, as we have seen, results in improved performance when traffic conditions change rapidly.

8. Conclusions

Real-time radar dwell scheduling is a challenging problem because of the multifarious constraints that need to be satisfied. We presented a periodic task model for radar dwells which actually have minimum and maximum temporal distance constraints. Our task model suggests a scheduling algorithm based on feasible intervals, a concept we have developed, for radar dwells. When jobs are scheduled in their feasible intervals, temporal constraints will always be satisfied.

Dwell packing, by nesting and interleaving dwells over a scheduling interval, is necessary for making effective use of the antenna. Prior work in this area required that dwells be packed offline and stored in a template library. We suggest a heuristic that is easy to implement and exploits the characteristics of radar dwells. The performance results suggest that online dwell packing may be as effective as a template library. Online dwell packing provides resource managers with a lot of flexibility in assigning parameters to tasks. Furthermore, scheduling intervals, in combination with feasible intervals, provide a nice isolation property that helps us insert new dwells without upsetting the entire schedule, which we call the modular schedule update property.

In this work, we have also introduced the notion of a finite horizon guarantee which allows the scheduler to add more tasks over a finite time window. In a highly dynamic system like a radar, not many tasks execute for extended periods of time, and permanent scheduling guarantees, such as those provided by schedulers operating over a hyperperiod, lead to

poor resource allocation. Finite time scheduling guarantees are sufficient and necessary for improved performance. Another application of finite-horizon scheduling is as a support for optimizing resource allocation. New tasks that arrive can be admitted, at the minimum QoS level, based on the finite scheduling horizon and this time can be used by a resource manager for reassigning task parameters to improve the system utility. A finite horizon would provide the resource allocation algorithm the deadline by which it should complete the allocation.

We evaluated the suggested algorithms via simulations and confirmed our belief that we can provide greater scheduling flexibility without compromising on antenna utilization or scheduling overhead by using the notion of feasible intervals and finite horizons. In future work, we would like to measure radar performance in terms of the overall system utility, rather than utilization and missing rates, because utility reflects the *benefit* that the system provides to the end-user, and is an easier measure to use in comparisons. The problem of determining the ideal finite horizon is still open although we have explored, empirically, several factors that affect the length of the horizon.

Appendix A: Deriving the cool-down time

The first step towards obtaining the cool-down time for dwell W is computing the tolerable system energy level, $\mathcal{E}(W)$, at the start of W. The energy (heat) generated by W over the course of its execution is

$$E^W = \int_{t=0}^{e_W} P(W,t) e^{\frac{t-e_W}{\tau}} dt.$$

Naively, we might assume that $\mathcal{E}_W$ is $E_{\mathrm{TH}} - E^W$. However, this is incorrect because this condition only guarantees that the system energy level does not exceed E_{TH} at the end of the dwell. However, it may be that the thermal energy level increases beyond E_{TH} in $[0, e_W)$ and cools to a value $\leq E_{\mathrm{TH}}$ by the end of the dwell. To avoid such situations, we need to ensure that the energy level does not exceed E_{TH} at any time instant $0 \leq x \leq e_W$ when dwell W executes.

If E_0 is the energy level at the time when dwell W starts transmission then at any time instant $0 \leq x \leq e_W$ the energy level of the system is

$$E_x^W = E_0 e^{-\frac{x}{\tau}} + \int_{t=0}^{x} P(W,t) e^{\frac{t-x}{\tau}} dt.$$

We require that $E_{\mathrm{TH}} \geq E_x^W$, $0 \leq x \leq e_W$. Therefore we obtain

$$\mathcal{E}(W) = \min \left\{ E_{\mathrm{TH}} e^{\frac{x}{\tau}} - \int_0^x P(W,t) e^{\frac{t}{\tau}} dt \mid 0 \leq x \leq e_w \right\}.$$

Having derived the tolerable energy level, we would like to calculate the cool-down time, $C(W)$, for dwell W. The cool-down time of a dwell is the time needed for the radar system to cool from energy level E to energy level $\mathcal{E}(W)$. The cooling process follows an exponential energy loss model. $C(W)$ is such that

$$E e^{-\frac{C(W)}{\tau}} = \mathcal{E}(W).$$

From this we obtain the cool-down time as

$$C(W) = \max\left\{\left\lceil -\tau \ln \frac{\mathcal{E}(W)}{E}\right\rceil, 0\right\}.$$

Acknowledgment We thank the anonymous reviewers for their feedback which greatly improved the content and the presentation of this article. This work was supported in part by the ONR MURI program under grant N00014-01-0576, in part by the ONR under grant N0004-02-0102, in part by the Lockheed Martin Corporation under grant 1-5-36137, in part by the NF under grant CCR-0237884, and in part by grant NSC 93-2213-E-002-090.

References

Baugh RA (1973) Computer control of modern radars. RCA Corporation, New York

Bettati R (1994) End-to-end scheduling to meet deadlines in distributed systems. PhD thesis, Department of Computer Science, University of Illinois at Urbana-Champaign

Baptiste P, Le Pape C, Nuitjen W (2001) Constraint-based scheduling: Applying constraint programming to scheduling problems. International series in operations research and management science. Kluwer.

Dong L (2001) Template-based scheduling algorithms for real-time tasks with distance constraints. PhD thesis, University of Pittsburgh

Dong L, Melhem RG, Mossè D (1998) Time slot allocation for real-time messages with negotiable distance constrains. In Proceedings of the IEEE Real-Time Technology and Application Symposium, pp 131–136

Garey MR, Johnson DS (1979) Computers and intractability: A guide to the theory of NP-completeness. W.H. Freeman

Ghosh S, Rajkumar R, Hansen J, Lehoczky J (2004) Integrated resource management and scheduling with multi-resource constraints. In Proceedings of the IEEE Real-Time Systems Symposium, pp 12–22

Gopalakrishnan S, Shih C-S, Ganti P, Caccamo M, Sha L, Lee C-G (2004) Radar dwell scheduling with temporal distance and energy constraints. In International Radar Conference

Hansen J, Ghosh S, Rajkumar R, Lehoczky J (2004) Resource management of highly configurable tasks. In Workshop on Parallel and Distributed Real-Time Systems

Han C-C, Lin K-J (1992) Scheduling distance-constrained real-time tasks. In Proceedings of the IEEE Real-Time Systems Symposium, pp 300–308

Hsueh C-W, Lin K-J, Fa N (1995) Distributed pinwheel scheduling with end-to-end timing constraints. In Proceedings of the IEEE Real-Time Systems Symposium

Han C-C, Lin K-J, Hou C-J (1996) Distance-constrained scheduling and its applications to real-time systems. IEEE Transaction on Computers 45(7):814–826

Jeffay K, Stanat DF, Martel CU (1991) On non-preemptive scheduling of periodic and sporadic tasks. In Proceedings of the IEEE Real-Time Systems Symposium, pp 129–139

Kuo T-W, Chao Y-S, Kuo C-F, Chang C, Su Y-L (2002) Real-time dwell scheduling of component-oriented phased array radars. In Proceedings of the IEEE 2002 Radar Conference, pp 92–97

Kirschmann R, ed. (1998) High-temperature electronics. Wiley-IEEE Press

Lee C-G, Kang P-S, Shih C-S, Sha L (2003) Radar dwell scheduling considering physical characteristics of phased array antenna. In Proceedings of the IEEE Real-Time Systems Symposium

Liu CL, Layland J (1973) Scheduling algorithms for multiprogramming in a hard real-time environment. Journal of the ACM 20(1):46–61

Raemer HR (1996) Radar system principles. CRC Press

Shih C-S, Gopalakrishnan S, Ganti P, Caccamo M, Sha L (2003a) Scheduling real-time dwells using tasks with synthetic periods. In Proceedings of the IEEE Real-Time Systems Symposium

Shih C-S, Gopalakrishnan S, Ganti P, Caccamo M, Sha L (2003b) Template-based real-time dwell scheduling with energy constraint. In Proceedings of the IEEE Real-Time and Embedded Technology and Applications Symposium

Shih C-S, Ganti P, Gopalakrishnan S, Caccamo M, Sha L (2004) Synthesizing task periods for dwells in multi-function phased array radars. In Proceedings of the IEEE Radar Conference

Sprunt B, Sha L, Lehoczky J (1989) Aperiodic task scheduling for hard-real-time systems. Real-time Systems Journal

Sun J (1997) Fixed-priority end-to-end scheduling in distributed real-time systems. PhD thesis, Department of Computer Science, University of Illinois at Urbana-Champaign

Van Hentenryck PV, Deville Y (1991) Operational semantics of constraint logic programming over finite domains. In Proceedings of the International Symposium on Programming Language Implementation and Logic Programming, pp 395–406. Springer-Verlag

Varga A (2000) OMNeT++ 2.0: Discrete Event Simulation System. Department of Telcommunications (BME-HIT), Technical University of Budapest, Hungary, http://www.hit.bme.hu/phd/vargaa/omnetpp.htm

Sathish Gopalakrishnan is a visting scholar in the Department of Computer Science, University of Illinois at Urbana-Champaign, where he defended his Ph.D. thesis in December 2005. He received an M.S. in Applied Mathematics from the University of Illinois in 2004 and a B.E. in Computer Science and Engineering from the University of Madras in 1999. Sathish's research interests concern real-time and embedded systems, and the design of large-scale reliable systems. He received the best student paper award for his work on radar dwell scheduling at the Real-Time Systems Symposium 2004.

Marco Caccamo graduated in computer engineering from the University of Pisa in 1997 and received the Ph.D. degree in computer engineering from the Scuola Superiore S. Anna in 2002. He is an Assistant Professor of the Department of Computer Science at the University of Illinois. His research interests include real-time operating systems, real-time scheduling and resource management, wireless sensor networks, and quality of service control in next generation digital infrastructures. He is recipient of the NSF CAREER Award (2003). He is a member of ACM and IEEE.

Chi-Sheng Shih is currently an assistant professor at the Graduate Institute of Networking and Multimedia and Department of Computer Science and Information Engineering at National Taiwan University since February 2004. He received the B.S. in Engineering Science and M.S. in Computer Science from National Cheng Kung University in 1993 and 1995, respectively. In 2003, he received his Ph.D. in Computer Science from the University of Illinois at Urbana-Champaign. His main research interests are embedded systems, hardware/software codesign, real-time systems, and database systems. Specifically, his main research interests focus on real-time operating systems, real-time scheduling theory, embedded software, and software/hardware co-design for system-on-a-chip.

Chang-Gun Lee received the B.S., M.S. and Ph.D. degrees in computer engineering from Seoul National University, Korea, in 1991, 1993 and 1998, respectively. He is currently an Assistant Professor in the Department of Electrical Engineering, Ohio State University, Columbus. Previously, he was a Research Scientist in the Department of Computer Science, University of Illinois at Urbana-Champaign from March 2000 to July 2002 and a Research Engineer in the Advanced Telecomm. Research Lab., LG Information & Communications, Ltd. from March 1998 to February 2000. His current research interests include real-time systems, complex embedded systems, QoS management, and wireless ad-hoc networks. Chang-Gun Lee is a member of the IEEE Computer Society.

Lui Sha graduated with the Ph.D. degree from Carnegie-Mellon University in 1985. He was a Member and then a Senior Member of Technical Staff at Software Engineering Institute (SEI) from 1986 to 1998. Since Fall 1998, he has been a Professor of Computer Science at the University of Illinois at Urbana Champaign, and a Visiting Scientist of the SEI. He was the Chair of IEEE Real Time Systems Technical Committee from 1999 to 2000, and has served on its Executive Committee since 2001. He was a member of National Academy of Science's study group on Software Dependability and Certification from 2004 to 2005, and is an IEEE Distinguished Visitor (2005 to 2007). Lui Sha is a Fellow of the IEEE and the ACM.