

Chapter 5

Model of Execution Time and Overhead in a Real-time System

In this chapter, we discuss the various scheduling overheads incurred by real-time systems, as well as, how they are measured and incorporated in our tool. We also present the modified scheduling equations of the different scheduling algorithms used by the analysis and predictor units of our tool.

5.1 Breakdown of Overhead in a Real-time System

5.1.1 Timer Overhead

A timer overhead is incurred in every timer-based real-time system. In a fully preemptive system, a timer is used to indicate the scheduling points of the system, while in non-preemptive system, it is used mainly to keep track of time. A timer overhead is incurred, when the timer interrupts the processor. As with any interrupt, the processor has to stop whatever instruction it was executing, disable all interrupts to prevent it from being interrupted while handling an interrupt, saves the contents of its registers, handles the interrupt, re-enable interrupts, and then resumes executing instructions. The higher the frequency of timer interrupt, the greater the overhead which is incurred. We discuss the timer interrupt resolutions of both our operating systems, and additional functions performed by the timer interrupt handlers below.

Since chimera is preemptive, the timer interrupt is also considered as the scheduling point. When the timer interrupts the system, the timer interrupt handler updates the system time, resets the timer counter registers if necessary, and then invokes the scheduler. We discuss the scheduler functions performed in section 5.2. The time spent in handling the timer is approximately $17\mu s$, and the timer period is $1ms$. Consequently, the percentage of processor utilization incurred by the timer in chimera is

approximately 1.7%. The chimera kernel rounds up all requested task periods to the nearest *ms*, therefore no additional overhead is incurred due to timer blocking.

For the Echidna operating system, the timer is only utilized to keep track of system time, since it is a non-preemptive environment. Consequently, the system does not need to be preempted often, and is only interrupted by the timer to reset the timer control registers, and update the system time. The amount of overhead incurred in handling the timer interrupt is approximately 5 μ s, and the period of the timer is 200ms. The percentage of processor utilization incurred by the timer using Echidna is approximately 0.0025%.

5.1.2 Scheduling of Tasks Overhead

This overhead is the time spent by the kernel in performing scheduler functions like determining which task to execute next, and moving various tasks to their respective queues. Four different tasks queues are available in the Chimera environment, namely *ready*, *pause*, *block*, and *active* task queue. The ready queue are tasks that are waiting to execute, the pause queue has the tasks that are waiting for the start of their next period, the block queue has the tasks that are blocked on signals like semaphores, and the active task queue is the task that is being executed. When the scheduler is invoked by the timer, it firsts moves tasks from the block queue to the pause or ready queue, it then wakes up tasks from the pause queue that have start times before or equal to the system time. If any of the tasks it wakes up has a higher priority than the active task, the scheduler forces a context switch.

The amount of time spent moving tasks to different queues is dependent on the number of tasks present in each queue, the sorting algorithm utilized by the queues, and the scheduling algorithm implementation. If the queues are sorted, the worst-case complexity of inserting or deleting an element into the queue becomes an order of n , where n is the number of tasks present in the system. In Chimera, the ready queue is sorted in the order of decreasing task priority, and the pause queue is sorted in the order of increasing task deadline. The block queue is unsorted since tasks are blocked on signals.

For simplicity of the extractor, we measure the scheduler overheads of each taskset on per cycle basis, and include the overhead in our equations by dividing the average scheduler overhead by the average period for which the scheduler is invoked.

In Echidna, the scheduler is only invoked at the end of cycle execution of each task. Echidna contains the same task queues as Chimera. When the scheduler is invoked, it moves the active task to the pause queue, wakes up tasks from the pause queue, and selects the task from the ready queue with the earliest deadline to be the next active task. Both the ready and pause queues are sorted in the order of increasing tasks deadlines. Like in Chimera, we measure the scheduler overheads of each taskset on per cycle basis, and include the overhead in our equations by dividing the average scheduler overhead by the average period for which the scheduler is invoked.

5.1.3 Context Switch Overhead

This overhead is only incurred in a preemptive environment. A context switch is forced only when the active task ceases to be the highest priority task in the task. When a context switch occurs, the context switch handler saves the context of the active task, selects the highest priority task from the ready queue, and swaps its context with that of the active task. The amount of time incurred in swapping tasks' context is constant regardless of the tasks being swapped, but the context switch overhead is not constant since it partially depends on searching the ready queue for the highest priority task. The overhead incurred in selecting this task is of the order n , where n is the number of tasks in the system. The average value of the context switch overhead per taskset is measured and considered in our scheduling equations.

5.1.4 Communication Overhead

Using our model of a software component, illustrated in Figure 3, a task uses its variable input and output ports to communicate with other tasks. This intertask communication is performed by the operating system kernel on behalf of the task, before

and after the execution of a task's cycle as shown in Figure 4. The SVAR communication mechanism, which is utilized by both our operating systems, is described in detail in chapter 3. The overhead incurred by the three different variations of the SVAR communication mechanism is discussed below.

In a preemptive environment, the SVAR overhead is divided into two parts, namely, the data transfer time, and the waiting time [15]. The data transfer time is the time it takes to lock and unlock the SVAR global table, plus the time incurred in transferring each variable. The data transfer time is characterized by the equations below [15]:

$$t_{IP} = V_1 + n_{IP}V_a + \sum_{i=1}^{n_{IP}} R(x_{Pi}) \quad (18)$$

$$t_{OP} = V_1 + n_{OP}V_a + \sum_{i=1}^{n_{OP}} R(x_{Pi}) \quad (19)$$

where:

- t_{IP} and t_{OP} are the time required to transfer the input and output variables of a task.
- V_1 is the overhead for locking and unlocking the global table.
- V_a is the overhead of transferring each additional variable.
- n_{IP} and n_{OP} are the number of input and output variables of a task.
- x_{IP} and x_{OP} are the number of transfers for the inputs and outputs of a task.
- $R(x)$ is the time required for x transfers.

The worst-case waiting time of any task, W_k , is the longest time the task may have to wait for another task to release the global table lock. W_k can be characterized by the equation below [15]:

$$W_k = \max(t_{IP}, t_{OP}) \Big|_{i=1}^n$$

where n is the number of tasks in the system. Only one task can request the lock at a time, therefore there is no contention with other tasks on the processor.

The SVAR overhead incurred in a non-preemptive environment is zero, since there is no need to copy input and outputs variables, and data can be accessed directly from the global table.

The SVAR overhead incurred in our limited preemption environment is the same as the overhead incurred in the preemptive environment if inputs are provided by tasks in the RM layer for the NPEDF layer, and vice versa. If this is not the case, the SVAR overhead incurred by tasks in the NPEDF layer is zero, since data can be accessed directly from the SVAR table with data integrity maintained. The SVAR overhead incurred by the RM and EDF layers will remain the same as those in a preemptive environment.

The SVAR overhead is determined by adding measured waiting and transfer times for the input variables of each task with the measured waiting and transfer times of its output variables. The average measured value of each task is utilized in our equations.

5.1.5 Task exit Overhead

This overhead is incurred upon the normal completion of a task. When a task finishes executing, the scheduler traps into the kernel, and the task exit handler restores the state of a task such that it can be executed for its next cycle. The functions performed by the task exit handler includes, updating tasks start times and deadlines, and moving the context of the task from the active task queue to the pause queue. The task exit overhead measured in Echidna was found to be negligible, since the amount of data stored by a task object is small in a non-preemptive. In Chimera, the average task exit overhead measured was $111.73\mu s$.

5.2 Modified Scheduling Equations

In this section, we provide the modified scheduling equations that we derived to accurately account for the overheads incurred in implementing scheduling algorithms in real-time systems. We provide justification for all our equations since they are approximations for real system implementations, in addition to the fact that most of the

equations we presented can be deduced from equations already provided in [1, 2, 3, 4, 5, 6, 7, 8, 10], and as such, the same proofs apply. The equations we provided are for RM, EDF, NPEDF, MUF, and MPLP. We do not provide equations for MCE, since this algorithm is the same as NPEDF in our case. We present the accuracy of these equations in chapter 6, where we provide the results from experimental validation.

Before we present the equations for the different scheduling algorithms, the following definitions are needed:

- C_i, T_i is the worst case execution time and the period of task i .
- C_{tmr}, T_{tmr} is the execution time and period of the timer interrupt handler.
- C_{com_i} is the communication overhead of task i . This includes the time to copy data from and to the SVAR global table.
- C_{sch} is the average overhead incurred in the scheduling of tasks by the scheduler. The scheduler overhead is described in section 5.1.
- C_{csw} is the context switch overhead of a task. We consider the average measured value per taskset.
- C_{exit} is the average measured overhead incurred by the task exit interrupt handler.
- $NP_{RM}(i)$ and $NP_{EDF}(i)$ are the number of preemptions incurred by task i using RM and EDF scheduling algorithms, respectively.
- $NP_{MUF_{RM}}(i)$ and $NP_{MUF_{EDF}}(i)$ are the number of preemptions incurred by task i using the MUF scheduling algorithm, in the RM and EDF layers, respectively.
- $NP_{MPLP_{RM}}(i)$, $NP_{MPLP_{NPEDF}}(i)$ and $NP_{MPLP_{EDF}}(i)$ are the number of preemptions incurred by task i using the MPLP scheduling algorithm, in the RM, NPEDF and EDF layers, respectively.
- $U_{RM}(n)$, $U_{EDF}(n)$, $U_{NPEDF}(n)$, $U_{MUF}(n)$, and $U_{MPLP}(n)$, are the processor utilization of executing a taskset of n tasks using RM, EDF, NPEDF, MUF, and MPLP scheduling algorithms, respectively.

- $U_{MUF_{pre}}(n)$ and $U_{MPLP_{pre}}(n)$ are the amounts of processor utilization spent on preemption using the MUF and MPLP scheduling algorithms, respectively.
- $p_{RM_i}(t)$ and $p_{NPEDF_i}(t)$ are the amount of processor time spent executing tasks 1 to i , using MUF and MPLP scheduling algorithms in the RM and NPEDF layers, respectively.

In all our equations, we assume that there is no overhead due to timer blocking, since the timer resolution in our preemptive environment is $1ms$, and the operating system rounds requested task periods to the nearest ms .

5.2.1 Modified Equations for RM Scheduling Algorithm

For a taskset consisting of n periodic tasks, ordered in non-decreasing order by period, the modified equations for RM algorithm, which are required for schedulability for $\forall i, t; i \leq n; t \geq 0$ are as follows:

$$NP_{RM}(i) = \min\left(\left\lceil \frac{C_i - T_{tmr}}{T_{tmr}} \right\rceil, \sum_{k=1}^{i-1} \left\lceil \frac{T_i}{T_k} \right\rceil\right) \quad (20)$$

$$U_{RM}(n) = \sum_{i=1}^n \frac{C_i + C_{com_i} + C_{exit}}{T_i} + \sum_{j=1}^n \frac{2NP_{RM}(j)C_{csw}}{T_j} + \frac{C_{tmr} + C_{sch}}{T_{tmr}} \leq n \left(2^{\frac{1}{n}} - 1\right) \quad (21)$$

$$\sum_{j=1}^i \frac{C_j + C_{com_j} + C_{exit}}{t} \left\lceil \frac{t}{T_j} \right\rceil + \sum_{j=1}^i \frac{2NP_{RM}(j)C_{csw}}{t} \left\lceil \frac{t}{T_j} \right\rceil + \frac{C_{tmr} + C_{sch}}{t} \left\lceil \frac{t}{T_{tmr}} \right\rceil \leq \quad (22)$$

Justification

The equations provided for RM scheduling algorithm were derived from the original Liu and Layland equations, Katcher's modified equations for a timer driven scheduler, and equations provided by Arora and Stewart for fixed priority schedulers [1, 3, 8, 10]. The terms we introduce are mainly the number of context switches due to process preemption, the communication overhead incurred by each task.

Equation (21) was derived from equations (1) and (9). Equation (1) gives the maximum utilization that can be incurred by any taskset for schedulability, while equation

(9) provides the overhead incurred by other tasks preemption, in addition to timer and blocking overhead [1, 8]. We introduce the communication term C_{com_i} , since that is the communication overhead incurred by a task at every cycle. Since our implementation of the RM algorithm is timer based, we also included the timer overheads incurred by timer interrupt handler, and the scheduler overhead incurred in moving tasks to their respective queues, and selecting the highest priority task to execute. The scheduler is invoked by the timer interrupt handler, and since this happens every timer period, we add this overhead C_{sch} , to the timer overhead, C_{tmr} .

Another term that we introduced is the worst case number of preemptions that a task can incur in its period. Even though Katcher does consider preemption in his equations, he treats it as a single constant term [2, 3]. We believe that the worst case number of preemption that a task can incur is $\sum_{k=1}^{j-1} \left\lceil \frac{T_j}{T_k} \right\rceil$, with the assumption that tasks are ordered in non-decreasing order by period. This term is the sum of the number of times the higher priority tasks before each task in a taskset, can execute. This term cannot exceed the maximum number of times can be interrupt by the timer, since that indicates the scheduling point. We subtract this value by one timer interrupt period, since a task cannot be preempted when its remaining execution time is less than or equal to the timer period. We multiply the number of preemptions by $2C_{csw}$ to determine the amount of time spent on context switching for this task, since every task preemption uses of two context switches.

Equation (22) is the amount of processor utilization spent executing tasks 1 to i , during the time interval of 0 to t . Since this equation is exactly the same format as equation (8), but with greater inclusion of system overhead, the same proofs apply.

5.2.2 Modified Equations for EDF Scheduling Algorithm

Below are the modified equations provided for EDF and NPEDF scheduling algorithms, respectively.

5.2.2.1 Modified Equations for EDF

The equations required for schedulability using EDF for $\forall i, t; i \leq n; T_1 < t \leq T_n$ are:

$$NP_{EDF}(i) = \min \left(\left\lceil \frac{C_i - T_{tmr}}{T_{tmr}} \right\rceil, \sum_{k=1}^{i-1} \left\lfloor \frac{T_i}{T_k} \right\rfloor \right) \quad (23)$$

$$J_{EDF}(n) = \sum_{j=1}^n \frac{C_j + C_{com_i} + C_{exit}}{T_j} + \sum_{j=1}^i \frac{2NP_{EDF}(j)C_{csw}}{T_j} + \frac{C_{tmr} + C_{sch}}{T_{tmr}} \leq 1 \quad (24)$$

$$\geq \sum_{j=1}^n (C_j + C_{com_i} + C_{exit}) \left\lfloor \frac{t}{T_j} \right\rfloor + \sum_{j=1}^i 2NP_{EDF}(j)C_{csw} \left\lfloor \frac{t}{T_j} \right\rfloor + (C_{tmr} + C_{sch}) \left\lfloor \frac{t}{T_{tmr}} \right\rfloor \quad (25)$$

Justification

Equations (24) and (25) were derived from Jeffay and Stone's equations provided on preemptive EDF, namely equations (4), (11), and (13) [1, 4, 5]. The necessary and sufficient condition for this scheduling algorithm as shown in chapter 2, is that the total processor utilization is less than 1. Equation (4), completely ignored system overhead spent on interrupts and other implementation costs, while equations (11) and (13) try to rectify this error by including interrupt handlers. However, the assumption made in this case was that all interrupt handlers were periodic, and could be evaluated in isolation. As we have shown in equations (21) and (22), some interrupts (system overheads), are dependent on the tasks period, like the communication and preemption overheads. Consequently, even though they are performed by the kernel, cannot be evaluated in isolation. We modify equation (11) to correspond with this kernel implementation, to produce equation (24). In this case, the estimated worst case number of preemptions is also the number of times that tasks with higher priorities, i.e. earlier deadlines, can execute within a task's period, and this term cannot exceed the maximum number of times can be interrupt by the timer as justified in equation (21). Equation (25), is the modified version of equation (13), i.e. processor utilization incurred by executing tasks 1 to n, with the same changes made in equation (24) to correspond with real systems implementation.

5.2.2.2 Modified Equations for NPEDF

The equations required for schedulability using the NPEDF algorithm for $\forall i, t; i \leq n; t \geq 0$ are:

$$U_{NPEDF}(n) = \sum_{i=1}^n \frac{C_i + C_{sch}}{T_i} + \frac{C_{tmr}}{T_{tmr}} \leq 1 \quad (26)$$

$$t \geq (C_i + C_{sch}) + \sum_{j=1}^{i-1} (C_j + C_{sch}) \left\lfloor \frac{t-1}{T_j} \right\rfloor + C_{tmr} \left\lfloor \frac{t}{T_{tmr}} \right\rfloor \quad (27)$$

Justification

Equations (26) and (27) are mainly extensions of earlier equations provided for non-preemptive EDF, namely equations (4), (5), (11) and (14), but with adjustments done to account for more accurate system overhead and interrupt handlers. Equations (4) and (5) are the earlier equations provided by Jeffay and Stone, where system overhead and interrupt handlers were ignored, while equations (11) and (14) only consider periodic interrupt handlers. Like we showed for equations (24) and (25), not all interrupt handlers and system overheads can be considered in isolation. In a non-preemptive algorithm, the scheduler overhead incurred C_{sch} , is also a function of a tasks period, since, the scheduler is only invoked after a task finishes executing every cycle. The communication and preemption overheads incurred in a non-preemptive are both zero, so we exclude these overheads from our equation. Timer interrupts also occur less frequently, since the system is only interrupted by the timer to ensure that the timer counter registers do not overflow. We also consider this overhead in our equations.

5.2.3 Equations for MUF Scheduling Algorithm

The equations required for schedulability using the MUF scheduling algorithm are as follows, for $\forall i, l; i \leq n; l \leq n$, and tasks l to i are scheduled using the RM algorithm:

$$NP_{MUF_{RM}}(i) = \min \left(\left\lfloor \frac{C_i - T_{tmr}}{T_{tmr}} \right\rfloor, \sum_{k=1}^{i-1} \left\lfloor \frac{T_i}{T_k} \right\rfloor \right) \quad (28)$$

$$NP_{MUF_{EDF}}(i) = \min\left(\left\lceil \frac{C_i - T_{tmr}}{T_{tmr}} \right\rceil, \sum_{k=l+1}^{i-1} \left\lfloor \frac{T_i}{T_k} \right\rfloor + \sum_{k=1}^l \left\lceil \frac{T_i}{T_k} \right\rceil\right) \quad (29)$$

$$J_{MUF_{pre}}(n) = \left(\sum_{j=1}^l \frac{2NP_{MUF_{RM}}(j)}{T_j} + \sum_{j=l+1}^n \frac{2NP_{MUF_{EDF}}(j)}{T_j} \right) C_{csw} \quad (30)$$

$$U_{MUF}(n) = \sum_{i=1}^n \frac{C_i + C_{com_i} + C_{exit}}{T_i} + U_{MUF_{pre}}(n) + \frac{C_{tmr} + C_{sch}}{T_{tmr}} \leq 1 \quad (31)$$

$$\leq l \quad p_{RM_i}(t) = \sum_{j=1}^i (C_j + C_{com_j} + C_{exit}) \left\lfloor \frac{t}{T_j} \right\rfloor + \sum_{j=1}^i 2NP_{MUF_{RM}}(j) C_{csw} \left\lfloor \frac{t}{T_j} \right\rfloor + \quad (32)$$

$$(C_{tmr} + C_{sch}) \left\lfloor \frac{t}{T_{tmr}} \right\rfloor \leq t$$

$$\sum_{i=l+1}^n (C_i + C_{com_i} + C_{exit}) \left\lfloor \frac{t}{T_i} \right\rfloor + \sum_{j=l+1}^n 2NP_{MUF_{EDF}}(j) C_{csw} \left\lfloor \frac{t}{T_j} \right\rfloor \leq t - p_{RM_l}(t) \quad (33)$$

Justification

As with the schedulability of all tasksets, total processor utilization must be less than 1, as seen in equation (31). Since this algorithm is the combination of the RM and EDF algorithms, the total processor utilization is a combination of the utilization of tasks in the RM layer, with the utilization of tasks in the EDF layer. We see that the preemption term for tasks in the RM layer remains that same as in equation (20), but the preemption term for tasks in the EDF layer slightly varies from the model in equation (23), since we also have to consider preemption for tasks in the RM layer since they have higher priority. Equation (32) represents the amount of processor time spent executing tasks in the RM layer, and this value must be less than the total amount of processor time available. Equation (33) was derived from equation (6), and it represents the amount of processor time spent executing tasks in the EDF layer. This time as proven by Liu and Layland in their mixed priority algorithm, must be less than or equal to the total amount of processor time available after tasks in the RM layer are executed [1].

5.2.4 Equations for MPLP Scheduling Algorithm

The equations required for schedulability of a taskset, using the MPLP algorithm, for $\forall i, l, m; i \leq n; 1 \leq m \leq n$, and tasks 1 to l , $l+1$ to m , and $m+1$ to n , are in the RM, NPEDF, and EDF layers, respectively, are as follows:

$$NP_{MPLP_{RM}}(i) = \min \left(\left\lceil \frac{C_i - T_{imr}}{T_{imr}} \right\rceil, \sum_{k=1}^{i-1} \left\lceil \frac{T_i}{T_k} \right\rceil \right) \quad (34)$$

$$NP_{MPLP_{NPEDF}}(i) = \min \left(\left\lceil \frac{C_i - T_{imr}}{T_{imr}} \right\rceil, \sum_{k=1}^l \left\lceil \frac{T_i}{T_k} \right\rceil \right) \quad (35)$$

$$NP_{MPLP_{EDF}}(i) = \min \left(\left\lceil \frac{C_i - T_{imr}}{T_{imr}} \right\rceil, \sum_{k=m+1}^{i-1} \left\lfloor \frac{T_i}{T_k} \right\rfloor + \sum_{k=1}^m \left\lceil \frac{T_i}{T_k} \right\rceil \right) \quad (36)$$

$$J_{MPLP_{pre}}(n) = \sum_{j=1}^l \frac{2NP_{MPLP_{RM}}(j)}{T_j} C_{csw} + \sum_{j=l+1}^m \frac{2NP_{MPLP_{NPEDF}}(j)}{T_j} C_{csw} + \sum_{j=m+1}^m \frac{2NP_{MPLP_{EDF}}(j)}{T_j} C_{csw} \quad (37)$$

$$U_{MPLP}(n) = \sum_{i=1}^n \frac{C_i + C_{com_i} + C_{exit}}{T_i} + U_{MPLP_{pre}}(n) + \frac{C_{imr} + C_{sch}}{T_{imr}} \leq 1 \quad (38)$$

$$\leq l \quad p_{RM_i}(t) = \sum_{j=1}^i (C_j + C_{com_j} + C_{exit}) \left\lceil \frac{t}{T_j} \right\rceil + \sum_{j=1}^i 2NP_{MPLP_{RM}}(j) C_{csw} \left\lceil \frac{t}{T_j} \right\rceil + (C_{imr} + C_{sch}) \left\lceil \frac{t}{T_{imr}} \right\rceil \leq t \quad (39)$$

$$\leq m \quad p_{NPEDF_i}(t) = \sum_{j=l+1}^i (C_j + C_{com_j} + C_{exit}) \left\lceil \frac{t}{T_j} \right\rceil + p_{RM_l}(t) + \sum_{j=l+1}^i 2NP_{MPLP_{NPEDF}}(j) C_{csw} \left\lceil \frac{t}{T_j} \right\rceil \leq t \quad (40)$$

$$\sum_{i=m+1}^n (C_i + C_{com_i} + C_{exit}) \left\lceil \frac{t}{T_i} \right\rceil + \sum_{j=m+1}^n 2NP_{MPLP_{EDF}}(j) C_{csw} \left\lceil \frac{t}{T_j} \right\rceil \leq t - p_{NPEDF_m}(t) \quad (41)$$

Justification

The equations we provided for this algorithm are a combination of the equations provided for the three different algorithms, namely RM, NPEDF, and EDF. However, we had to modify the model we provided for the NPEDF algorithm, since the rules of this scheduling algorithm are such that tasks in the NPEDF layer can be preempted by tasks in the RM layer, as described in Chapter 2. Therefore, we added a preemption term to the NPEDF layer, as shown in equation (35), and moved the C_{sch} term in this layer to the C_{tmr} term since the scheduler must be invoked during every timer interrupt to check for the arrival of higher priority tasks. Tasks in the RM and EDF layers, have the same equations as those provided for the MUF algorithm, with the exception that tasks in the EDF layer can now be also preempted by tasks in the NPEDF layer. Consequently, the preemption term in the EDF layer has been updated to include the number of preemptions by tasks in the NPEDF layer, as shown in equation (36). We have also provided total processor time used in executing tasks 1 to i , based on whether the task is in the RM, EDF, or NPEDF layers, respectively, as shown in equations (39), (40), and (41). These processor times must be less than the total amount of processor time available after executing tasks in the higher layers, as proven for the MUF equations [1].