Utility Accrual Real-Time Scheduling with Probabilistically Assured Timeliness Performance

Peng Li and Binoy Ravindran
ECE Department,Virginia Tech
Blacksburg, VA 24061, USA
{peli2,binoy}@vt.edu

E. Douglas Jensen
The MITRE Corporation
Bedford, MA 01730, USA
jensen@mitre.org

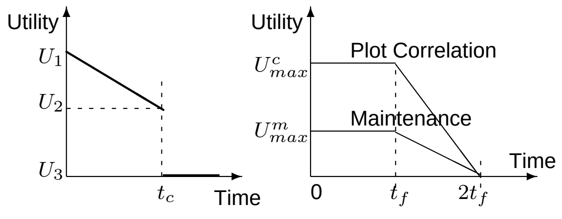
Abstract

We present time/utility function (TUF) algorithms that provide probabilistic assurance on timeliness behavior. A TUF, which is a generalization of the classical deadline constraint, specifies the utility of completing an application activity as a function of that activity's completion time. The algorithms consider a stochastic model where activity execution times and arrivals are probabilistically described. Further, activity time constraints are specified using TUFs. We consider the dual optimization objective of probabilistically satisfying application-specified lower bounds on individual activity utility, and maximizing system-wide total utilities. We present algorithms that achieve this dual objective.

1 Introduction

Dynamic real-time systems such as multi-mode phased array radars [5] and battle management [4] have time constraints which are "soft" (besides those that are hard) in the sense that completing an activity at any time will result in some (positive or negative) utility to the system, and that utility depends on the activity's completion time.

Jensen's time/utility functions [7] (or TUFs) allow the semantics of soft time constraints to be precisely specified. A TUF specifies the utility to the system of completing an application activity as an application- or situation-specific function of when that activity completes.



(a) Track Asso. (b) Corrln. & Maint.

Figure 1: TUFs in MITRE/TOG AWACS and GD/CMU Air Defense

Figures 1(a) and 1(b) show time constraints of two dynamic real-time applications specified using TUFs: (1) the AWACS (Airborne WArning and Control System) surveillance mode tracker system [2] built by The MITRE Corporation and The Open Group (TOG); and (2) a coastal air defense system [9] built by General Dynamics (GD) and Carnegie Mellon University (CMU). Figure 1(a) shows the TUF of the *track association* activity of the AWACS and Figure 1(b) shows the TUFs of two activities of the air defense system called *plot correlation* and *track maintenance*.

Note that the classical deadline is a binary-valued, downward "step" shaped TUF.

When time constraints are expressed with TUFs, the scheduling criteria are based on maximizing accrued utility from those activities—e.g., maximizing the sum, or the expected sum, of the activities' attained utilities. Such criteria are called *Utility Accrual* (or UA) criteria, and sequencing (scheduling, dispatching) algorithms that consider UA criteria are called UA sequencing algorithms.

Most existing UA scheduling algorithms provide assurances on timeliness behavior for some special cases, such as optimal timeliness during under-load situations for step TUFs [3, 8, 13] or assured lower bounds on accrued utilities for deterministic task arrival models [12].

The problem of performance assurance is complicated by the fact that the systems of interest are dynamic and they cannot be described using deterministic task models. For example, task execution times in [2, 9] are highly context dependent, and therefore worst-case execution time (WCET) analysis is infeasible, or is too pessimistic to be useful. Furthermore, task arrivals do not have known minimum inter-arrival times.

In this paper, we consider the problem of achieving performance assurance in dynamic real-time systems that have TUF time constraints. To better account for non-determinism in task execution times and arrival patterns, we stochastically describe those properties. For such a model, we consider the twofold objective of probabilistically satisfying lower bounds on individual task utilities, and maximizing the sum of the utilities. We present resource allocation and scheduling algorithms that achieve this dual objective.

Thus, the contribution of the paper is our algorithms that achieve this dual objective for stochastically described task execution times and arrivals. We are not aware of any other efforts that have studied this problem.

To present the algorithms, we first introduce our models and state our objectives in Section 2. In Sections 3, 4, and 5, we present the resource allocation algorithms. We describe our UA scheduling algorithm in Section 6. Finally, we conclude the paper in Section 7.

2 Models and Objectives

In a system of interest, each task T_i is a sequential execution of a segment of code. We call each instance of a task T_i a job, denoted as $J_{i,j}, j \geq 1$.

We describe task arrivals using a generalized model of the unimodal arrival model [6], called Probabilistic Unimodal Arrival Model (PUAM). A PUAM specification is a tuple of $\langle p(k), w\rangle, \forall k \geq 0$, where $p(k)$ is the probability of k arrivals during any time interval w. Note that $\sum_{k=0}^{\infty} p(k) = 1$. Poisson distributions $\mathcal{P}(\lambda)$ and Binomial distributions $\mathcal{B}(n, \theta)$ are commonly used arrival distributions. Furthermore, the standard unimodal arrival model, periodic arrival model, and sporadic arrival models are special cases of our PUAM model. Besides task arrival patterns, task execution times are also described using application-specified non-negative random variables e.g., gamma distributions.

We focus on non-increasing TUFs, as they encompass the majority of time constraints in many applications. The TUF for task T_i is denoted as $u_i(t)$. For each task, a user also specifies the assurance requirement as a tuple $\langle AU_i, AP_i\rangle$. AU_i is the desired utility that T_i needs to accrue and AP_i is the probability that T_i accrues at least the utility of AU_i.

As introduced previously, our objective is two-fold: (1) satisfy all $\langle AU_i, AP_i\rangle$ if it is possible; and (2) maximize the the sum of utilities accrued by all tasks.

3 Solution Approach

For non-increasing TUFs, satisfying a designated AU_i requires that the task's sojourn time is upper bounded by a "critical time" (CT_i). Given a desired assurance utility AU_i, CT_i satisfies that $\forall t_1 \leq CT_i, u_i(t_1) \geq AU_i$ and $\forall t_2 > CT_i, u_i(t_2) < AU_i$. Once the requirement of accruing AU_i is converted to bounding task sojourn time by CT_i, a probabilistic feasibility analysis similar to that for deadlines can be conducted.

We consider the processor demand analysis approach [1]. The key to using processor demand approach here is allocating a portion of processor bandwidth to each task. We first define *processor bandwidth*:

Definition 1. If a task has a processor bandwidth ρ, then it receives at least ρL processor time during any time interval of length L.

Once a task T_i is allocated a processor bandwidth of ρ_i, jobs of task T_i execute on a "virtual processor" that is not affected by the behavior of other tasks. Therefore, highly dynamic and efficient UA scheduling can be performed on these jobs. The definition of processor bandwidth naturally lends it to proportional share (PS). In this work, we assume a PS has a maximal lag Q. That is, a task will receive at least $(\rho_i L - Q)$ processor time during any time interval of length L. In [10], the authors establish the conversions among assurances provided by PS and resource reservations. Therefore, we focus on resource allocation and scheduling in an abstract level hereafter.

Theorem 1. *Suppose there are at most k arrivals of a task T during any time window of length w and all jobs of T have identical relative critical time D. Then, all job critical times can be satisfied if the underlying PS algorithm provides T with at least a processor bandwidth of $\rho = \max\{(C+Q)/D, C/w\}$, where C is the total execution time of k jobs released by T in a time window of w, and Q is the maximal lag of the PS algorithm.*

Proof. Let $C_p(0, L)$ be the processor demand and $S_p(0, L)$ be the available processor time for task T_i on a time interval of $[0, L]$, respectively. The necessary and sufficient condition for satisfying job critical times is

$$S_p(0, L) \geq C_p(0, L), \forall L > 0 \qquad (1)$$

Let ρ be the processor bandwidth allocated to T. Thus, $S_p(0, L) = \rho L - Q$. Furthermore, the total amount of processor time demand on $[0, L]$ is $C_p(0, L) = \left(\left\lfloor (L-D) \big/ w \right\rfloor + 1\right) C$. Therefore, Equation 1 can be rewritten as

$$\rho L - Q \geq \left(\left\lfloor (L-D) \big/ w \right\rfloor + 1\right) C, \forall L > 0 \qquad (2)$$

Since $\left(\left\lfloor \frac{L-D}{w} \right\rfloor + 1\right) \leq \left((L-D)/w + 1\right)$, it is sufficient to have $\rho L - Q \geq \left(\frac{L-D}{w} + 1\right) C, \forall L > 0$. This leads to

$$\rho \geq \frac{C}{w} + \frac{1}{L}\left(C + Q - C\frac{D}{w}\right), \forall L > 0 \qquad (3)$$

It is easy to see that ρ is monotone of L. For a positive $C + Q - C\frac{D}{w}$, the maximal ρ occurs when $L = D$, which yields $\rho = (C+Q) \big/ D$. For a negative $C + Q - C\frac{D}{w}$, the maximal ρ occurs when $L = \infty$. Combining these two cases, we can prove this theorem. $\square$

For simplicity, we only consider the case $\rho \geq (C + Q)/D$, which implies $D < w$. Furthermore, note that critical sections in a PS algorithm can be handled by setting Q the longest critical section of all tasks.

Let N_i be the random variable for the number of arrivals during a time window of w_i. Then, the processor demand of task T_i during a time window of w_i is $C_i = \sum_{j=1}^{N_i} c_{i,j}$, where $c_{i,j}$ is the execution time of job

$J_{i,j}$. By Theorem 1, $\rho_i \geq (C_i + Q)/CT_i$, where CT_i is the critical time of task T_i. To satisfy the assurance probability, we require

$$\Pr\left[\sum_{j=1}^{N_i} c_{i,j} \leq \rho_i CT_i - Q\right] \geq AP_i \tag{4}$$

The above condition is the fundamental bandwidth requirement for satisfying a task's critical time. If $N_i = k$, the total processor time demand during a time window becomes $\sum_{j=1}^{k} c_{i,j}$. Therefore, Equation 4 can be rewritten as a sum of conditional probabilities:

$$\sum_{k=0}^{\infty}\left(p_i(k) \times \Pr\left[\sum_{j=1}^{k} c_{i,j} \leq \rho_i CT_i - Q\right]\right) \geq AP_i \tag{5}$$

4 A General Solution

The feasibility condition (Equation 4) can be rewritten as:

$$1 - \Pr\left[C_i \geq \rho_i CT_i - Q\right] \geq AP_i \tag{6}$$

By Markov's Inequality, $\Pr[X \geq t] \leq E(X)/t$ for any non-negative random variable. Therefore,$1 - \Pr[C_i \geq \rho_i CT_i - Q] \geq 1 - E(C_i)/(\rho_i CT_i - Q)$. If we can determine a ρ_i so that $1 - E(C_i)/(\rho_i CT_i - Q) \geq AP_i$, $\Pr[C_i \leq \rho_i CT_i - Q] \geq AP_i$ is also satisfied. This becomes

$$\rho_i \geq \frac{E(C_i)}{CT_i\,(1 - AP_i)} + \frac{Q}{CT_i} \tag{7}$$

Note that N_i in Equation 4 is a random variable and follows a distribution specified by $p_i(a)$. By Wald's Equation, $E(C_i) = E\left(\sum_{j=1}^{N_i} c_{i,j}\right) = E(c_i)E(N_i)$. Thus,

$$\rho_i \geq \frac{E(c_i)E(N_i)}{CT_i\,(1 - AP_i)} + \frac{Q}{CT_i} \tag{8}$$

This solution is applicable for any distributions of c_i and N_i, and only requires the average number of arrivals and the average execution time.

5 A Binary Search Strategy

The previous section assumes minimal information regarding task arrivals and execution times. Therefore, the solution in Equation 8 may be pessimistic for some distributions. This section presents a binary search strategy that demands and utilizes the information of full distributions of task arrivals and execution times.

For brevity, we introduce a shorthand notation for the left hand side of Equation 5. Let $feasibleProb_i(\rho_i) = \sum_{k=0}^{\infty}\left(p_i(k) \times \Pr\left[\sum_{j=1}^{k} c_{i,j} \leq \rho_i CT_i - Q\right]\right)$. This function calculates the probability of satisfying all job critical times if task T_i were assigned a processor bandwidth of ρ_i.

Then, Equation 5 can be restated as solving the minimal ρ_i that satisfies $feasibleProb_i(\rho_i) \geq AP_i$. Notice that function $feasibleProb_i(\rho_i)$ is a monotone in terms of ρ_i. Therefore, a binary search is applicable on the set of $\rho_i \in [0, 1]$.

The binary search strategy, presented in Algorithm 1 works as follows: The algorithm accepts an error bound ϵ, a range to search the minimal bandwidth, denoted as $[a, b]$, and the required assurance probability AP_i. In our case, invoking $minBW(\epsilon, 0, 1, AP_i)$ returns either the minimal required ρ_i, or **failure** if even the maximal processor bandwidth i.e., $\rho_i = 1$, cannot satisfy AP_i. In the worst case, function $minBW(\epsilon, a, b, AP_i)$ performs $\log_2((b - a)/\epsilon)$ searches. If $a = 0$ and $b = 1$, the worst case complexity of $minBW(\epsilon, a, b, AP_i)$ becomes $\log_2(1/\epsilon)$.

```
Input   : $\epsilon, a, b, AP_i$
Output  : the minimal $\rho_i$ that satisfies
          $feasibleProb(\rho_i) \geq AP_i$ for
          task $T_i$; failure if even the maximal
          bandwidth $b$ cannot satisfy $AP_i$
if $feasibleProb_i(b) < AP_i$ then
    return failure;
if $b - a \leq \epsilon$ then
    return $a$;
if $feasibleProb_i((a + b)/2) \geq AP_i$ then
    return minBW ($\epsilon, a, (a + b)/2, AP_i$);
else
    return minBW ($\epsilon, (a + b)/2, b, AP_i$);
```

Algorithm 1: minBW(ϵ, a, b, AP_i) Function

Let $S_k = \sum_{j=1}^{k} c_{i,j}$. Given a task arrival pattern $\langle p_i(a), w_i\rangle$, the key to using Algorithm 1 is to calculate the sum distribution $\Pr\left[S_k \leq \rho_i CT_i\right]$. Computing the sum distribution of a set of independent random variables, in general, requires convolutions. In practice, convolutions are performed for small k. When k is large, the sum distribution can be approximated by the Central Limit Theorem (CLT), regardless of the original distribution of job execution times. The Central Limit Theorem states that when k is large, S_k converges to a normal distribution. Furthermore, if $E\left(|c_i - E(c_i)|^3\right) < \infty$, the error of using CLT to approximate the sum distribution is bounded by the Berry-Esséen Theorem.

6 A UA Scheduling Algorithm

The EDF optimality is only meaningful for satisfying job critical times—it does not maximize accrued utilities, which is one of our objectives. Therefore, we develop a UA job scheduling algorithm, called UJSsched, that possesses the following properties:

- If all job critical times can be satisfied by EDF, then `UJSsched` should be able to do so and accrue at least the same utility as EDF does; and
- In case that not all jobs critical times can be satisfied, `UJSsched` seeks to accrue as much utility as possible.

We desire a fast job scheduling algorithm. This is particularly true in the context of PS, where the PS mechanism itself may be implemented by another scheduling algorithm, such as EEVDF [11]. Thus, we adopt the Highest Utility Density First (HUDF) heuristic as a way to improve accrued utility. Our rational for doing so is because HUDF is easy to implement, incurs small overhead, and exhibits high performance during both underloads and overloads.

Let t_0 be the time instant when a scheduling event occurs. The Utility Density (UD) of a job J is defined as the ratio of its utility at predicted completion time over its remaining execution time. That is, $UD = u\left(t_0 + c(t_0)\right)/c(t_0)$, where $u()$ is the TUF of job J and $c(t_0)$ is J's remaining execution time at instant t_0.

The design of the `UJSshced` algorithm closely follows its desired properties. As shown in Algorithm 2, the algorithm first examines if the set of ready jobs are schedulable under both EDF and HUDF, assuming jobs are executed on a slow processor with an execution rate of ρ_i. If they are, the highest utility density job is selected. If the jobs are only schedulable under EDF, then the earliest critical time job should be executed. In case that the jobs are not schedulable, the next job is selected by HUDF.

```
Input  : a queue of ready jobs, denoted as $RQ$
Output : job $J_s$ to be executed next; or $NULL$
if $RQ$ is empty then
    Select $NULL$ job;
Let $U_d$ be the accrued utilities of all jobs in $RQ$ under EDF;
Let $U_h$ be the accrued utilities of all jobs in $RQ$ under Highest Utility Density First (HUDF);
if $RQ$ is feasible under EDF then
    if $RQ$ is feasible under HUDF and $U_h > U_d$ then
        Select the highest utility density job;
    else
        Select the earliest critical time job;
else
    Select the highest utility density job;
```

Algorithm 2: `UJSsched` Algorithm

Given n jobs in the ready queue, the worst-case complexity of `UJSsched` is $O(n)$. Further, `UJSsched` has the following important property:

Lemma 2. *If all job critical times can be satisfied by EDF, then `UJSsched` is able to do so and accrues at least the same utility as EDF does.*

This lemma directly follows Algorithm 2.

7 Conclusions, Future Work

This paper presents resource allocation and scheduling algorithms for real-time systems with TUF time constraints. The algorithms consider the two-fold objective of probabilistically satisfying lower bounds on task-level accrued utilities and maximizing system-level total accrued utilities, for a stochastic task model.

There are several directions for future research. An important problem is to probabilistically satisfy lower bounds on task-level *and* system-level accrued utilities. Particularly, a tradeoff mechanism is desired when system-level assurance requirement contradicts task-level requirements and vice versa.

References

[1] S. K. Baruah, L. E. Rosier, and R. R. Howell. Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor. *Real-Time Systems*, 2(4):301–324, November 1990.

[2] R. Clark et al. An adaptive, distributed airborne tracking system. In *IEEE WPDRTS*, pages 353–362, April 1999.

[3] R. K. Clark. *Scheduling Dependent Real-Time Activities*. PhD thesis, Carnegie Mellon University, 1990.

[4] GlobalSecurity.org. Bmc3i battle management, command, control, communications and intelligence. `http://www.globalsecurity.org/space/systems/bmc3i.htm/`.

[5] GlobalSecurity.org. Multi-platform radar technology insertion program. `http://www.globalsecurity.org/intell/systems/mp-rtip.htm/`.

[6] J.-F. Hermant and G. L. Lann. A protocol and correctness proofs for real-time high-performance broadcast networks. In *IEEE ICDCS*, pages 360–369, 1998.

[7] E. D. Jensen, C. D. Locke, and H. Tokuda. A time-driven scheduling model for real-time systems. In *IEEE RTSS*, pages 112–122, December 1985.

[8] C. D. Locke. *Best-Effort Decision Making for Real-Time Scheduling*. PhD thesis, Carnegie Mellon University, 1986.

[9] D. P. Maynard et al. An example real-time command, control, and battle management application for alpha. Technical Report TR-88121, Carnegie Mellon University, 1988.

[10] J. Regehr and J. A. Stankovic. HLS: A framework for composing soft real-time schedulers. In *IEEE RTSS*, pages 3–14, London, UK, Dec. 2001.

[11] I. Stoica, H. A.-Wahab, et al. A proportional share resource allocation algorithm for real-time, time-shared systems. In *IEEE RTSS*, pages 288–299, 1996.

[12] H. Wu, B. Ravindran, et al. Energy-efficient, utility accrual scheduling under resource constraints for mobile embedded systems. In *ACM EMSOFT*, September 2004.

[13] H. Wu, B. Ravindran, et al. Utility accrual scheduling under arbitrary time/utility functions and multiunit resource constraints. In *IEEE RTCSA*, August 2004.