Novel Algorithms for Open-loop and Closed-loop Scheduling of Real-time Tasks in Multiprocessor Systems Based on Execution Time Estimation [*]

R. Al-Omari
Processor Performance Group
IBM Austin, TX
alomari@us.ibm.com

G. Manimaran M. V. Salapaka Arun K. Somani
Dept. of Electrical and Computer Engineering
Iowa State University, Ames, IA 50011, USA
gmani@iastate.edu murti@iastate.edu arun@iastate.edu

Abstract

Most dynamic real-time scheduling algorithms are open-loop in nature meaning that they do not dynamically adjust their behavior using the performance at run-time. When accurate workload models are not available, such a scheduling can result in a highly underutilized system based on an extremely pessimistic estimation of workload. In recent years, "closed-loop" scheduling is gaining importance due to its applicability to many real-world problems wherein the feedback information can be exploited efficiently to adjust system parameters, thereby improving the performance.

In this paper, we first propose an open-loop dynamic scheduling algorithm that employs overlap in order to provide flexibility in task execution times. Secondly, we propose a novel closed-loop approach for dynamically estimating the execution time of tasks based on both deadline miss ratio and task rejection ratio. This approach is highly preferable for firm real-time systems since it provides a firm performance guarantee. We evaluate the performance of the open-loop and the closed-loop approaches by simulation and modeling. Our studies show that the closed-loop scheduling offers a significantly better performance (20% gain) over the open-loop scheduling under all the relevant conditions we simulated.

Keywords: Real-time scheduling, Feedback control, Open-loop scheduling, Closed-loop scheduling, Modeling, Multiprocessor systems.

1 Introduction and Motivation

Most real-time scheduling algorithms are based on estimations of the worst-case execution time *(WCET)* of tasks. In practice, it is very difficult to obtain a tight bound on *WCET*, and very few timing analysis tools are available [1]. Also, the computer market is dominated by general purpose computing where the average case performance matters the most. As a result, obtaining a tight bound on *WCET* is becoming irrelevant. In many cases, it is preferable to base scheduling decisions on average execution time and to deal with bounded transient overloads dynamically. This approach is especially preferable in firm/soft real-time systems as it provides low deadline miss ratio while achieving high utilization and throughput. The requirements of an ideal firm real-time scheduling algorithm are to (1) provide (firm) performance guarantees to admitted tasks, i.e., maintain low deadline miss ratio; and (2) admit as many tasks as possible, i.e., achieve high guarantee ratio.

The rest of the paper is organized as follows. In Section 2, we define the task and scheduler models, terminology, and performance metrics. In Section 3, we propose an open-loop real-time scheduling algorithm. In Section 4, closed-loop scheduling, related work, and related issues are discussed. In Section 5, we propose the closed-loop scheduling approaches, present their modeling, and analyze their performance. In Section 6, we compare the performance of the closed-loop scheduling algorithms through simulation, and validate the results through Matlab Simulink model based experiments. Finally, Section 7 is concluding remarks.

2 System Model

(i) Tasks are aperiodic, i.e., task characteristics are not known a priori. Every task T_i has the following attributes: arrival time (a_i), ready time (r_i), worst-case execution time ($WCET_i$), best-case execution time ($BCET_i$), and firm deadline (d_i); (ii) Tasks are non-preemptable, i.e., when a task starts execution, it finishes to its completion; (iii) $st(T_i)$ is the scheduled start time of task T_i, which satisfies $r_i \leq st(T_i) \leq d_i - EET_i$. $ft(T_i)$ is the scheduled finish time of task T_i, which satisfies $r_i + EET_i \leq ft(T_i) \leq d_i$, where EET_i is the estimated execution time for task T_i; (iv) Earliest start time of a task T_i ($EST(T_i)$) is the earliest

[*] This research was funded in part by the NSF under grant CCR-0098354

0-7695-1926-1/03/$17.00 (C) 2003 IEEE

time it can be executed; (v) In our dynamic multiprocessor scheduling, all the tasks arrive at a central processor called the scheduler, from where they are dispatched to other processors for execution.

2.1 Performance Metrics

Task guarantee ratio (GR): This is the ratio of the number of tasks admitted to the total number of tasks that arrived at the system. The rejection of tasks depends on factors such as schedulability check algorithm, EET, and the time at which the schedulability check is performed. Task rejection ratio is defined as $RR = 1 - GR$.

Deadline hit ratio (HR): This is the ratio of the number of admitted tasks that meet their deadlines to the total number of tasks admitted into the system. Though the schedulability check of tasks are performed while admitting them, tasks can still miss their deadline when their actual execution time (*AET*) is greater than their estimated execution time (*EET*) or due to unanticipated faults in the system. Deadline miss ratio is defined as $MR = 1 - HR$.

Task effective ratio (ER): This is the ratio of the number of tasks that meet their deadlines to the total number of tasks arrived at the system. $ER = HR * GR$.

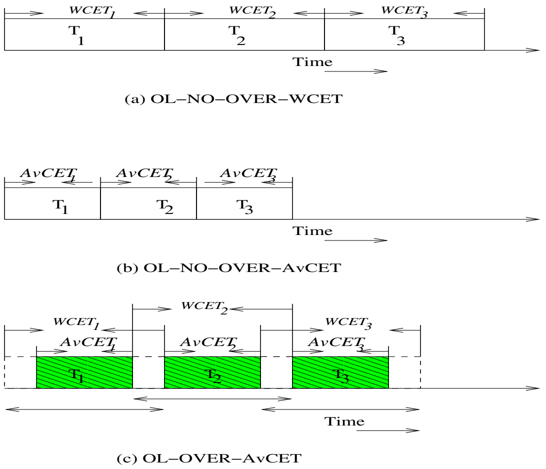
3 Open-Loop Dynamic Planning Scheduling

In order to guarantee that tasks meet their deadlines once they are scheduled, scheduling algorithms schedule tasks based on their worst-case execution time (*WCET*) [2, 3, 4]. In this approach (called *OL-NO-OVER-WCET* (Open-Loop NO OVERlap with *WCET* estimation) scheduling algorithm), *WCET* is used to perform the schedulability check (i.e. it checks whether $EST(T_i)$ + $WCET_i \leq d_i, \forall\ T_i$). Moreover, each task is assigned a time slot equals to its *WCET* without overlapping when the schedule is constructed as shown in Figure 1a. Therefore, in *OL-NO-OVER-WCET*, tasks are allowed to execute to their *WCET* if needed. AET of tasks varies between their *BCET* and *WCET* due to non-deterministic behavior of several low-level processor mechanisms (e.g. caching, prefetching, and DMA data transfer), and also due to the fact that AET are function of system state and input data [5, 1]. A resource reclaiming algorithm can compensate for the performance loss due to the inaccuracy of the estimation of $WCET$ [6, 7]. Resource reclaiming on multiprocessor systems with resource constraints is expensive and complex. Therefore, in a non-hard real-time system it will be more effective to schedule tasks based on their average-case execution time ($AvCET = \lfloor \frac{WCET+BCET}{2} \rfloor$) rather than their *WCET*.

In firm real-time systems, the consequences of not meeting the deadline are not as severe as hard real-time systems. Hence, tasks can be scheduled based on their *AvCET*. This reduces the amount of resources unused. In the second approach (called *OL-NO-OVER-AvCET* (Open-Loop NO OVERlap with *AvCET* estimation) scheduling algorithm), *AvCET* is used to perform the schedulability check (i.e. it checks whether $EST(T_i)$ + $AvCET_i \leq d_i, \forall\ T_i$). Moreover, each task is assigned a time slot equals to its *AvCET* (the tasks are not allowed to overlap) when the schedule is constructed as shown in Figure 1b. In the *OL-NO-OVER-AvCET* approach, the tasks are allowed only to execute to their *AvCET*. Therefore, the system can guarantee more tasks to enhance GR. Scheduling tasks with their *AvCET*, however, increases the chances of them missing their deadlines. Indeed, all tasks that have an *AET* greater than their *AvCET* would miss their deadlines.

3.1 New Open-Loop Firm Scheduling Algorithm

To achieve a GR comparable to *OL-NO-OVER-AvCET* and a MR comparable to *OL-NO-OVER-WCET*, we propose *OL-OVER-AvCET*. *WCET* is used to perform the schedulability check (i.e. it checks whether EST(T_i) + $WCET_i \leq d_i, \forall\ T_i$). Each task is assigned a time slot *AvCET* when the schedule is constructed and it is overlapped with its both neighbors by a time slot $\frac{WCET-AvCET}{2}$ as shown in Figure 1c. In *OL-OVER-AvCET*, the time at which the processor P_j is available for executing a task is $ft(T_i)$ minus the overlap time. The overlapped time is $\frac{WCET_i - AvCETT_i}{2}$. Therefore, a task T_i can start and finish any time within $[EST(T_i),\ EST(T_i) + WCET_i]$. So the task is allowed to execute to its *WCET* if its previous neighbor has completed its execution before the overlapped time.



(a) OL–NO–OVER–WCET

(b) OL–NO–OVER–AvCET

(c) OL–OVER–AvCET

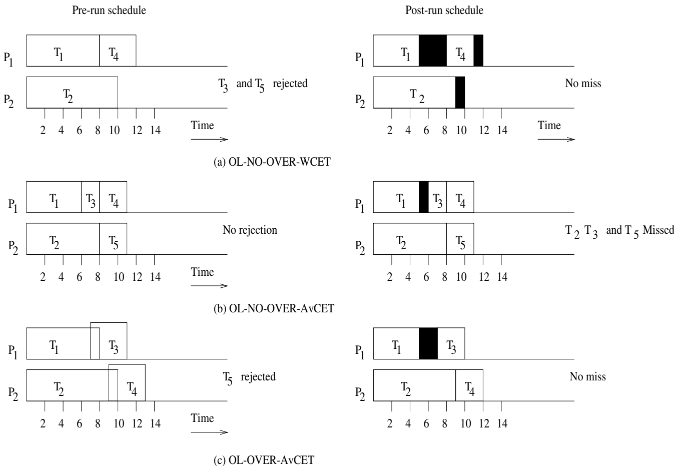
Figure 1. Open-loop scheduling algorithms

Example of open-loop scheduling algorithms

Table 1 gives *WCET*, *BCET*, *AvCET*, *AET*, and the deadlines (d) for five tasks whose ready times are all equal to zero. Tasks do not have any resource requirement. Figure 2 shows the feasible schedules on two processors and the post-run schedules. From Figure 2a, we notice that only three tasks have been accepted (T_1, T_2, and T_4) using the *OL-NO-OVER-WCET* scheduling algorithm. Since the tasks are scheduled using their *WCET*, all accepted tasks have met their deadlines. The processors are idle for 5 time units. From Figure 2b, we notice that all the tasks have been accepted using *OL-NO-OVER-AvCET*. Since the tasks are scheduled using their *AvCET*, only 2 tasks have met their deadlines (T_1, and T_4). The processors are idle only for 1 time unit. From Figure 2c, we notice that four tasks have been accepted (T_1, T_2, T_3, and T_4) using *OL-OVER-AvCET*. Since the tasks are scheduled using the overlap approach, all accepted tasks have met their deadlines. The processors are idle for 2 time units. This example shows the superiority of *OL-OVER-AvCET* over the other two algorithms.

Task	*BCET*	*WCET*	*AvCET*	*AET*	*d*
T_1	4	8	6	5	9
T_2	6	10	8	9	10
T_3	1	4	2	3	11
T_4	2	4	3	3	12
T_5	2	4	3	4	13

Table 1. Parameters for the tasks



(a) OL-NO-OVER-WCET

(b) OL-NO-OVER-AvCET

(c) OL-OVER-AvCET

Figure 2. Pre-run and post-run schedules

4 Closed-loop Scheduling

The disadvantage of basing the schedulability test on a priori estimation is that an underestimation of execution times may jeopardize the correct behavior of the system, whereas an overestimation will under-utilize system resources and cause performance degradation [8]. In this section, a feedback of MR and RR is used to generate an error, which is input to a control unit that adjusts the estimated AET. We model and analyze the closed-loop scheduling algorithm using control theory. This scheduling algorithm accepts significantly more tasks and meets more deadlines than open-loop scheduling algorithms, thereby improving ER.

4.1 Related Work

Very little work has been done in the area of real-time feedback scheduling. In [9] and [10], the authors proposed design tools to consider scheduling at the early stage of control systems design. The scheduling algorithms used are off-line open-loop scheduling algorithms such as RMS and EDF. [11, 12, 13, 14] utilized the flexible timing constraints as a mechanism for graceful performance degradation in control systems. In [11, 12] an elastic task model [15] is used to design an elastic control approach, which integrates the *continuous design with digitization* (CDD) [12] and the *direct digital design* (DDD) [11] with adjustable frequencies. However, all the work assumed the execution times are known and focused on how to reassign the periods to satisfy the utilization constraints. Instead, our work focuses on using feedback loops to maintain satisfactory performance when the task execution times change dynamically.

A notable work in real-time feedback scheduling is [16, 17] where a PID controller is used in the on-line scheduling, called Feedback Control-EDF (FC-EDF). Recently, [8] proposed an approach in which task periods can be dynamically adjusted based on the current load. When the estimated load is greater than a certain threshold, the elastic theory is used to enlarge the task periods to find a feasible configuration. In [17, 8], the authors assume that each task has multiple versions that differ in their execution time, the higher the execution time the better the quality. Instead, our work assumes only one version for each task which is more general and realistic. Moreover, in [17] the feedback mechanism is used to reject tasks in order to keep the total number of missed deadlines below a desired value. Instead, in our work we maximize both GR and HR.

4.2 Dynamic Planning based Closed-Loop Scheduling Framework

To apply feedback control techniques in scheduling, we need to identify the set points, control variables, regulated variables, and measured variables. The principle of feedback is to employ the information gathered through the measured variables (y) that affect the controlled variable (u) to produce the desired effect on the regulated variables. Therefore, the control law K must dictate how to alter u in response to y.

Regulated Variables are quantities used to depict the performance of the system. The choice of the regulated variables depends on the system goal. The performance of a firm real-time system usually depends on: (1) how many tasks it admits (rejects); (2) how many tasks among the accepted tasks meet (miss) their deadlines. Therefore, MR and RR are natural choices.

Control Input are variables that are able to affect the value of the regulated variables. In the non-preemptive multiprocessor firm real-time system, it is widely known that MR and RR highly depend on EET (see the example in Section 3.1). Thus *EET* is an appropriate choice. EET_i of a task (T_i) is calculated as follows:

$$EET_i = AvCET_i + etf_{kT}\,[AvCET_i - BCET_i] \qquad (1)$$

where etf_{kT} is the estimation factor at instant time kT which can have a value in $[-1,\ 1]$. For $etf_{kT} = -1$ $EET = BCET$, for $etf_{kT} = 0$ $EET = AvCET$, for $etf_{kT} = 1$ $EET = WCET$.

In the case of open-loop scheduling algorithms, etf is 0 for the algorithms that use *AvCET* as AET, and it is 1 for the algorithms that use *WCET* as AET. In the case of closed-loop scheduling algorithms etf is used as u.

Exogenous Inputs are variables that effect the system and cannot be changed by the designer. Typically, any reference set point or disturbances form the exogenous signal. MR_s and RR_s are chosen to have a small, but a non-zero value (e.g. $MR_s = RR_s = 1\%$). Note that 0 is not chosen for the set points. A system with $MR_s = 0$ can achieve a 0% MR but causes extremely low utilization by rejecting too many tasks (i.e., overestimate the execution time of tasks). In contrast, $MR_s \neq 0$ will always try to (lightly) overload the system to achieve high utilization. Similarly, a system with $RR_s = 0$ can achieve a 0% RR but causes extremely low throughput by missing too many deadlines (i.e., underestimate the execution time of tasks). In contrast, $RR_s \neq 0$ will always try to (lightly) underload the system to achieve high throughput.

Measured Variables are variables that can be measured and used as feedback information. The instantaneous miss ratio (MR_{kT}) is calculated every sample period (T):

$$MR_{kT} = \frac{\#\ of\ missed\ tasks_{[(k-1)T,\ kT]}}{\#\ of\ finished\ tasks_{[(k-1)T,\ kT]}} \qquad (2)$$

where *k* is the current time instant. In contrast, the average miss ratio (MR) is defined as the time average of MR_{kT} for the entire run-time.

The instantaneous rejection ratio (RR_{kT}) is calculated every sample period (T):

$$RR_{kT} = \frac{\#\ of\ rejected\ tasks_{[(k-1)T,\ kT]}}{\#\ of\ arrived\ tasks_{[(k-1)T,\ kT]}} \qquad (3)$$

In contrast, the average rejection ratio (RR) is defined as the time average of RR_{kT} for the entire run-time.

Actuator Model: The controller computes Δetf_{kT}. Then the execution time estimator updates the estimated factor:

$$etf_{kT} = etf_{(k-1)T} - \Delta etf_{kT} \qquad (4)$$

Control Law: The proportional control formula is:

$$\Delta etf_{kT} = K \times error_{kT} \qquad (5)$$

where K and T are a tunable parameters. K is the coefficient of the controller and T is the sample period. The tuning of these parameters will be discussed later. Different controller types can be used (e.g. PID-controller as in [17]). The controller maps the performance of the accepted tasks (i.e., error) to the change in EET so as to drive the system performance back to the set point. The system performance is fed back to the controller every T second.

Uncertainty Model Since there is no mathematical system that can precisely model a scheduling algorithm, uncertainty is inevitable. Uncertainty can result because we cannot predict the outputs even when we know the inputs, and/or when the inputs are unpredictable. We denote the uncertainty due to error in modeling by Δ.

In Section 5.2, we approximate the relation between etf and RR and MR as a linear line ($\Delta = 0$). The modeling of uncertainty will be addressed in our future work.

5 Closed-loop Scheduling Algorithms

In this section, we propose an algorithm, called closed-loop overlap dynamic scheduling algorithm (CL-OVER), which integrates feedback controller with the open-loop overlap scheduling algorithm in Section 3.1. The loop is closed by feeding back *(i)* MR in the first approach which is called *CL-OVER-MISS*; *(ii)* RR in the second approach which is called *CL-OVER-REJ*; *(iii)* both MR and RR in the final approach which is called *CL-OVER-MISSREJ*.

5.1 Architecture of CL-OVER Approaches

The *CL-OVER* schedulers, as shown in Figure 3, consist of a controller, an execution time estimator and an overlap scheduler. In these approaches MR_{kT} and/or RR_{kT} are periodically fed back to the controller. The controller computes Δetf and then calls the execution time estimator to change EET. The overlap scheduler schedules the arrived tasks according to their EET.

The control formula used by the *CL-OVER-MISS* (*CL-OVER-REJ*) controller, as shown in Figure 3(a)(b), is:

$$\Delta etf_{kT}^{m(r)} = K_{m(r)} \times error_{kT}^{m(r)} \qquad (6)$$

where $K_{m(r)}$ is the coefficient of the controller, $\Delta etf_{kT}^{m(r)}$ is due to the error in MR (RR), and $error_{kT}^{m(r)}$ is the difference between MR_{kT} (RR_{kT}) and MR_s (RR_s) at time instant *k*, i.e., $error_{kT}^{m} = MR_s - MR_{kT}$, and $error_{kT}^{r} = RR_s - RR_{kT}$. This approach can only control MR (RR) and may result in rejecting too many tasks (missing too many deadlines) in order to keep MR (RR) equal to MR_s (RR_s).

In order to control both MR and RR (enhance ER), we propose *CL-OVER-MISSREJ* in which both MR and RR are fed back to the controller. The control formula used by *CL-OVER-MISSREJ*, as shown in Figure 3c, is:

$$\Delta etf_{kT}^{mr} = nf_{m/r} \times K_{mr} \times error_{kT}^{m} - nf_{r/m} \times K_{mr} \times error_{kT}^{r} \quad (7)$$

where K_{mr} is the coefficient of the controller, Δetf_{kT}^{mr} is due to the error in both MR and RR, $nf_{m/r}$ ($nf_{r/m}$) is normalization factor used to normalize Δetf_{kT}^{m} (Δetf_{kT}^{r}). The idea behind using these normalization factors is that the sensitivity of MR_{kT} to Δetf_{kT} is different from the sensitivity of RR_{kT} to the same variation. The value of $nf_{m/r}$ and $nf_{r/m}$ will be discussed in the next section.

This approach tries to maximize ER. Note that we did not feedback ER or $1 - ER$ to control ER directly. This is because there is no defined actuators that can be used to change etf to control ER to the desired value.

5.2 Modeling of CL-OVER Approaches

Before presenting the models, we define the following notions in the Z-transform:

1. T: constant sampling period, which is the time elapsed in interval $[kT,\ (k+1)T]$, k being time instants.

2. $MR(z)$ and $RR(z)$: the miss ratio and the rejection ratio respectively (system outputs and controlled variables).

3. MR_s and RR_s: set points (miss and rejection ratio).

4. $etf(z)$: the estimated factor (control variable).

5. $\Delta etf(z)$: the change in the estimated factor ($etf(z)$).

6. mgf and rgf: the gain that maps the estimated factor to the miss ratio and the rejection ratio respectively.

7. mdf and rdf: the disturbance associated with the miss ratio and the rejection ratio respectively which are resulting from the system bounds (e.g., $MR(z) \in [0,1]$, $RR(z) \in [0,1]$, $etf \in [-1,\ 1]$, and processor utilization bound).

8. $nf_{m/r}$ and $nf_{r/m}$: normalization factors that are used in CL-OVER-MISSREJ to normalize the errors generated from the miss ratio and the rejection ratio respectively. The values for these normalization factors are $nf_{m/r} = \frac{mgf}{rgf}$ and $nf_{r/m} = \frac{rgf}{mgf}$.

Using the above notations and Equation (4), the Z-transform for the estimated factor follows the equation

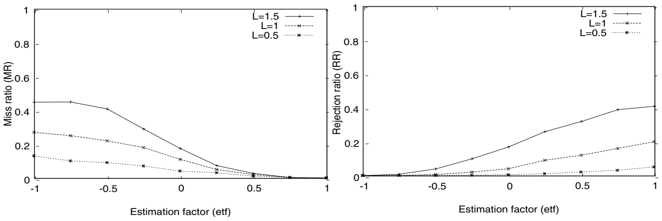
$$etf(z) = \frac{z\Delta etf}{1-z} \quad (8)$$

We can derive MR (RR) based on the correlation between $MR(z)$ ($RR(z)$) and $etf(z)$:

$$MR(z) = -mgf \times etf(z) + mdf \quad (9)$$

$$RR(z) = rgf \times etf(z) + rdf \quad (10)$$

mgf (rgf) and mdf (rdf) vary with the scheduling algorithm, the system load, and the system parameter (e.g., number of processors). The values of these factors can be found by studying the relation between etf and MR (RR) using the simulation model.



(a) MR as etf varies (b) RR as etf varies

Figure 4. MR **and** RR **as** etf **varies**

In Equations (9) and (10), we approximate the relation between etf and RR and MR as a linear line. Figure 4 shows how MR and RR vary as etf changes for different task loads L in the overlap scheduling algorithm.

These figures have been generated using the simulation model (see Section 6). For each point in Figures 4, the system was simulated for 10,000 tasks, which have been chosen to have a 98% confidence interval within ±0 0017, and ±0 0022 around each value of MR, and RR respectively.

From the figure, we notice that the relation between etf and both RR and MR are not linear. Figure 4a shows that MR starts from a maximum value when $etf = -1$ and reduces quadratically as etf increases. For all L that are plotted MR reaches zero when ($etf > 0.75$). Figure 4b shows that RR starts from approximately zero when $etf = -1$ and increases quadratically to a maximum value as etf increases to 1. We also notice that the relation between etf and both RR and MR varies as L varies.

5.3 Analysis of CL-OVER Approaches

In this section, we compare the performance of the closed-loop scheduling algorithms (*CL-OVER-MISS, CL-OVER-REJ*, and *CL-OVER-MISSREJ*) using the transfer function for the block diagrams in Figure 5. The steady state effective ratio (ER_f) is used as the performance metric.

From Figure 5a and the inverse Z-transform the miss ratio and the rejection ratio in the discrete time domain are:

$$MR_{kT} = \begin{cases} 0 & \text{for } kT < 0 \\ \frac{mdf - MR_s}{1+G_m}\left(\frac{1}{1+G_m}\right)^{kT} + MR_s & \text{for } kT \geq 0 \end{cases} \quad (11)$$

$$RR_{kT} = \begin{cases} 0 & \text{for } kT < 0 \\ \frac{(MR_s - mdf)\ nf_{r/m}}{1+G_m}\left(\frac{1}{1+G_m}\right)^{kT} & \\ +(mdf - MR_s)\ nf_{r/m} + rdf & \text{for } kT \geq 0 \end{cases} \quad (12)$$

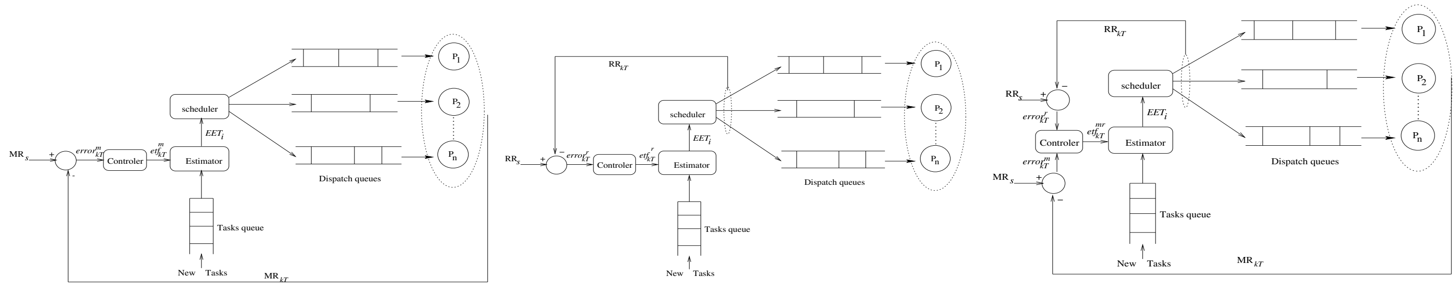
From these equations, we notice that the steady state values for miss ratio and rejection ratio (MR_f and RR_f) in the *CL-OVER-MISS* are MR_s and $(mdf - MR_s)\ nf_{r/m} + rdf$, respectively. Therefore, ER_f is:

$$ER_f = (1 - MR_s) \times (1 - rdf - (mdf - MR_s)\ nf_{r/m}) \quad (13)$$

For *CL-OVER-REJ* (Figure 5b), RR_{kT} and MR_{kT} are:

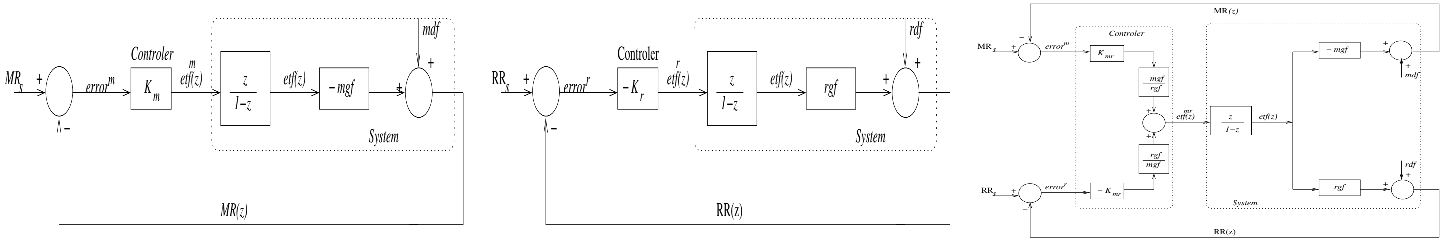
$$RR_{kT} = \begin{cases} 0 & \text{for } kT < 0 \\ \frac{rdf - RR_s}{1+G_r}\left(\frac{1}{1+G_r}\right)^{kT} + RR_s & \text{for } kT \geq 0 \end{cases} \quad (14)$$

$$MR_{kT} = \begin{cases} 0 & \text{for } kT < 0 \\ \frac{(RR_s - rdf)\ nf_{m/r}}{1+G_r}\left(\frac{1}{1+G_r}\right)^{kT} & \\ +(rdf - RR_s)\ nf_{m/r} + mdf & \text{for } kT \geq 0 \end{cases} \quad (15)$$



(a) CL-OVER-MISS (b) CL-OVER-REJ (c) CL-OVER-MISSREJ

Figure 3. Architecture of CL-OVER approaches



(a) CL-OVER-MISS (b) CL-OVER-REJ (c) CL-OVER-MISSREJ

Figure 5. Block diagram of the CL-OVER approaches

These equations show that ER_f in *CL-OVER-REJ* is:

$$ER_f = (1 - RR_s) \times (1 - mdf - (rdf - RR_s) nf_{m/r}) \qquad (16)$$

The rejection ratio (RR_{kT}) and the miss ratio (MR_{kT}) for *CL-OVER-MISSREJ* (Figure 5c) are:

$$MR_{kT} = \begin{cases} 0 & \text{for } kT < 0 \\ \left(\frac{(RR_s - rdf)\ nf_{m/r}\ G_r + (mdf - MR_s)\ G_m}{G_{mr}\ (1+G_{mr})}\right) \times \left(\frac{1}{1+G_{mr}}\right)^{kT} + \frac{MR_s\ G_m + \left((rdf - RR_s)\ nf_{m/r} + mdf\right)\ G_r}{G_{mr}} & \text{for } kT \geq 0 \end{cases} \qquad (17)$$

$$RR_{kT} = \begin{cases} 0 & \text{for } kT < 0 \\ \left(\frac{(MR_s - mdf)\ nf_{r/m}\ G_m + (rdf - RR_s)\ G_r}{G_{mr}\ (1+G_{mr})}\right) \times \left(\frac{1}{1+G_{mr}}\right)^{kT} + \frac{RR_s\ G_r + \left((mdf - MR_s)\ nf_{r/m} + rdf\right)\ G_m}{G_{mr}} & \text{for } kT \geq 0 \end{cases} \qquad (18)$$

These equations quantify ER_f as:

$$ER_f = \begin{array}{l} (1 - \frac{MR_s\ G_m + \left((rdf - RR_s)\ nf_{m/r} + mdf\right)\ G_r}{G_{mr}}) \\ \times (1 - \frac{RR_s\ G_r + \left((mdf - MR_s)\ nf_{r/m} + rdf\right)\ G_m}{G_{mr}}) \end{array} \qquad (19)$$

Equations (13), (16), and (19) have been plotted in Figure 6 for different L with $MR_s = RR_s = 0.01$, and $K_m = K_r = K_{mr} = 0.5$. The figures show that *CL-OVER-MISSREJ* offers the highest ER for all L. This is because, in this approach the feedback mechanism and the controller try to adapt, as L varies, EET to maximize the product of HR by GR (i.e., maximize ER). The figure also shows that the *CL-OVER-REJ* approach offers the lowest ER for all L. This is because this approach tries to achieve $RR = 1\%$. This results in significantly decreasing HR as L increases, which degrades ER.

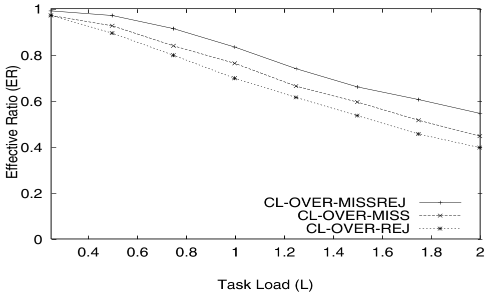


Figure 6. Steady state effective ratio

Table 2. Simulation parameters.

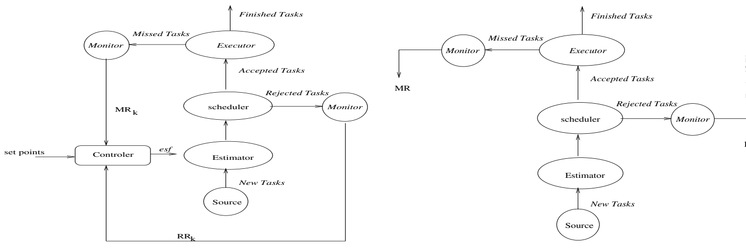
Parameter	Explanation	Values
Min_{BCET}	minimum BCET of tasks	10 sec.
Max_{BCET}	maximum BCET of tasks	20 sec.
f_{WCET}	the factor between BCET and WCET of tasks	4
L	the offered task load to the system	0.25. . . 2
T	sample period	40,80,120,160 sec.
K_m, K_r, K_{mr}	The coefficients of the controllers	0.5 . . . 5
R	laxity parameter	4
N	number of processors	5

6 Simulation Studies

A simulator was developed to evaluate the performance of the open-loop and the closed-loop scheduling algorithms. The parameters used are given in Table 2. The tasks for the simulation are generated as follows:

1. *BCET* of tasks is chosen uniformly between Min_{BCET} and Max_{BCET}.

2. $WCET_i$ is equal to f_{WCET} times its $BCET_i$.



(a) Closed-loop (b) Open-loop

Figure 7. Simulated models

3. $AvCET_i$ of a task (T_i) is equal to $\frac{WCET_i+BCET_i}{2}$.

4. AET_i of task (T_i) is computed as a uniform random variable in $[BCET_i\ WCET_i]$.

5. The firm deadline (d_i) is uniformly chosen between $r_i + 2 \times WCET_i$ and $r_i + R \times WCET_i$, where R $\geq$ 2.

6. The arrival time of tasks follows exponential distribution with mean θ.

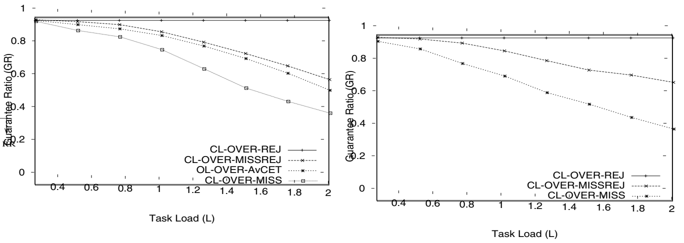
7. The task load L is defined as the expected number of task arrivals per mean service time and its value is approximately equal to $\frac{C}{\theta}$, where C is the mean computation time of the system. The mean computation time C has been calculated based on *AvCET* (i.e., $C = \frac{\frac{1}{n}\sum_{i=1}^{n} AVCET_i}{N}$, where n is the total arriving tasks).

The closed-loop simulator, shown in Figure 7a, has six components: source which generate tasks; an estimator that estimates tasks execution time; a scheduler (overlap scheduling algorithm) that makes admission/rejection decisions on submitted tasks; an executor that models the execution of the tasks on multiprocessor system; monitors that periodically collect the performance statistics; and a controller that compares the feedback MR and/or *RR*, and the set points to calculate the change to be applied to EET. The open-loop simulation, as shown in Figure 7b, does not have the feedback loops.

6.1 Performance of CL-OVER Approaches

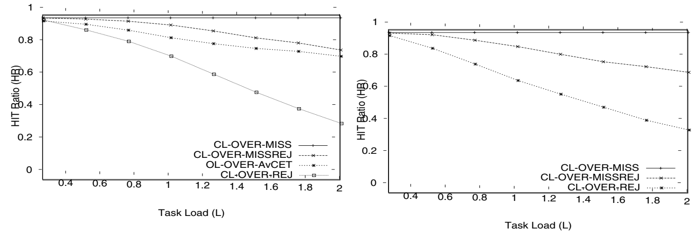
In this section, we compare the three closed-loop scheduling algorithms using the simulation model with *OL-OVER-AvCET*. The average GR, HR, and ER have been used as the performance metrics. For each point in the performance plots (Figures 8a-10a), the system was simulated for 10,000 tasks. This number of tasks has been chosen to have a 97% confidence interval within ± 0.0013, ± 0.0020, and ± 0.0028 around each value of HR, GR, and ER, respectively. In the analytical studies (Figures 8b-10b), we compare the performance of the closed-loop scheduling algorithms using the Matlab Simulink tools to verify the simulation results. mgf, rgf, mdf and rdf are estimated from the simulation studies for each task load.

Effect of L on GR and HR



(a) Simulation (b) Analytical

Figure 8. Effect of Task Load on GR



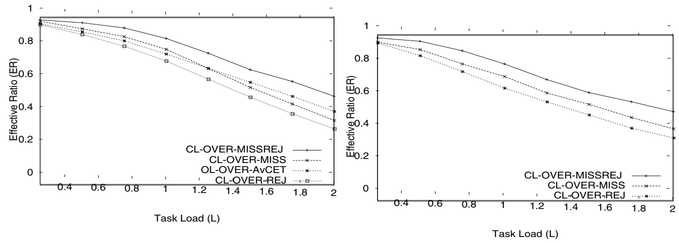
(a) Simulation (b) Analytical

Figure 9. Effect of Task Load on HR

Figure 8 (9)depicts GR changes with respect to L. Figure 8a (9a) is the result from the simulation model and Figure 8b (9b) is the result from the Matlab simulation. As expected, increasing L decreases GR (HR) for all the algorithms except for *CL-OVER-REJ* (*CL-OVER-MiSS*) where $GR = \%99$ ($HR = \%99$). This is because the feedback mechanism always drives RR (MR) to $RR_s = 1\%$ ($MR_s = \%1$) and since $GR = 1 - RR$ ($HR = 1 - MR$), *CL-OVER-REJ* (*CL-OVER-MISS*) offers $GR = 1 - RR_s$ ($HR = 1 - MR_s$). The figures also show that *CL-OVER-MISS*(*CL-OVER-REJ*) offers the minimum GR (HR) for all L and the difference between this approach and the other approaches increases as L increases. This is because *CL-OVER-MISS* (*CL-OVER-REJ*) tries to achieve $MR = 1\%$ ($RR = 1\%$) for the L. Since MR (RR) increases as L increases, for *OL-OVER*, the feedback mechanism increases EET as L increases. This results in significantly decreasing GR (HR). From Figure 8a (9a), we notice that *CL-OVER-MISSREJ* offers a GR (HR) greater than *CL-OVER-MISS* (*CL-OVER-REJ*) and *OL-OVER-AvCET*. Figure 8b (9b) shows that the results obtained by Matlab simulating confirms the simulation studies behavior. Since the average GR (HR) has been calculated for a long run, its value approximately equals to the steady state guarantee (hit) ratio from the Matlab simulation.

Effect of Task Load (L) on Effective Ratio (ER)

Figure 10 shows the effect of L on ER. *CL-OVER-MISSREJ* offers the highest ER for all L. This is because, the feedback mechanism and the controller try to adapt, as L varies, EET to maximize the product of HR and GR (i.e., maximize ER). The figures also show that *CL-OVER-REJ* offers the lowest ER for all L. This is because it tries to achieve $RR = 1\%$. This results in significantly decreasing HR as L increases, which degrades ER. From Figure 10a, we notice that *CL-OVER-MISS* offers an ER greater



(a) Simulation (b) Analytical

Figure 10. Effect of Task Load on ER

than *OL-OVER-AvCET* for $L \leq 1.2$, whereas it offers an ER lower than *OL-OVER-AvCET* for $L > 1.2$. Because for $L > 1.2$ GR offered by *CL-OVER-MISS* decreases significantly as compared to *OL-OVER-AvCET* when L increases.

7 Conclusions

In this paper, we proposed an open-loop overlap scheduling algorithm, which dynamically guarantees incoming tasks via on-line admission control and planning based on their average execution time. Secondly, we used the feedback control theory to design three closed-loop approaches. The contributions and the results are:

1. Proposed a new performance metric (ER) to evaluate the open-loop and closed-loop scheduling algorithms.

2. Developed and analyzed *OL-OVER-AvCET* with task overlap for firm real-time tasks. The overlap based algorithm improves ER 15% over non-overlap algorithms.

3. Developed and analyzed closed-loop algorithms using the feedback control theory.

4. Developed and analyzed *CL-OVER-MISS, CL-OVER-REJ*, and *CL-OVER-MISSREJ* for firm real-time tasks. The *CL-OVER-MISSREJ* algorithm performs better, in terms of ER, than *CL-OVER-MISS* or *CL-OVER-REJ*. Also, it performs better (20% gain) than the open-loop scheduling algorithm *(OL-OVER-AvCET)*. Also, the *(OL-OVER-AvCET)* performs better, in terms of ER, than both *CL-OVER-MISS* and *CL-OVER-REJ* under high system loads.

5. Simulation and analytical experiments to study the instantaneous behavior of MR_{kT} and RR_{kT} in response to dynamic L. In particular, we have studied this behavior for step task load. The purpose is to verify the analytical model against the simulation model. (Due to space limitations, these results are not shown).

Our current work focuses on developing a Real-time Linux based test-bed implementing the proposed schedulers and carrying out experimental studies.

References

[1] Patrik Persson and Górel Hedin, “Interactive execution time predictions using reference attributed grammars,”*Proceedings of WAGA ’99: Second Workshop on Attribute Grammars and their Applications,* Amsterdam, The Netherlands, Mar. 1999.

[2] G. Manimaran and C. Siva Ram Murthy, “An efficient dynamic scheduling algorithm for multiprocessor real-time systems,” *IEEE Transactions on Parallel and Distributed Systems,* vol. 9, no. 3, pp. 312-319, Mar. 1998.

[3] K. Ramamritham, “Allocation and scheduling of precedence-related periodic tasks,” *IEEE Transactions on Parallel and Distributed Systems,* vol. 6, no. 4, pp. 412-420, Apr. 1995.

[4] K. Ramamritham, J.A. Stankovic, and P.F. Shiah, “Efficient scheduling algorithms for real-time multiprocessor systems,” *IEEE Trans. on Parallel and Distributed Systems,* vol. 1, no. 2, pp. 184-194, Apr. 1990.

[5] K.-E. A^orzén, B. Bernhardsson, J. Eker, A. Cervin, K. Nilsson, P. Persson, and L. Sha, “Integrated control and scheduling”, *Internal report TFRT-7582*, Department of Automatic Control, Lund Univ., Aug. 1999.

[6] G. Manimaran, C. Siva Ram Murthy, M. Vijay, and K. Ramamritham, “New algorithms for resource reclaiming from precedence constrained tasks in multiprocessor real-time systems,” *Journal of Parallel and Distributed Computing,* vol.44, no.2, pp.123-132, Aug. 1997.

[7] C. Shen, K. Ramamritham and J.A. Stankovic, “Resource Reclaiming in Multiprocessor Real-Time Systems,” *IEEE Trans. on Parallel and Distributed Systems,* Vol. 4, No. 4, pp. 382-397, April 1993.

[8] Giorgio Buttazzo and Luca Abeni, “Adaptive rate control through elastic scheduling,” *Proceedings of the 39th IEEE Conference on Decision and Control,* Sydney, Australia, Dec. 2000.

[9] D. Seto, et. al., “On task schedulability in real-time control systems,” *IEEE Real-Time Systems Symposium,* Dec. 1996.

[10] M. Ryu and S. Hong, “Toward automatic synthesis of schedulable real-time controllers,” *Integrated Computer-Aided Engineering,* 5(3) pp. 261-277, 1998.

[11] X. Liu, L. Sha, M. Caccamo, and G. Buttazzo “Online control optimization using load driven scheduling,” *Proceedings of the* 39^{th} *IEEE Conference on Decision and Control,* Sydney, Australia, Dec. 2000.

[12] M. Caccamo, G. Buttazzo, and L. Sha, “Elastic feedback control,” *IEEE Proceedings of the* 12^{th} *Euromicro Conference on Real-Time Systems,* Stockholm, Sweden, pp.121-128, June 2000.

[13] T. F. Abdelzaher, E. M. Atkins, and K. G. Shin, “QoS negotiation in real-time systems and its application to automated flight control,” *IEEE Real-Time Technology and Applications Symposium,* June 1997.

[14] K. G. Shin and C. L. Meissner, “Adaptation and graceful degradation of control system performance by task reallocation and period adjustment,” *EuroMicro Conference on Real-Time Systems,* June 1999.

[15] Giorgio Buttazzo, Giuseppe Lipari, and Luca Abeni, “Elastic task model for adaptive rate control,” *Proceedings of the IEEE Real-Time Systems Symposium,* Madrid, Spain, Dec. 1998.

[16] J. Stankovic, C. Lu, S. Son, and G. Tao, “The case for feedback control real-time scheduling,” *EuroMicro Conference on Real-Time Systems,* pp. 11-20, June 1999.

[17] C. Lu, J. A. Stankovic, G. Tao, and S. H. Son, “Design and evaluation of feedback control EDF scheduling algorithm”, *Real-Time Systems Symp. (RTSS),* pp. 56-67 Dec. 1999.