Scheduling Analysis of Fixed Priority Hard Real-Time Systems with Multiframe Tasks

Areej Zuhily

Submitted for the degree of doctor of philosophy

University of York
Department of Computer Science

January 2009

Abstract

Scheduling analysis of real time systems has been studied by most researchers assuming the tasks of the systems have constant worst case execution time bounds during their cycle of execution. However, this is not the case in a multiframe task where the execution time could be different from one instance to another, as in multimedia applications like MPEG.

Some researchers have introduced sufficient scheduling analyses for a restricted model of multiframe tasks. The contributions in this thesis present scheduling analysis for a less strict model of multiframe tasks. The analysis is presented in two steps. In the first step, exact scheduling analysis is presented by response time analysis; where the worst case response time of multiframe tasks is formulated. This formulation is then extended to multiframe tasks that are subjected to blocking, release jitter and arbitrary deadlines. Another extension of the formulation is given to cover frame specific deadlines; where a multiframe task has more than one deadline relative to its frames.

With large systems of multiframe tasks, the exact response time analysis becomes computationally intractable. So, in the second step we present and compare some sufficient approaches that analyze the schedulability of large systems with multiframe tasks. In this step we first study the safety of each approach then we compare them to find out the schedulability performance each of them provides.

Contents

Acknowledgements

Many thanks to some organisations and people who contributed in achieving this thesis. First of all, huge thanks and gratefulness to Damascus University who sponsored me during my study and to The University of York who gave me the opportunity to get a degree from a 5* department.

A particular gratitude to my supervisor Prof. Alan Burns who patiently provided me valuable advice and support throughout years of study. Also, many thanks to Dr. Robert Davis for the bright ideas we end up with each time I got stuck with a problem.

A special thanks to the person who without him I would not have finished this thesis, the one who enthusiastically encouraged me and supported me as much as he could without any doubt, my husband, Ehsan.

Finally, many thanks to the people who keep providing me all encouragement and support, my parents and family.

Declaration

Parts of this thesis have been published in some proceedings and journals . This material represents the author's contributions, but was published jointly with the author's supervisor Prof. Alan Burns.

Material based on Chapter 3 was published in the international conference "International Colloquium on Theoretical Aspects of Computing (ICTAC)"[77]. Material based on Chapter 4 was published in the international conference "IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)"[78].

Parts of the material based on Chapter 5 was published as a technical report [75]. Other parts of the material based on Chapters 5 and 6 were published in the international conference "International Conference on Real-Time and Network Systems (RTNS) "[79]. An extended version of this RTNS paper has been invited to be published in Real Time Systems Journal.

Theorem 16 in the appendix was published in the journal "Information Processing Letters"[76].

List of Tables

List of Figures

1 Introduction

Timing requirements are the basic aspects of real-time systems; where a real-time system, RTS, is a system that is required to react to stimuli from the environment within time intervals dictated by the environment [25]. For example, an application running on an operating system, like real-time Unix, can be considered as a real-time system if it is expected to respond to a command within a defined time interval. Process control is another example of a real-time system where the computer controls the operations of the sensors and actuators to ensure that the correct operations are performed at the appropriate times. RTSs are divided, according to timing requirements, into: *hard* and *soft real-time systems*. A hard real-time system is a system whose responses must occur within specified deadlines. A soft real-time system is a system that functions correctly if the deadline is occasionally missed [25, 53]. Contributions in this thesis are concerned with hard real-time systems.

From an analysis point of view, a RTS is usually represented by a set of tasks; and each task consists of a number of jobs that are executed in a cyclic way. Execution of the tasks is controlled by the operating system using some scheduling algorithms[1]; where the operating system controls and coordinates the use of the hardware among the various application programs for the user tasks [59, 68]. In other words, application software is usually designed as a number of separate tasks that are scheduled by the operating system [67, 63] via the scheduler; which is the part of the kernel that determines the next runnable task [46].

The real-time tasks are divided, according to the arrival times of the tasks, into periodic tasks and sporadic tasks. The arrival times of periodic tasks are fixed so that each task arrives into the system every fixed interval of time, called a period. On the

[1] A scheduling algorithm is a set of rules that determine the executing task at a particular moment [52]).

other hand, the arrival times of a sporadic task are not fixed, instead, the task has a minimum interval of time to arrive in the system. Within the contributions of this thesis, we primarily consider periodic tasks.

A basic ordinary periodic real-time task is usually characterised by three parameters. The first parameter is the execution time of the task to characterise the time that this task takes during the execution of its jobs. The second one is the period of the task to characterise the arrival times of this task. The third one is the deadline of the task to characterise the time in which this task has to complete the execution of its jobs.

Most research considers the execution time of the real-time task as a constant value for all invocations of its jobs. However, for some real-world applications the execution times of the task are not constant for all its jobs. We call the task whose execution time could vary from one invocation to the next a multiframe task.

1.1 Multiframe Tasks

The fundamental principle in the real-time *multiframe*, MF, task is that its worst-case execution time is different from one invocation to another, for instance, a task that executes with the worst-case execution times of 10ms and 5ms is said to have two different frames. An example, found in industrial applications [26], is a periodic task that does a small amount of data collection in each period consuming a small execution time, but then summarises and stores this data every n cycles using a much more expensive algorithm that consumes a larger execution time.

Scheduling research into MF tasks started when Mok and Chen [56, 57] introduced this MF concept in 1996 as a generalisation of the classic Liu and Layland model [52]. They proposed a utilisation based schedulability test, for fixed priority scheduling, under Rate Monotonic, RM, [52] priority assignment[2]. They gave a utilisation bound, assuming the execution time sequence of each MF task has a particular restrictive property called *Accumulatively Monotonic*, AM. Subsequent papers have improved this utilisation bound but their tests remain inexact (sufficient but not necessary). These tests and the formal definition of the AM restriction will be given in Chapter 2.

[2] In RM priority assignment, the greater period the task has, the lower priority it is assigned.

An example of scheduling MF tasks is found within the MPEG coding standard where there are three types of video invocations (usually represented by the letters I, P and B). The I invocation usually takes much more decoding than the others, but may occur only every 10 invocations. The assumption that all invocations are I invocation leads to poor utilisation and the system could be theoretically unschedulable whilst practically it is schedulable. In addition, recently some researchers show how to efficiently utilise MF tasks using Dynamic Voltage Scaling, DVS, techniques for energy-efficient scheduling [74]. Adopting MF tasks in the system reduces the overall energy consumption of the system without missing its deadlines. Also, MF tasks may implement state machines, as in some avionics and automotive applications, with a well defined cycle of behaviour and worst case execution time bounds for each state.

1.2 Fixed Priority Scheduling

As scheduling is a fundamental function of an operating system to determine the order in which tasks execute, many researches are concerned with this area to either construct schedulable systems or to analyze the schedulability of proposed systems. The most popular scheduling policies are known as: Fixed Priority Scheduling (FPS), Earliest Deadline First (EDF), and Value Based Scheduling (VBS). This thesis is concerned with scheduling analysis of MF tasks for a fixed priority scheme.

Fixed priority scheduling, FPS, is a scheme where a priority is associated with each task in the system and the CPU is allocated to the highest priority runnable task. In FPS scheme all invocations of each task are assigned the same priority [53] so the priority of each task is fixed relative to other tasks in the system.

Fixed priority scheduling is recommended for many years as it is able to predict the ability to meet application response requirements [54]. From this recommendation, different operating systems support this fixed priority scheduling. For example, OSCAN, which is a preemptive[3] real-time multitasking operating system[4], offers

[3] In the preemptive systems, if a higher priority task is released during the execution of a lower priority task, there is an immediate switch to the higher priority task and the lower priority task has to wait until the higher priority task has finished its execution.

[4] In the preemptive multitasking operating system, tasks are preempted by the scheduler, and this preemption is accomplished with the aid of a timer interrupt [35].

priority-controlled task management [1]. Many commercial operating systems support FPS, for example, VxWorks, which is a real-time operating system, has a priority based preemptive scheduler[11]. PSOS, which is an object oriented operating system, schedules tasks using priority based criteria [15].

Likewise, there are academic operating systems supporting FPS, for example, server scheduling in the real-time operating system SHaRK can be based on fixed priority servers [2]. MaRTE [64] is another operating system that supports FPS. LynxOS [3], which is POSIX compatible, multitasking operating system, uses priority based scheduling [15].

1.3 Thesis Goal

The most popular paradigms for analysing the schedulability of real time systems are utilisation analysis and response time analysis. Having exact attributes of a system, the utilisation based analysis provides a sufficient but not necessary scheduling test whilst response time analysis provides an exact scheduling test in many situations. This thesis is concerned with the exact scheduling analysis of hard real-time systems with MF tasks supported by preemptive FPS, where a hard real-time system is considered as schedulable if all its MF tasks meet their relative deadlines.

Thesis Hypothesis

"The schedulability of real-time systems with multiframe tasks can be exactly analysed using formulated response time analysis that is extensible to a wide variety of situations. Where response time analysis is intractable, appropriate non-optimal heuristics exist and allow all systems to be analysed."

As the response time scheduling test is an exact test and the worst case response time analysis of MF tasks has not been fully studied yet, the objective of the thesis is to provide worst case response time analysis of MF tasks, so the schedulability of systems with MF tasks can be decided. However, exact response time analysis of large systems with un-restricted MF tasks is intractable, so the other objective of the thesis

is to provide some approaches to determine the schedulability of large systems with general MF tasks. The objectives of the thesis can be achieved in three steps as in the following:

1. In the first step, we present exact worst case response time analysis for systems with AM multiframe tasks. Analysis in this step starts from introducing a basic response time analysis and ends up with the response time analysis of AM multiframe tasks with blocking, release jitter, and arbitrary deadlines (i.e. including deadline greater than period).

2. Then in the second step, we relax the AM restriction and extend the response time analysis to cover non-AM multiframe tasks. In this step, a new concept called *critical frame* is used. In general, testing the schedulability of a set of MF tasks requires all possible phases of the tasks to be examined, which leads to an exhaustive enumeration problem (i.e. an intractable problem). However, for a particular application, not all invocations may need to be examined. We show how the critical frames, that can give rise to the worst-case response times of lower priority tasks, can be identified and their usage reduces the processing required for the response time analysis. Analysis in this step is developed in two further directions, the first direction is to be applicable to MF tasks with blocking, release jitter and arbitrary deadline; whilst the second direction is to cope with the scenario of having different deadlines per MF task where the deadline is relative to the frame of the MF task.

3. Having an intractable scheduling problem for large systems with non-AM multiframe tasks, some tractable but sufficient approaches are introduced in this step. Three of these tests depend on transforming all multiframe tasks in the system into AM tasks, which have only one critical frame, and then applying the exact response time formula on the transformed systems. The fourth approach depends on off-line calculation of the maximum interference from all higher priority MF tasks within the deadline of the analysed task. These different approaches are then compared.

1.4 Thesis Structure

This thesis is divided into nine chapters starting from this introduction and ending up with the conclusions of the contributions, whilst chapters in between are arranged according to the dependency and generalisation level. Chapter 2 defines the system model that is used throughout the thesis and presents a historical study of related research that has been done in fixed priority scheduling of multiframe tasks.

In Chapter 3, the exact scheduling analysis of a specific restricted model (i.e. Accumulatively Monotonic (AM) model), is given. The goals of this chapter is to present the basic response time formula of the AM multiframe tasks and show the performance of this exact scheduling analysis by a comparison with the most recent published, but non-optimal, schedulability analysis. Exact analysis in this chapter considers the situation where tasks share resources, which causes blocking to the MF tasks. Chapter 4 extends the analysis of the AM model, that is given in Chapter 3, to include blocking, release jitter and to cope with the arbitrary deadline scenario.

Chapter 5 relaxes the restriction of AM and presents the basic exact response time analysis of non-AM multiframe tasks, where the number of frames that have to be considered in such analysis is reduced using the critical frame concept. An evaluation of this analysis is given in this chapter by investigating the number of critical frames of randomly generated multiframe tasks. Further, this analysis is extended in Chapter 6 to again include blocking, release jitter and to cope with the arbitrary deadline scenario.

Chapter 7 presents an exact response time analysis of MF tasks, where each frame of a MF task has its own deadline which could be different from other deadlines of the frames of the same MF task. A new concept called *covering frames* is used in the analysis to reduce the number of frames that have to be analysed per MF task. An optimal priority assignment is also considered in this chapter.

As the schedulability analysis becomes intractable for large systems, Chapter 8 introduces four approaches for sufficient schedulability tests of systems with non-AM multiframe tasks. A comparison between those four approaches is presented in this chapter to show the percentage of their scheduling performance rates.

The final evaluations and conclusions of the contributions in this thesis are given in

Chapter 9. Further directions for future work are also presented in this chapter.

2 System Model and Related Work

This chapter defines the model of the basic system that is analysed in this thesis and provides a review for all related contributions to this thesis. The following section introduces the basic system model whilst Sections 2.2 and 2.3 present a historical review of the related work.

2.1 System Model

The basic system model that is considered in this thesis is a system that consists of N multiframe tasks that execute on a uniprocessor using the preemptive fixed priority scheduling policy. Each MF task τ_i consists, in its turn, of a sequence of n_i frames that are distinguished by their execution times; where a MF task, τ_i, has n_i worst case execution times, $C_i^k; k = 0..n_i - 1$. All frames in the same MF task have the same priority which is represented by the priority of the MF task and these priorities are assigned according to a priority assignment such as Rate Monotonic (RM) [52, 45] which is an optimal priority assignment for certain systems with MF tasks [57]. Priorities of the MF tasks in the system are ordered consecutively with τ_1 having the highest priority in the system and τ_N the lowest priority (i.e. 1 in τ_1 refers to the highest priority and N in τ_N refers to the lowest priority).

MF tasks in the system are permitted to share resources, so there could be a situation where the execution of a MF task is stopped by a lower priority task and we say that the MF task is blocked by a lower priority task. However, due to using some priority ceiling protocols, a MF task has an opportunity to be blocked at most once per invocation during its execution. So, we assume in the model that each MF task τ_i is considered to have a maximum blocking time equal to B_i. Further explanation for

blocking and priority protocols is given in Section 2.3.2. All system overheads such as context switch are ignored and assumed to be zero as we assume that there is an immediate switch between the MF tasks in the system.

Without loss of generality, we assume that the sequence of the execution time values is always within shortest form; where **the shortest form of a sequence** is the shortest sub-sequence when repeated a number of times generates the original sequence. This is because from the analysis point of view, the behaviour of the execution of a MF task whose execution times consist of repetitive subsequences is the same as the behaviour of the original sequence. For example, the execution behaviour of the MF task whose execution times are presented by the sequence $(8,1,4,3,8,1,4,3)$ is the same as the execution behaviour of the subsequence $(8,1,4,3)$. The extracted subsequence, $(8,1,4,3)$, is referred to as the shortest form of the sequence $(8,1,4,3,8,1,4,3)$.

Frames of the same MF task, τ_i, arrive in the system with minimum inter arrival time, T_i, and as soon as they have arrived, they are released having a relative deadline D_i. T_i is presented as constant for all frames of a MF task. So, a MF task τ_i is characterised by a triple $< C_i, T_i, D_i >$, where C_i is a vector of n_i values, $C_i = (C_i^0, C_i^1, .., C_i^{n_i-1})$, whilst T_i and D_i are vectors with one value. As an initial restriction on the model, D_i is considered to be less than or equal to T_i so no execution (i.e. interference) from the analysed task itself is considered when analysing its worst case response time.

Later on in Chapters 4, 6 and 7, the basic system model is extended from three points of view. Firstly, in Sections 4.1 and 6.1 the MF task τ_i is considered to have release jitter, J_i, so the minimum time between two successive releases of a MF task is less than the fixed time interval T_i. Secondly, in Sections 4.2 and 6.2 τ_i is considered to have $D_i > T_i$ so τ_i could have interference from previous frames during the execution of τ_i itself. Thirdly, in Chapter 7 each frame of a MF task has a deadline that could be different from other frames in the same MF task, so D_i is a vector of n_i values that are relative to the frames of the MF task, τ_i but no blocking or release jitter are considered in this chapter.

As this thesis is about the scheduling analysis of MF tasks from the worst case response time point of view, a definition of the symbol R_i is given in the following. R_i

of the MF task τ_i is defined as the longest time from when any frame of τ_i is released until it finishes its execution, so R_i has only one value per MF task τ_i. However, in Chapter 7 the MF task τ_i has n_i deadlines relative to each frame of τ_i, so R_i in this case is a vector of n_i values relative to the deadlines of τ_i.

To illustrate the problem of analysing the response time of MF tasks, Table 2.1 represents a simple example system with 2 tasks τ_1 and τ_2 where τ_1 is a MF task with 4 frames represented by the execution time values 8, 1, 4 and 3, and τ_2 has just one frame.

$task,\ \tau_i$	C_i	$T_i = D_i$	priority
τ_1	8, 1, 4, 3	10	1
τ_2	x	20	2

Table 2.1: Example System

Initial Frame Location	exe. seq.	1 inv.	2 inv.	3 inv.	4 inv.
0	8, 1, 4, 3	**8**	9	13	**16**
1	1, 4, 3, 8	1	5	8	**16**
2	4, 3, 8, 1	4	7	**15**	**16**
3	3, 8, 1, 4	3	**11**	12	**16**

Table 2.2: Possible Interference from τ_1

Finding the worst case response time R_2 of τ_2, whatever its execution time is, requires finding the maximum amount of possible interference from τ_1. Table 2.2 shows values of interference that τ_1 generates from different initial frames in the execution sequence (exe. seq. and inv. respectively stand for *execution time sequence* and *number of invocations*). It can be seen from Table 2.2 that the maximum amount of interference τ_1 generates, in the case of one invocation (i.e. 1 inv.), is when it is firstly released having an execution time of 8. While the maximum amount of interference, in the case of two invocations, is when it is firstly released having an execution time of 3 followed by 8. The maximum amount of interference, in the case of three invocations, is when τ_1 is firstly released having an execution time of 4 followed by 3

followed by 8. While, in the case of four invocations, the amount of interference from τ_1 remains the same (i.e. 16 in this example) whatever the release frame is.

Frames that could generate the maximum amount of interference are called *critical frames*; which are, in this example, frame whose execution times are 8, 4, and 3, but not 1 since any of 8, 4, 3 can be considered as a critical frame on behalf of 1 (full details of the reasons are given in Chapter 5). A frame of a MF task τ_j is considered as critical when it has two properties; firstly, it can generate the maximum amount of interference within lower priority task for at least one number of τ_j's invocations; and secondly there are no other frames in τ_j that generates greater or equal amount of interference for all possible number of τ_j's invocations.

So, to calculate the amount of interference a frame release generates within the response time of a lower priority task, we have to know the relative number of invocations (i.e. interference) the MF task is experiencing within this response time. For this reason we define a *cumulative function* of the x^{th} frame release[1] of a MF task τ_j to represent the amount of interference this frame generates. Definition 1 illustrates this cumulative function.

Definition 1 . *Given a MF task τ_j with n_j execution times $(C_j^0, C_j^1, .., C_j^{(n_j-1)})$. The* **cumulative function (ξ_j) of the x^{th} frame release for a given number of τ_j's invocations,** *k, is the amount of interference that the MF task generates starting from that frame and proceeding for that number of invocations and is given by Equation (2.1)*

$$\xi_j^x(k) = \sum_{f=x}^{x+k-1} C_j^{f \bmod n_j} \tag{2.1}$$

where $x = 0, .., n_j - 1,$ *and* $k = 1, 2, ..$, for example, the value of $\xi_1^0(2)$ for the MF task τ_1 in Table 2.1 is 9. In fact, for an ordinary single frame task the cumulative function is well defined as $\xi_j(k) = kC_j$ because of the constancy of C_j for all frames of the multiframe task.

From the criticality point of view, a frame in a MF task is considered critical when it can give rise to the maximum interference within lower priority tasks and so it can

[1] x^{th} frame release is the frame that is released with the x^{th} execution time of the MF task.

lead to the worst case response time of a lower priority task. On the other hand, when the cumulative function of a frame of a MF task is always greater than the cumulative function of all other frames of the same MF task for at least one possible number of interference, this frame definitely generates the maximum interference within lower priority tasks for that number of interference. The following definition formally introduces a condition on a frame of a MF task to be a definitely critical frame.

Definition 2 . The x^{th} frame of a MF task τ_j, whose execution time sequence is in its shortest form, is definitely critical *if* $\exists\, k = 1,2,..,n_j - 1, \forall y \neq x$:

$$\xi_j^x(k) > \xi_j^y(k) \tag{2.2}$$

For example, the first frame (i.e. the frame whose location is 0) of the MF task τ_1 in Table 2.1 is a critical frame because $\exists\, k = 1, \forall y \neq 0; \xi^0(1) > \xi^y(1)$.

We call the frame whose execution time is maximum the *Peak Frame*.

Definition 3 A Peak frame of a MF task *is one of the frames, in the MF task, whose execution time is the maximum of the execution times of this MF task.*

For example, the MF task τ, whose execution time sequence is $(8,4,8,3)$, has two peak frames with locations 0 and 2, where their execution times are both 8.

Note from Definition 2 that having the execution time sequence in its shortest form means that if we have more than one peak frame then at least one of the peak frames must be a critical frame; otherwise the execution time sequence is not in its shortest form. For example, in the above MF task τ whose two peak frames with locations 0 and 2, the first peak is critical but the other one is not.

Mok and Chen [56] force one of the peak frames of a MF task to be the only critical frame of this MF task by introducing the accumulatively monotonic, AM, condition on the execution time sequence. The AM condition depends on the peak frame being the only frame that generates the maximum amount of interference for all possible number of interference (i.e. invocations). Informally, all frames of the AM multiframe task are dominated by one of its peak frames. The AM restriction is mathematically

formalised by an equation using the *mod* function to reach the execution time values from its sequence. Equation (2.3) represents this AM restriction

$$\sum_{k=m}^{m+j} C^{(k \bmod n)} \geq \sum_{l=i}^{i+j} C^{(l \bmod n)}; \tag{2.3}$$

$$\forall i, j = 0, 1, 2, .., n-1;$$

where C^m is one of the peak values in a list of execution times $(C^0, C^1, .., C^{n-1})$ that satisfies Equation (2.3). For example, for the AM multiframe task whose execution time sequence is $C = (8, 4, 8, 3)$, $m = 0$ and $C^0 = 8$, also the frame whose execution time is C^0 is the only critical frame of this AM multiframe task.

2.2 Related Work to Scheduling MF Tasks within Fixed Priority Scheduling Scheme

The most popular scheduling tests for real-time systems within fixed priority policy are the utilisation test and the response time test. In the utilisation test, the system can be scheduled if the overall processor utilisation of the system is less than a pre-defined upper bound. In the response time test, the system can be scheduled if all its tasks meet their relative deadlines, and the task meets its deadline if its worst case response time is less than or equal to its relative deadline.

As this thesis is concerned with the worst case response time scheduling analysis of multiframe tasks within fixed priority policy, previous contributions within fixed priority scheduling policy must be covered within two fields. The first field is the contributions of scheduling MF tasks, which covers the contributions within the utilisation domain and other scheduling contributions related to MF tasks. The second field is response time analysis.

The MF model is a generalisation of Liu and Layland's model where in Liu and Layland's model the execution time of the task is constant for all its jobs, so the first contribution to start the review with is Liu and Layland's contribution. Liu and Layland [52] were the first who employed FPS on the uni processor system, the following

section explains Liu and Layland model.

2.2.1 Liu and Layland Contributions

Liu and Layland introduced a simple system model with the following assumptions:

1. tasks of the system are periodic, independent, fully preemptive and with no overheads;
2. no sharing of resources is permitted, so the runnable task is always the highest priority task;
3. all tasks are released at the beginning of their relative periods;
4. deadline of each task is equal to its period;
5. no task may suspend itself.

Worst case execution time of each task is considered as constant for all its jobs, so they do not vary from one invocation to another of the task. Tasks in this model are assigned priorities according to what is called Rate Monotonic, RM. In RM priority assignment, priorities are assigned to the tasks according to their periods; where the shorter period the task has, the higher priority it obtains. The executing task at a specific moment is the runnable task whose priority is the highest one. Liu and Layland [52] and Labetoulle [45] showed, for a single processor, that if a task set can be scheduled with any priority assignment it is scheduled with the RM assignment. In this sense RM is optimal.

Liu and Layland [52] and Serlin[65], with the RM algorithm for FPS, introduced a sufficient but not necessary utilisation scheduling test. The test was based upon the upper bound of the processor utilisation factor; where they proved that a task set is schedulable if its processor utilisation is less than or equal to a pre-defined upper bound. This test is represented by Equation (2.4).

$$\sum_{i=1}^{i=N} \frac{C_i}{T_i} \leq N(2^{\frac{1}{N}} - 1). \tag{2.4}$$

Where C_i stands for the execution time of the i^{th} task, T_i represents the period of the i^{th} task, and N is the number of tasks in the system. When the number of tasks N, becomes very large, the upper bound of the processor utilisation factor simplifies to 0.693. This utilisation scheduling test is inexact as it is sufficient but not necessary, hence it is pessimistic. For example assume we have a simple system with two tasks, each task has a worst case execution time equals half of its relative period (i.e. $C_1 = \frac{T_1}{2}$ and $C_2 = \frac{T_2}{2}$) and one of the periods is half of the other period (i.e. $T_2 = 2T_1$) then the task set, depending on Liu and Layland's test (i.e. Equation (2.4)), is unschedulable. However, the set is in practice schedulable as when the two tasks are released at the same time (which is the worst case situation) the first task executes for one half of its period and the second task executes for the other half of its period and both of them are schedulable. Lehoczky et al. [48] estimated the average maximum utilisation for rate monotonic fixed priority scheduling and they showed by simulation that this average is around 88% for uniformly distributed tasks.

Within the context of the preemptive system, the critical instance of a task is defined as the instant when this task is preempted the most so the processor is occupied the most with the execution of this task. Liu and Layland proved in their model that the critical instance, for any task, occurs, when the task is released simultaneously with all higher priority tasks in the system. So, the critical instance of the system is when all tasks in the system are simultaneously released at the same time.

However, this model restricts the worst case execution time for each task to be constant for all its jobs. In 1996 Mok and Chen [56, 57] relaxed this constancy restriction to introduce the multiframe model; and proposed a utilisation based schedulability test, for fixed priority scheduling, under RM priority assignment assuming the AM restriction for all multiframe tasks in the system. The following section covers Mok and Chen's contribution.

2.2.2 Mok and Chen Contribution

In Mok and Chen's model [56, 57], execution time values of each task in the system are not presented as a constant value any more. Instead the execution time values of each task are presented as a vector and the values of this vector satisfy the AM

restriction that is given by Equation (2.3). In the AM multiframe task, one of the peak frames always generates the maximum amount of interference within the execution of lower priority tasks, for any number of its invocations (i.e. interference). So, an AM task has only one critical frame which is the peak frame whose execution time satisfies Equation (2.3). For example, the critical frame of the multiframe task whose execution time sequence is $C = (8,4,8,3)$, is the first frame whose execution time is 8 (i.e. the 8 that is followed by 4 but not the 8 that is followed by 3).

In Mok and Chen's model, all jobs of a MF task are assigned the same priority which is called the priority of the MF task. Mok and Chen proved that the optimal priority assignment of a system with AM multiframe tasks is RM, where the lower period the MF task has the higher priority it is assigned. Also, they considered the critical instance of an AM multiframe task as the instant from when its critical frame is released simultaneously with the critical frames of all higher priority AM multiframe tasks. So, this AM multiframe task is schedulable if it is schedulable at its critical instance.

The main contribution of Mok and Chen was in the utilisation domain. They proved an upper bound for the peak utilisation of a system with AM multiframe tasks. They proved that the system is schedulable if its peak utilisation factor which is given by $U^m = \sum_{i=1}^{N} \frac{\max_{j=0}^{n_i-1} \{C_i^j\}}{T_i}$, is less than or equal to an upper bound given by $r.N.((\frac{r+1}{r})^{\frac{1}{N}} - 1)$. Equation (2.5) represents the schedulability test of a system with N AM multiframe tasks.

$$\sum_{i=1}^{N} \frac{\max_{j=0}^{n_i-1} \{C_i^j\}}{T_i} \leq r.N.((\frac{r+1}{r})^{\frac{1}{N}} - 1); \tag{2.5}$$

where r is the minimum ratio, over all AM multiframe tasks in the system, of the execution times of the critical frame and the frame that follows the critical. r is given by $r = \min_{i=1}^{N} \{r_i\}$; r_i, in its turn, is given by $r_i = 1$ if $N = 1$ or $r_i = \frac{C_i^0}{C_i^1}$ if $N > 1$. Note that Equation (2.5) returns to Liu and Layland's test when the execution times of each MF task are constant. This is because, for Liu and Layland's model, $r = 1$ as $C_i^0 = C_i^1$ and $\max_{j=0}^{n_i-1} \{C_i^j\} = C_i$.

Although Mok and Chen's utilisation test is an improvement test of Liu and Lay-

land, both tests are inexact (i.e. sufficient but not necessary) as well as being only applicable to RM priority assignment. However, Mok's utilisation bound has been improved by subsequent papers but these tests remain inexact. The following section covers subsequent contributions for scheduling MF tasks including the contributions that improved Mok and Chen's utilisation bound.

2.2.3 Subsequent Contributions for Scheduling MF Tasks

As Mok and Chen's test was the first scheduling test for MF tasks, Han [37] presented another scheduling test and compared its results with the results of Mok and Chen's test. Han's scheduling test [37] was also under RM priority assignment and was better than Mok's test in the sense that multiframe task sets with peak utilisation (i.e. the utilisation of the peak frame) larger than Mok's bound were not feasible using Mok and Chen's utilisation bound but can be found feasible by Han's test. The test was not based on utilisation test, it was based upon transforming the AM system to a system with harmonic periods, using a proposed algorithm for the transformation process, and then if the transformed system is schedulable, the original system is schedulable. Although Han showed by evaluation that his test is always better than Mok and Chen's test, Han's model restricts periodic AM multiframe tasks in the system while Mok's model is applied to sporadic AM multiframe tasks as well as periodic. However, both tests are inexact and only applicable to RM priority assignment as well as assume a non-flexible model as the model has to satisfy all restrictions of Liu and Layland's that are given in Section 2.2.1 apart from having non constant execution times and also all execution time sequences have to be AM.

Another scheduling test was given by Kuo et al. [44] who improved Mok's utilisation bound; where they gave another improved utilisation bound for a schedulability test of systems with AM multiframe tasks. The main idea of the test was to merge the tasks whose periods are harmonic (i.e. one of the period is a multiple of the others) to reduce the number of tasks that has to be considered in the schedulability test and then apply Mok's bound to the merged tasks. The combined task, under Kuo's test, will have a period of $\hat{T}$ and a sequence of execution times $\hat{C}^j$ with the size $\hat{n}$; where $\hat{T}$ is the maximum period of the merging tasks, $\hat{n}$ is the least common multiple of the number

of execution times of the merged tasks, and $\hat{C}^j$ is given by the following formula

$$\hat{C}^j = \sum_{i=1}^{N} (\sum_{k=0}^{(\frac{\hat{T}}{T_i})-1} C_i^{(j(\frac{\hat{T}}{T_i})+k) \mod n_i});$$

where $j = 0, 1, .., \hat{n}-1$, N is the relative number of tasks that are under merging procedure, n_i and T_i are respectively the relative number of frames and the relative period of the i^{th} AM multiframe task. The example below gives more explanation about these calculations.

In 2007, Lu et al. [55] improved Kuo's utilisation test and presented new scheduling conditions for AM multiframe tasks within the utilisation domain and assuming RM priority assignment. They considered the ratio of the periods in their test. The improvement was that they used Kuo's method to merge the tasks and then they applied their test to the merged tasks. The schedulability status, under their approach, depends on the total peak utilisation, U, of the AM multiframe tasks being less than a defined upper bound. They called this upper bound the *Conditional Bound* function, *CB*. Symbolically, the AM task set is schedulable if inequality (2.6) is satisfied.

$$U \leq CB; \tag{2.6}$$

where the total peak utilisation, U, is the summation of all peak utilisations of the multiframe tasks in the system; and it is given by

$$U = \sum_{i=1}^{\hat{N}} \max_{0 \leq j \leq n_i - 1} \{ \frac{\hat{C}_i^j}{\hat{T}_i} \}.$$

Whilst the *CB* function is defined by Equation (2.7); for number of tasks, $N > 1$, and with regard to two parameters r and z.

$$CB(r,z) = z + r(z-1) + r(\hat{N}-1)((\frac{1}{z})^{\frac{1}{\hat{N}-1}} - 1) \tag{2.7}$$

where $\hat{n}_i$ and $\hat{T}_i$ are respectively the number of frames and the period of the i^{th} MF task. r is given as
$r = \min_{1 \leq i \leq \hat{N}} \{r_i\}$, where r_i is defined depending on $\hat{n}_i$ as

$r_i = \frac{\hat{C}_i^0}{\hat{C}_i^1}$; $for\ \hat{n}_i > 1$, and $r_i = 1$; $for\ \hat{n}_i = 1$.
z is given as
$z = \max\{\min_{1 \le i \le \hat{N}-1}\{\frac{V_i}{\hat{T}_{\hat{N}}}\}, \frac{r}{1+r}\}$, where V_i is called a virtual period and is given by $V_i = \lfloor \frac{\hat{T}_N}{\hat{T}_i} \rfloor \hat{T}_i$.

Section 3.4 in Chapter 3 compares between the response time scheduling test of AM multiframe tasks and Lu's scheduling test as Lu's analysis is the most recent published scheduling analysis for MF tasks within FPS. So, we fully illustrate Lu's test by the following detailed example to give more explanation of the test.

Example

Table 2.3 represents an example system that consists of five tasks with their attributes.

$task$	C	$T = D$
τ_1	(1)	3
τ_2	(2)	9
τ_3	$(3,1)$	18
τ_4	$(2,1)$	20
τ_5	$(6,3)$	60

Table 2.3: Example Illustrates Lu's Analysis- Original System's Attributes

$task$	$\hat{C}$	$\hat{T}$
$\hat{\tau}_1$	$(7,5)$	18
$\hat{\tau}_2$	$(31,27)$	60

Table 2.4: Merged System Using Kuo's Method

Using Lu's approach, τ_1, τ_4, and τ_5 are merged using Kuo's method [44] to $\hat{\tau}_2$ with a period equal to the maximum period of T_1, T_4 and T_5; which is 60 in this example. $\hat{\tau}_2$ has number of execution times equal to the least common multiple of n_1, n_4 and n_5; which is 2 in this example. Values of $\hat{\tau}_2$'s execution times are found by applying

$$\hat{C}_2^j = (\sum_{k=0}^{(\frac{60}{T_1})-1} C_1^{(j(\frac{60}{T_1})+k) \ mod \ n_1}) + (\sum_{k=0}^{(\frac{60}{T_4})-1} C_4^{(j(\frac{60}{T_4})+k) \ mod \ n_4}) + (\sum_{k=0}^{(\frac{60}{T_5})-1} C_5^{(j(\frac{60}{T_5})+k) \ mod \ n_5})$$

$$for\ j = 0, 1.$$

Therefore, $\hat{C}_2^0 = 31$ and $\hat{C}_2^1 = 27$. Also, τ_2 and τ_3 are merged, using Kuo's method, to $\hat{\tau}_1$ with the number of execution time equal to $\hat{n_1} = 2$ and execution time values

$\hat{C}_1 = (7,5)$ and a period of 18. Table 2.4 represents the attributes of the merged tasks.

Once the merged tasks are identified, the scheduling test is to check if the total peak utilisation, U, is less than or equal to a pre defined conditional bound, CB. U, is the summation of all peak utilisations of the multiframe tasks in the system; and it is given by

$$U = \sum_{i=1}^{2} \max_{0 \le j \le \hat{n}_i - 1} \{\frac{\hat{C}_i^j}{T_i}\} = \frac{7}{18} + \frac{31}{60} = 0.905.$$

CB is found depending on two parameters r and z.

r is given as $r = \min_{1 \le i \le \hat{N}} \{r_i\}$, where r_i is the ratio of the first two execution times of $\hat{\tau}_i$ and is defined by
$r_i = \frac{\hat{C}_i^0}{\hat{C}_i^1}, so\ r_1 = \frac{7}{5}, r_2 = \frac{31}{27}$.
Therefore, $r = \min\{\frac{7}{5}, \frac{31}{27}\} = 1.148$.
z is given by $z = \max\{\min_{1 \le i \le \hat{N}-1}\{\frac{V_i}{T_{\hat{N}}}\}, \frac{r}{1+r}\}$, where V_i is called a virtual period and is given by
$V_i = \lfloor \frac{\hat{T}_N}{\hat{T}_i} \rfloor \hat{T}_i = \lfloor \frac{60}{\hat{T}_i} \rfloor \hat{T}_i$. So, $V_1 = \lfloor \frac{60}{18} \rfloor 18 = 54$.
Therefore, $z = \max\{\frac{54}{60}, \frac{1.148}{1+1.148}\} = 0.9$.

Once r and z are identified, $CB(r,z)$ is given by

$$\begin{aligned} CB(r,z) &= z + r(z-1) + r(\hat{N}-1)((\tfrac{1}{z})^{\frac{1}{\hat{N}-1}} - 1) \\ &= 0.9 + 1.148(0.9-1) + 1.148(2-1)((\tfrac{1}{0.9})^{\frac{1}{2-1}} - 1) \\ &= 0.912. \end{aligned}$$

Therefore, the total peak utilisation of the system is less than the conditional bound function (CB) of the merged tasks (i.e. $U < CB$) which means using Lu's test that the original system that is given by Table 2.3 is schedulable.

Moving on to non-AM multiframe tasks, Takada et al. [69] investigated the schedulability of the general MF tasks and gave a necessary and sufficient condition for the schedulability of the MF model, under the fixed priority scheme. They showed that the complexity of the feasibility decision becomes at least[2] $\prod_{i=1}^{N} n_i$. They also introduced an efficient feasibility decision algorithm using a maximum interference function. However, Takada's estimation of the complexity of the exact analysis is pes-

[2] $\prod_{i=1}^{N} n_i$ means the product of all numbers of frames over all tasks in the system.

simistic as we show in Chapter 5 that the complexity of the exact scheduling analysis is $\prod_{i=1}^{N}(n_i - 1)$ in the worst case. Also, his test was applicable to a restricted model where the deadline of the task should not extend beyond its period.

Baruah et al. [13] used the fixed point approach motivated by the response time analysis to give a tractable but sufficient schedulability test for a system of general MF tasks. They preprocessed the execution time sequences of the MF tasks taking into account the maximum amount of interference that higher priority MF tasks provide. Then, they apply the fixed point algorithm to estimate the worst case response time of the peak frame of the lower priority MF task considering the maximum amount of interference each higher priority MF task can provide. Although this analysis is in some sense related to response time domain, the test is inexact as it estimates the maximum interference before processing the response time analysis; while in our contribution we provide an exact analysis of the response time. However, an approach called complementary approach; which is equivalent to Baruah et al.'s approach is presented in Chapter 8 in this thesis.

Baruah et al. [12] also did some work in scheduling multiframe tasks related to Earliest Deadline First, EDF, scheduling scheme; which is an alternative scheduling scheme. However, this thesis is concerned with FPS so this EDF approach is not expanded upon here.

As can be seen from the above contributions, all schedulability analyses are inexact as all of them are either in the utilisation domain or only sufficient. For example, Lu's analysis improves previous results, but still remains inexact as well as it is dependent on the RM priority assignment. Moreover, their test is only applicable to a system whose deadlines are identical to their relative periods. Whilst response time analysis that is presented in this thesis gives an exact scheduling analysis for less strict models (systems with sharing resources, release jitter and arbitrary deadlines) and is applicable to any priority assignment.

2.3 Contributions of Response Time Analysis

Most research within fixed priority scheduling assume RM as an optimal priority assignment assuming deadlines of the tasks are identical to their relative periods. However, if deadlines of the tasks are permitted to be less than their relative periods RM priority assignment is not optimal any more [41] and Deadline Monotonic, DM, takes the place [51] (the smaller relative deadline the task has the higher priority it is assigned). So, as most utilisation scheduling tests depend on RM priority assignment or restrict the system to satisfy most of Liu and Layland's assumptions, studying the schedulability of a system from the utilisation point of view is not flexible enough to be extendable to the systems with sharing resources, release jitter, and arbitrary deadlines. However, Harter [58] solved this problem by introducing the idea of analysing the schedulability of a system using worst case response time analysis.

2.3.1 Basic Response Time Analysis

Basically, analysing the worst case response time of a task τ_i within Liu and Layland's model can be achieved once three issues are identified: τ_i's critical instance, τ_i's amount of execution and the amount of execution of tasks other than τ_i. Joseph and Pandya [40] followed by Audsley et al. [9] mathematically applied response time analysis and introduced an iterative equation, Equation (2.8), for finding the worst case response time of a task τ_i assuming the basic model of Liu and Layland (see Section 2.2.1 for details). They assumed Liu and Layland's critical instance [52]; where the worst case response time, of a task is when this task is released simultaneously with all higher priority tasks.

$$R_i = C_i + I_i = C_i + \sum_{j \in hp(\tau_i)} \lceil \frac{R_i}{T_j} \rceil C_j \tag{2.8}$$

$hp(\tau_i)$ is the set of tasks whose priorities are higher than the priority of τ_i. As τ_i is a preemptive task, $I_i = \sum_{j \in hp(\tau_i)} \lceil \frac{R_i}{T_j} \rceil C_j$ represents the maximum amount of interference from higher priority tasks within the execution of τ_i. In other words, I_i represents the maximum amount of interference within the worst case response time of τ_i, from the

tasks whose priorities are higher than the priority of τ_i.

As the priorities of the tasks are assigned from 1 being the highest priority and N is the lowest, $hp(\tau_i)$ returns to the values $1, .., i-1$. So, Equation (2.8) is rewritten to be as Equation (2.9)

$$R_i = C_i + I_i = C_i + \sum_{j=1}^{i-1} \lceil \frac{R_i}{T_j} \rceil C_j. \tag{2.9}$$

To solve Equation (2.9), a recurrence relation is given as in Equation (2.10); where $l = 0, 1, 2, ...$ and $R_i^0 = C_i$. The smallest non-negative solution of Equation (2.10) represents the worst case response time of τ_i, R_i. In other words, the worst case response time is obtained when it is found that $R_i^{l+1} = R_i^l = R_i$ (for the smallest value of l). However, in the case that R_i^{l+1} becomes greater than the deadline of τ_i, then τ_i is not guaranteed to meet its deadline, so we say that the task is unschedulable.

$$R_i^{l+1} = C_i + \sum_{j=1}^{i-1} \lceil \frac{R_i^l}{T_j} \rceil C_j. \tag{2.10}$$

Equation (2.9) assumes that there is no sharing resources between the tasks, so only the runnable task, τ_i, can access the resource. In fact, there are situations where τ_i asks for resources that are occupied by tasks whose priorities are lower than τ_i, so τ_i can not access this resource until the lower priority tasks give up this resource. In this case we say that τ_i is blocked awaiting lower priority tasks to finish their execution. The following section gives details about response time analysis of tasks with blocking.

2.3.2 Tasks with Blocking Time

To explain the blocking scenario, assume there are two tasks τ_1 and τ_2 attempting to access shared data (τ_1 has higher priority than τ_2). If τ_2 gains access first and then τ_1 request access to the shared data; the higher priority task τ_1 would be blocked until the lower priority task τ_2 completes its access to the shared data. Blocking in this example is a form of priority inversion; where τ_2 completes its execution with a priority higher than or equal to τ_1, as τ_2 executes before τ_1 whilst τ_1 actually has higher priority than τ_2.

Long duration of blocking could lead to missed deadlines of the task and so the system could be unschedulable as, in some cases, a low priority task may unnecessarily block the execution of higher priority tasks. So, researchers in this area attempt to minimise this blocking time to reduce the chance of missing the timing requirements. This minimisation was achieved by introducing some priority inheritance protocols.

Lampson and Redell [47] were the first who discussed priority inheritance in the context of monitors. Each monitor was associated with the priority of the highest priority task which enters that monitor. Then, whenever a task enters a monitor, its priority increases temporarily, to the monitor's priority.

In 1990, Sha and his colleagues [66] gave two protocols to minimise this blocking time, basic priority inheritance protocol and priority ceiling protocol . The following are the details of these protocols.

Basic Priority Inheritance Protocol (BPIP)

The basic priority inheritance protocol is described as following: when a task τ_k blocks one or more higher priority tasks, it ignores its original priority and executes the critical section with the highest priority level of all tasks τ_k blocks. After exiting its critical section, τ_k returns to its original priority level. Sha and his colleagues proved in their work that, under BPIP, if there are m semaphores that can block τ_i, then τ_i can be blocked at most m times.

Priority Ceiling Protocols (PCP)

In the priority ceiling protocols, priorities of the tasks at run time are not strictly fixed, although priorities of the tasks and resources are assigned before run time. The best known two priority ceiling protocols are the original ceiling priority protocol and the immediate ceiling priority protocol.

In the original ceiling priority protocol [66], each resource has a static ceiling value which is the maximum priority of the tasks that use this resource. Whilst the task that shares the resources has two kinds of priorities one of them is fixed which is the original default priority and the other is dynamic which copes with the execution of the critical sections. The dynamic priority of the task is the maximum of its own default priority and any it inherits due to blocking higher priority tasks. A task can lock a resource if its dynamic priority is higher than the ceiling of any currently locked resource. The benefit of the original ceiling priority protocol is that once the task is

released it will be blocked at most once during its execution.

In the immediate ceiling priority protocol, each resource has a static ceiling value which is the maximum priority of the tasks that use this resource. Whilst a task that shares the resources has two kinds of priorities one of them is fixed which is the original default priority and the other is dynamic which copes with the execution of the critical sections. The dynamic priority of the task is the maximum of its own default priority and the ceiling values of the shared resources. Priority of the task at run time is chosen according to its dynamic priority. The benefit of the immediate ceiling priority protocol is that the task could be blocked at most once at the beginning of its execution.

The immediate priority ceiling protocol was derived from the basic protocol for incorporation in programming languages and operating system standards. For example it is available in Ada, in POSIX (where it is known as the Priority Protect Protocol) and Real-Time Java (where it is known as Priority Ceiling Emulation) [25]. Immediate ceiling priority protocol is a significant protocol for tasks executing on a uni processor because applying immediate ceiling protocol to a uni processor system with sharing resources allows the task to be blocked at most once at the beginning of its execution. This is because once a task τ_i requires an occupied resource, τ_i's priority increases to the maximum ceiling value of the shared resources. So, once the resource becomes free, τ_i access it and completes its execution with its dynamic priority without any interruption from any lower priority task. In addition, Pilling, Burns and Raymond [60] proved formally that immediate ceiling protocol prevents the deadlocks[3]. Also, immediate ceiling protocol prevents transitive blocking as the task returns to its default priority after finishing the execution of its critical section.

In 1991, Baker [10] extended the PCP to the Stack Resource Policy, SRP, that supports three issues: multiunit resources, sharing runtime stack resources, and EDF as well as FPS, schemes. SRP depends on the preemption level of the task; which might be its priority in some cases. As this thesis uses PCP rather than SRP no more details of SRP are introduced.

[3] In the deadlock situation, a task is blocked forever as it and another tasks are waiting each other to finish its critical section, and thus neither ever does.

Adding Blocking Time to the Response Time Analysis

We showed in the previous discussion that PCP allow the task to be blocked at most once during its execution, so the worst case response time formula of the task that is subjected to blocking must take into account the maximum expected blocking time. Audsley et al. [9] enhanced the response time equation (i.e. Equation (2.9)) to include the maximum blocking time, B_i, as in Equation (2.11) assuming the PCP.

$$R_i = C_i + B_i + \sum_{j=1}^{i-1} \lceil \frac{R_i}{T_j} \rceil C_j \tag{2.11}$$

Similar to how Equation (2.9) is solved, Equation (2.11) is solved by forming a recurrence relation as in Equation (2.12); where $l = 0, 1, 2, \ldots$ and $R_i^0 = C_i$. The smallest non-negative solution of Equation (2.12) represents the worst case response time of τ_i. In other words, the worst case response time of τ_i is obtained when it is found that $R_i^{l+1} = R_i^l = R_i$ for the smallest value of l. However, in the case that r_i^{l+1} becomes greater than the deadline of τ_i, τ_i is not guaranteed to meet its deadline, so we say that the task is unschedulable.

$$R_i^{l+1} = C_i + B_i + \sum_{j=1}^{i-1} \lceil \frac{R_i^l}{T_j} \rceil C_j \tag{2.12}$$

2.3.3 Tasks Subjected to Release Jitter

One of the flexibilities of the response time scheduling test is being applicable to tasks that are subjected to *Release Jitter*. A periodic task τ_j has to arrive in the system within a fixed time which is its period, T_j, then it will be released as soon as it arrives. However, when this periodic task τ_j is subjected to release jitter, its arrival time becomes under some circumstances different from its release time. So, τ_j does not become strictly periodic and a variation in its release times has arisen. So, release jitter of a task τ_j is defined as the maximum variation in τ_j's release times [39]. To clarify, when τ_j is subjected to release jitter, its release times take place somewhere within time interval of length J_j and then every period T_j. Mathematically, let s_j^k be

the time when the k^{th} release of τ_j takes place, then

$$k.T_j + x \leq s_j^k \leq k.T_j + y; \ \forall k \in \mathrm{Z} \tag{2.13}$$

$$where\ J_j = y - x.$$

More explanations and diagrams are given in Chapters 4 and 6 when a generalised model of MF tasks that are subjected to release jitter is analysed.

The problem of release jitter happens when the task is not released as soon as it arrives [71, 70]; which mostly happens within two popular situations represented by "end to end jitter" [61] and granularity of the system timer. The situation of "end to end jitter" is an important issue to be considered in distributed systems as a task could be delayed awaiting the arrival of a periodic message that is not delivered completely regularly. Whilst the situation of granularity of the system timer is an important issue to be considered in uni processor systems. The following is an illustration of both situations of granularity of the system timer and "end to end jitter".

From the granularity of the system timer point of view, in some cases, the granularity of the system timer forces the periodic task to experience release jitter because of the bounded time the scheduler mechanism takes to recognise the arrival of a task [9]. For instant, a task with period of 10 but a system granularity of 9 will imply a jitter value of 8 at time 18 the periodic task will be released for its 2^{nd} invocation.

From end to end jitter point of view, the following example clarifies this phenomenon, assume there are, on different processors, two related periodic tasks: τ_f and τ_j with the same period. Task τ_f calls τ_j as soon as τ_f has finished its execution. Due to system load, τ_f does not finish its first execution until the end of its period; while it executes at the very beginning of its next period. As a result, τ_j is released twice within its period instead of once (i.e. the time between the two successive frames of τ_j, on the processor τ_j is executing on, is less than the usual minimum inter arrival time of the task τ_j). It is obvious that as a result of this scenario, the amount of interference from task τ_j, on a lower priority task τ_i on the same processor, may be greater than that assumed for with a purely periodic task.

As an estimation of jitter, some researchers [36] considered the optimal instant of

the task, that was presented by Bril et al. [21], and derived an upperbound for jitter considering the best case response time (BCRT). An optimal instant of a task occurs when the completion of the task coincides with a simultaneous release of all higher priority tasks. BCRT, in its turn, was defined as the minimum response time of a task. Figure 2.1 explains the optimal instant of τ_3 for a system with three tasks with attributes in Table 2.5.

task	C	$T = D$
τ_1	3	10
τ_2	11	19
τ_3	5	56

Table 2.5: Example System

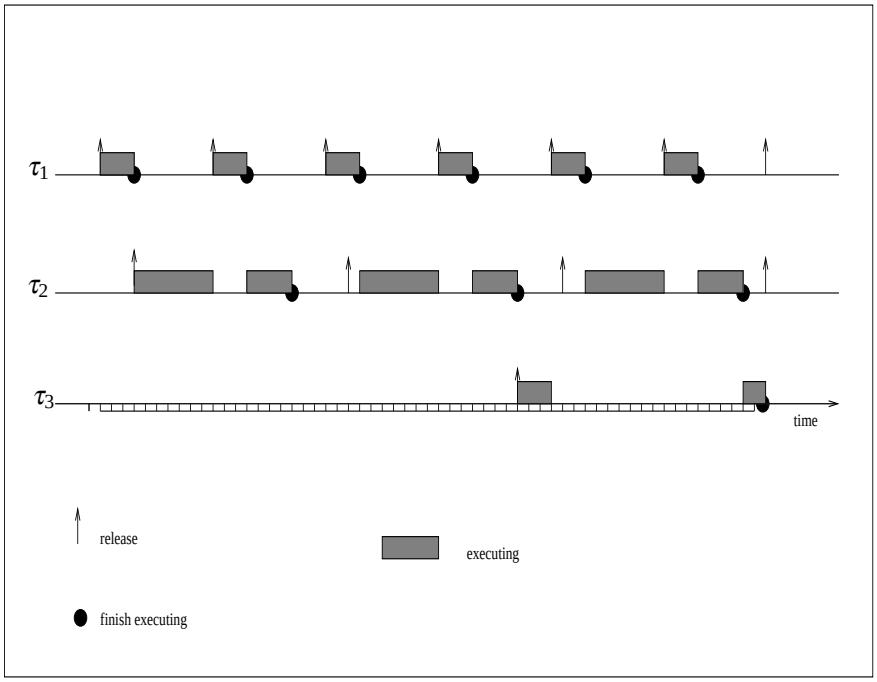


Figure 2.1: Optimal Instant Situation of τ_3

Kim et al. [42] and Bril et al. [21] enhanced the best case response time analysis and gave simpler best case response time equation. However, this thesis is concerned with the worst case response time analysis, hence no need for further details about best case response time analysis.

Release jitter analysis can also be used for predicting the behaviour of deferrable servers [18] and devices such as Bus Gurdians [23].

Worst Case Response Time for Tasks Subjected to Release Jitter

As tasks with release jitter are not purely periodic, the worst case response time formula (i.e. Equation (2.9)) requires modification to cope with the release jitter situation. The first issue to be considered in any response time analysis is to identify the critical instance of the analysed task τ_i. Tindell [71, 70] identified the critical instance of τ_i within the release jitter situation as when τ_i is released at the same time as when higher priority tasks finish waiting. For example, consider a system with the attributes in Table 2.6. Figure 2.2 represents the critical instance of the task τ_2.

Task	*T*	*C*	*J*
τ_1	11	5	2
τ_2	13	4	4

Table 2.6: Tasks Description's

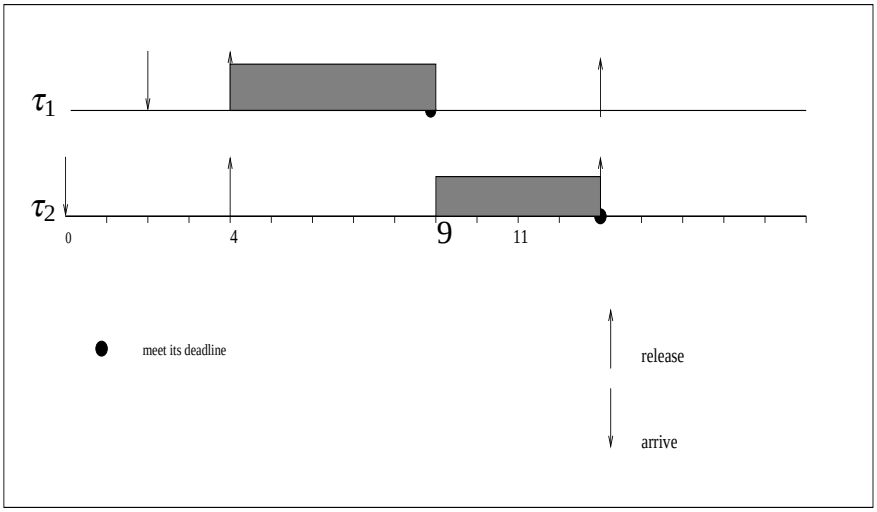


Figure 2.2: Execution of τ_1 and τ_2 at the Critical Instance of τ_2

As the interference from higher priority tasks could be increased by release jitter, the required modification within the response time formula is at the side I_i of the

response time formula (i.e. Equation (2.9)). Audsley and his colleagues [9] modified the side of the interference and gave a complete formula for the response time that takes into account release jitter situation. They gave Equation (2.14) that represents the interference on the worst case response time of τ_i, R_i, from all higher priority tasks assuming Tindell's critical instance.

$$I_i = \sum_{j=1}^{i-1} \lceil \frac{R_i + J_j}{T_j} \rceil C_j. \tag{2.14}$$

So, the worst case response time of τ_i is presented in Equation (2.15)

$$R_i = C_i + B_i + I_i. \tag{2.15}$$

Solving Equation (2.15) is similar to Equation (2.10) by forming a recurrence relation and once $R_i^{l+1} = R_i^l$ has been found, the worst case response time of τ_i is $R_i = R_i^{l+1}$. Schedulability of τ_i is guaranteed if $R_i \leq D_i - J_i$, however, if R_i^{l+1} becomes greater than $D_i - J_i$, the task is not guaranteed to meet its deadline so we say that τ_i is unschedulable.

Optimal Priority Assignment for Tasks with Release Jitter

Although the response time formula is applicable to any priority assignment, an interesting issue to mention in fixed priority scheduling for tasks with release jitter is that neither deadline monotonic nor rate monotonic priority assignments are optimal in the case of release jitter. Priorities are assigned according to the optimal priority assignment technique that depends on feasibility. Audsley[5], in his report, covered this technique, which is explained, in summary, as following. For a task set $S = \{\tau_1, \tau_2, ...\tau_N\}$, firstly, attempt to find a task τ_A that is feasible at priority level $j = N$. Next, find a feasible task at priority $j = N-1$. Successively, feasible tasks will be found at priorities N to 1. If a feasible task, at priority level i, could not be found, no feasible priority assignment function exists. Full details can be found in Audsley's report [5].

However, Burns et al. [24] mentioned, without proof, that for tasks that are subjected to release jitter, priorities should be assigned according to (D-J) since DM is no

longer optimal. In (D-J)-Monotonic priority assignment, the lower value of $(D-J)$ the task has, the higher priority it is assigned. Aproof of the optimality of (D-J)-Monotonic priority assignment is given in the appendix[4].

2.3.4 Tasks with Arbitrary Deadlines

Up to this point, contributions within fixed priority scheduling have been covered for a system model whose tasks are assumed to have deadlines less than or equal to their relative periods. So the response time of the analysed task does not need to take into account interference from the analysed task itself as it is not released during its execution. However, some contributions have been done for systems whose tasks have deadlines greater than their relative periods.

Lehoczky [49] proved that the critical instance of a task within the arbitrary deadline model is the simultaneous release of the task itself and higher priority tasks. He also introduced a sufficient but not necessary feasibility tests based upon utilisation. The test was an extended utilisation test of Liu and Layland's test with the restriction that all tasks have deadlines equal to multiple of their periods. However, as this thesis is interested in response time scheduling, no further details of scheduling within the utilisation domain for arbitrary deadline model are given; whilst response time scheduling is covered.

To illustrate the arbitrary deadline scenario, Figure 2.3 represents the timeline diagram of a small example system that is given by Table 2.7. The system consists of two tasks τ_1 andτ_2; where the deadline of τ_2 extends beyond its period.

task	C	T	D
τ_1	2	5	5
τ_2	4	7	8

Table 2.7: Example System of Arbitrary Deadlines

Figure 2.3 shows how τ_2's second invocation has interference from τ_2's first invocation; where the second release of τ_2 does not start its invocation until its first invocation

[4]This proof was also published in [76].

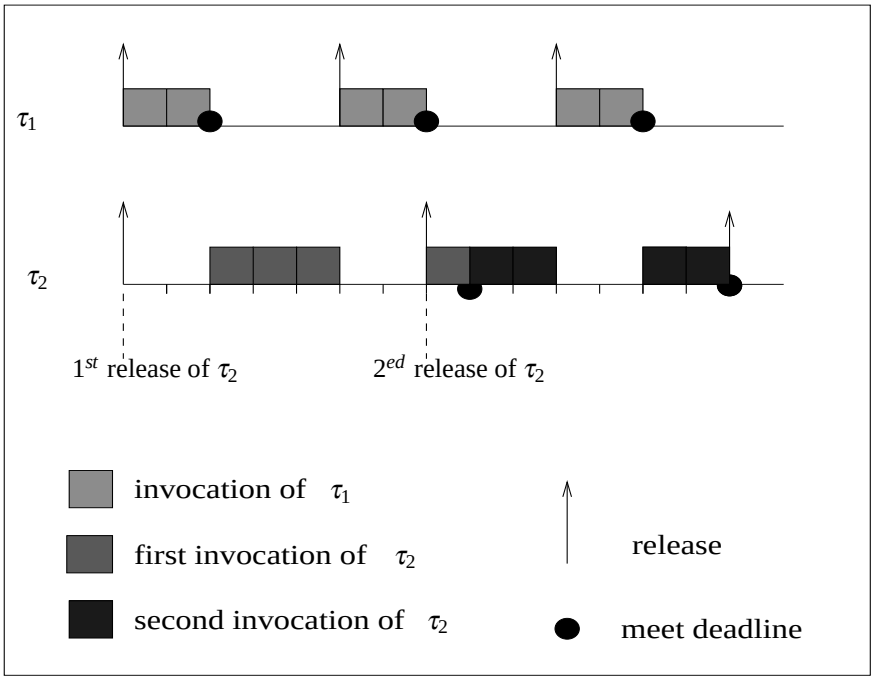


Figure 2.3: Timeline Diagram of the System in Table 2.7

has finished. So, response time of τ_2 has to take into account the interference from the task itself as well as interference from higher priority tasks. In other words, response time formula (i.e. Equation (2.9)) requires modification to cope with the tasks whose deadlines are greater than their periods.

Tindell [70, 71, 72] modified Equation (2.9) and analysed the response time of tasks with arbitrary deadlines, blocking, and release jitter within the same model. Analysis of the response time of τ_i is summarised in five steps. In the first step, define the busy period of a task as the time from when this task is released until it finishes the execution that is related to this release. In the second step, define q and $r_i(q)$ as the number of invocations of τ_i and the length of the continuous q busy periods respectively. In the third step, $r_i(q)$ is found by a recurrence relation as in Equation (2.16).

$$r_i(q) = B_i + qC_i + \sum_{\forall j \in hp(i)} \lceil \frac{r_i(q) + J_j}{T_j} \rceil C_j \tag{2.16}$$

Solving Equation (2.16) is achieved by forming an iterative equation as in Equation (2.17).

$$r_i^{l+1}(q) = B_i + qC_i + \sum_{\forall j \in hp(i)} \lceil \frac{r_i^l(q) + J_j}{T_j} \rceil C_j \tag{2.17}$$

where $r_i^0 = qC_i$ and $l = 0, 1, ..$ until $r_i^{l+1} = r_i^l$. However, if $r_i^{l+1}(q) - (q-1)T_i > D_i - J_i$, τ_i is not guaranteed to meet its deadline and we say that τ_i is unschedulable.

Forth step of the analysis is to find all needed busy periods for the analysis. Assuming $w_i(q)$ is the q^{th} busy period of τ_i, $w_i(q)$ is given by Equation (2.18).

$$w_i(q) = r_i(q) - (q-1)T_i + J_i \tag{2.18}$$

q is a finite integer value starting from 1 until no further interference from τ_i occurs; which happens when the busy period of τ_i finishes within the period it is released in. In other words, $q = 1, 2, ..$ until condition (2.19) is satisfied.

$$w_i(q) \leq T_i - J_i \tag{2.19}$$

Once all busy periods of τ_i are identified, the last step of the analysis is to find the worst case response time of τ_i, R_i, by maximising the busy periods over all number of its possible invocations.

$$R_i = \max_{q=1,2,..} \{w_i(q)\} \tag{2.20}$$

The following simple numeric example clarifies how to apply response time analysis to tasks with arbitrary deadlines. Suppose a example system in Table 2.7, for simplicity of the explanation we assumed all blockings and jitter in the example are zero. To analyze the response time of τ_2, we begin with finding r_2 and w_2 of τ_2 by applying Equations (2.17) and (2.18) respectively for different values of q.

$q = 1$

$r_2^0(1) = 4$

$r_2^1(1) = 4 + \lceil \frac{4}{5} \rceil 2 = 6$

$r_2^2(1) = 4 + \lceil \frac{6}{5} \rceil 2 = 8$

$r_2^3(1) = 4 + \lceil \frac{8}{5} \rceil 2 = 8$

Therefore, $r_2(1) = 8$. So, $w_2(1) = 8 - 0(7) = 8$. As $w_2(1) > T_2$, we increase q to be 2 and find $r_2(2)$ and w_2 by applying Equations (2.17) and (2.18) respectively.

$q = 2$
$r_2^0(2) = 8$
$r_2^1(2) = 8 + \lceil \frac{8}{5} \rceil 2 = 12$
$r_2^2(2) = 8 + \lceil \frac{12}{5} \rceil 2 = 14$
$r_2^3(2) = 8 + \lceil \frac{14}{5} \rceil 2 = 14$. Therefore, $r_2(2) = 14$. So, $w_2(2) = 14 - 1(7) = 7\ w_2(1) \leq T_2$; which satisfies the condition of Equation (2.19), so we stop increasing the values of q. Therefore, by applying Equation (2.20), we find that the worst case response time of τ_2 is $R_2 = max\{8, 7\} = 8$

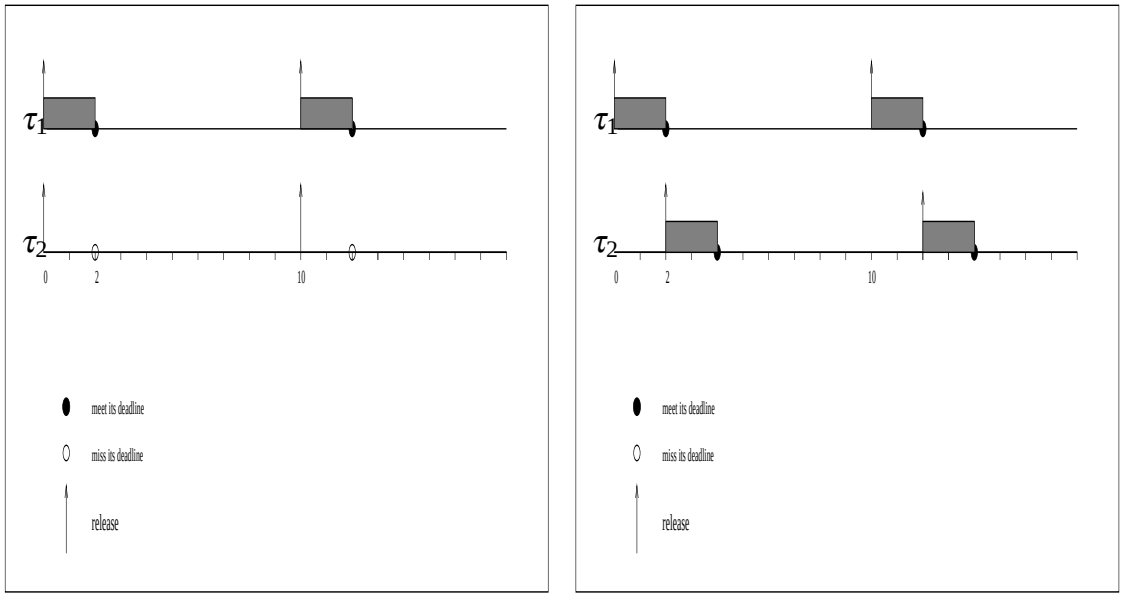
Analysis in this section assumes that each task in the system has constant execution time. However, Section 4.2 in Chapter 4 and Section 6.2 in Chapter 6 relax the restriction of constant execution time and present full analysis of the worst case response time of MF tasks within arbitrary deadlines.

2.3.5 Tasks with Offsets

Fixed priority scheduling contributions that have been mentioned so far consider system models where the critical instance of a task is the simultaneous release of the task itself and all higher priority tasks. In 1980, Leung and Mirrell [50] generalised Liu and Layland's [52] model from the point of view that all tasks are not always released at the beginning of their relative periods. Instead, the first invocation of each task in the system is allowed to have a specific offset and then the other invocations (i.e. second, third, ..) are released at the beginning of the relative period. The motivation behind the offset model is to increase the feasibility of the system. For example, a system with two tasks, that have periods of 10, execution time of 2 and deadline of 2, is unschedulable if the tasks do not have offsets, but the system is schedulable if either tasks has an offset equals to 2. Figure 2.4 illustrates how both τ_1 and τ_2 are schedulable when τ_2 has an offset equals to 2.

Leung and Mirrell [50] gave an interval of the scheduling analysis duration, of a task τ_i, that was improved later on by Audsley [6].

In the offset analysis, many researchers used a concept called transaction; that is a collection of related tasks and each task, that is a member of the transaction, has



(Case 1: $Offsets = 0$) (Case 2: τ_2 has Offset = 2)

Figure 2.4: Usage of Offsets for Increasing Schedulability

a relative offset. Tindell [70] gave an exact but not tractable test for a system with offsets. The intractability problem comes from the fact that the critical instance of a system with offsets is hard to identify and it is no longer as in Liu and Layland's model. The second case in Figure 2.4 is an example to deny Liu and Layland's critical instance. So, the basic feasibility analysis can not be applied directly to a system with offsets. Bate[16, 17] presented a tractable but non-exact "composite" approach to analyze task sets featuring offsets. The approach depends on transferring the system into another one by composing, according to a specific algorithm, tasks with non-zero offsets and the same period; into one task with zero offset. The benefit of the composite task approach is that the computational complexity is kept sufficiently low.

Many researches have been done in scheduling systems with offsets like Audsley et al. [8] who presented some work for a system with offsets using Generalised Chinese Remainder Theorem [43]; where they introduced the concept of common release. Goossens et al. [34, 33] who showed that neither RM nor DM is optimal for systems with offsets and presented two scheduling rules to choose the offsets, one of them is optimal but computationally unreasonable for large systems; while the other one is a

nearly optimal heuristic scheduling rule. Baruah et al. [14] and Goossens [32] who have shown that if an offset free system with arbitrary deadlines is not schedulable for all non-negative integer offset assignments, then this is also the case for all offset assignment with a granularity of m for all m (m is a non zero positive integer).

In 2006, Traore et al. [73] mentioned in their paper that the MF model is a particular case of tasks with offset (transactions), so they assumed that their offset analysis can be applied to the systems with MF tasks; where a MF task τ_i can be modeled to a transaction with period equals $n_i T_i$ (n_i and T_i are respectively the number of frames and the period of the MF task τ_i). In fact, analysis in this thesis would assume that the multiframe model is different from the transaction model as Traore's suggestion could only be applicable to a very strict MF models. For example, this offset analysis is applicable only to MF task that is AM (having its critical frame at the first position of its execution time sequence) and all frames in the same MF task have the same deadline; whilst offset analysis is not applicable to the general MF task and frame specific deadlines.

The incorrectness of the assumption that the MF model is a particular case of the offset model lies in the fact that offset model fails to correctly identify the worst-case combination of MF tasks. For example, a MF task with execution times $(1, 2)$, deadline 2 and period 10 would be considered equivalent, for scheduling analysis purposes, to two tasks τ_1^1 and τ_1^2 such that both have deadline 2 and period 20, τ_1^1 has an execution time equals 1, and τ_1^2 has an execution time equals 2 and also has an offset from τ_1^1 by 10 units (in the sense that the first invocation of τ_1^1 is released at 0 and successive invocations are released exactly 20 units apart, while the first invocation of τ_1^2 is released at 10 and successive invocations are released exactly 20 units apart). However, to see why such an approach for scheduling analysis is incorrect, consider a MF system consisting of two tasks; the one above, and the task τ_2 with execution time 1, deadline 2, and period 20. Using the same assumption, this second MF task would be transformed to a task with execution time 1, and its first invocation at 0 and the successive invocations exactly 20 units apart. The system would therefore be considered schedulable as τ_1^2 and τ_2 are not released simultaneously according to the offset assumption. However, in reality τ_2 is actually unschedulable because we assume in the system model that all frames of a MF have same priority and periods,

$task$	C	D	T
τ_1	$(3,2)$	$(5,5)$	10
τ_2	$(8,6,7,4)$	$(15,9,9,10)$	15

Table 2.8: Original Example System

so both τ_1^1 and τ_1^2 have the same priority and periods. So there is a situation where τ_1^2 and τ_2 are released simultaneously which results that τ_2 does not meet its deadline and so it is unschedulable.

However, even for the AM multiframe tasks whose deadlines are different from one frame to another within the same MF task, Traore's suggestion is not applicable. For example, assume a system in Table 2.8, according to Traore's suggestion, the system will be transformed to the system in Table 2.9.

$task$	$Offset$	C	D	T
τ_1	$(0,10)$	$(3,2)$	$(5,5)$	20
τ_2	$(0,15,30,45)$	$(8,6,7,4)$	$(15,9,9,10)$	60

Table 2.9: Transformed System Having Offsets

So the frames of τ_1 and τ_2 whose execution times are 3 and 7 respectively, do not share a simultaneous release in the transformed system (Figure 2.5) whilst in reality they do. Figure 2.5 represents the execution scenario of the system in Table 2.9. According to the offset analysis, τ_2 is considered as schedulable as all its deadlines are met. Whilst in reality it is not schedulable; as when τ_2 is released having an execution time of 7 simultaneously with τ_1 having the execution time of 3, τ_2 does not meet its deadline as its response will extends beyond 9.

Therefore, this thesis considers the MF model as a different model from the offset model.

2.3.6 Other Contributions Related to Response Time Analysis within Fixed Priority Scheduling

Eisenbrand et al. [29] has recently showed that the response time computation for RM preemptive scheduling is NP-hard. However, some research [38, 22, 28] has been

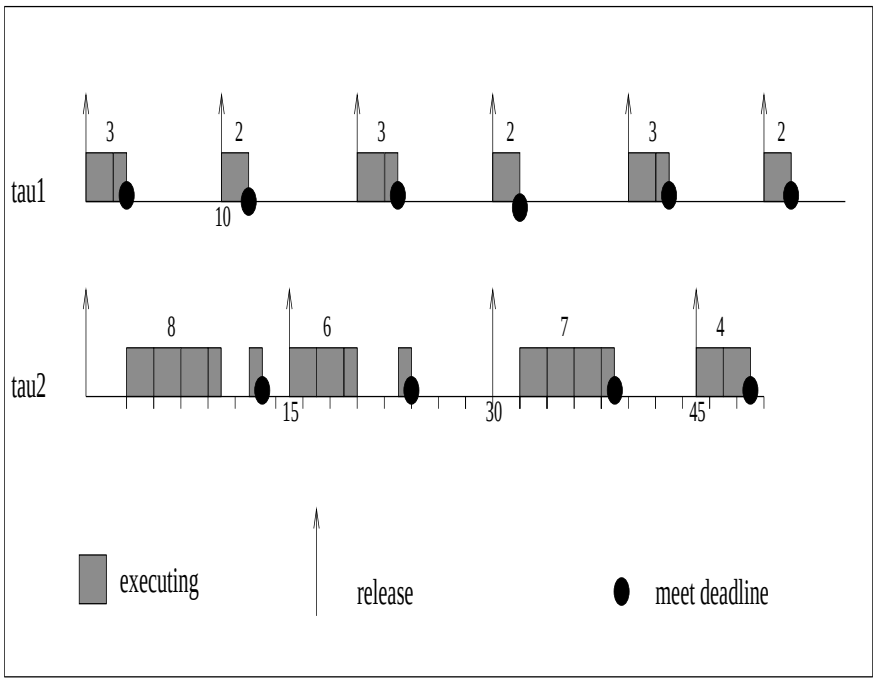


Figure 2.5: Execution Scenario of the Transformed System in Table 2.9

done to improve the efficiency of the exact response time test, providing an effective initial value of the fixed point solution of the response time equation.

Another sufficient response time test was developed by Fisher and Baruah [31, 30]; where they estimated the workload requested by higher priority tasks using an exact request bound function for a specific number of invocations and a linear function thereafter. In 2007, Richard et. al. [62] extended this work to include tasks that are subjected to release jitter.

Bini and Baruah [19] derived a closed form upper bounds on the response times and an associated linear-time sufficient test for independent preemptive tasks with arbitrary deadlines but no jitter. Davis et al.[27] had derived another flexible closed form upper bounds on the response times of tasks with arbitrary deadlines, release jitter and blocking.

2.4 Summary

As can been from all covered contributions, all contributions that are related to scheduling MF tasks are inexact. Moreover, non of the exact worst case response time contributions within fixed priority scheduling considered MF tasks. However, this thesis presents an exact scheduling test of MF tasks by analysing their worst case response times. The analysis depends on formulating the response time of a MF task assuming the MF tasks are released synchronously (i.e. they share a common release). The response time analysis in this thesis is hierarchically presented depending upon the generalisation of the MF model starting by the classic AM model and AM with blocking time, release jitter and arbitrary deadlines then ending up with non-AM model with blocking time, release jitter, arbitrary deadlines and frame specific deadlines.

3 Basic Exact Scheduling Analysis of AM Multiframe Tasks

This chapter[1] provides exact and tractable analysis based on the response time formulation for multiframe tasks when the AM restriction is applied. In general, to test the schedulability of a set of multiframe tasks, regardless of the AM restriction, requires examining all possible phases of the tasks [69]; which leads to an intractability problem for the scheduling analysis. But, having the AM restriction applied to a multiframe task, we show that only the *critical* frame can give rise to the worst-case response times for lower priority tasks. As a result the analysis is tractable. The following section provides the response time analysis of basic AM multiframe tasks[2].This basic analysis is given in two stages, firstly we give the basic formula of the worst case response time of an AM multiframe task. Secondly, we extend this formula to include blocking time. An evaluation of this analysis is given as a comparison between this exact scheduling analysis and the most recent published, but non-optimal, scheduling analysis.

This chapter is organised as follows: the following section gives the exact response time analysis of AM multiframe tasks; then the analysis is developed to include blocking in Section 3.2. Numeric examples are given in Section 3.3 to illustrate the two scheduling schemes: the worst case response time scheduling analysis of AM multiframe tasks and Lu's scheduling analysis [55]; which is the most recent published scheduling analysis for multiframe tasks. In this section (i.e. Section 3.3), we also show how the response time analysis determines the schedulability of the system

[1] Material based on this chapter was published in [77].

[2] A basic AM multiframe task means that the task does not have release jitter and does not include invocations from previous frames of the analysed MF task but is permitted to share resources, so it has blocking.

where Lu's analysis does not. Section 3.4 provides an analysis of randomly generated task sets to show how the response time test is better than any one previously published. A summary of the chapter is provided in Section 3.5.

3.1 Basic Response Time Analysis of AM Multiframe Tasks

This section covers the response time analysis of a basic multiframe task assuming that all multiframe tasks in the system satisfy the AM restriction (i.e. Equation (2.3)). The worst case response time of the AM multiframe task is the maximum response time of all frames of the MF task assuming their critical instance. Mok and Chen [56] identified the critical instance of an AM multiframe task as the simultaneous release of the critical frames[3] of both the analysed MF task and MF tasks whose priorities are higher than the analysed task (see Section 2.2.2 for details). As we assume that no frame interferes with any other frame in the same MF task, we will consider Mok and Chen's critical instance of the AM multiframe task to analyze its worst case response time.

For the AM multiframe task, τ_j, the cumulative function of its only critical frame is presented by Equation (3.1)

$$\xi_j^{m_j}(k) = \sum_{l=m_j}^{m_j+k-1} C_j^{l \bmod n}; k = 1, 2, .. \tag{3.1}$$

where m_j is the location of the critical frame of the AM multiframe task. For example, the value of $\xi_1^0(3)$ for the AM multiframe task τ_1 whose execution times are $(8,4,8,3)$ is 20. Using Equation (3.1) to present the amount of interference the higher priority AM multiframe tasks generate, the basic response time formula that is represented by Equation (2.9) is modified to be in the form used in the following theorem (i.e. Theorem 1).

[3] In the AM multiframe task, the critical frame is a peak frame.

Theorem 1 *Given a real-time system consisting of N independent AM multiframe tasks, the worst case response time of the multiframe task τ_i is given by the smallest non-negative solution to Equation (3.2):*

$$R_i = C_i^{m_i} + \sum_{j=1}^{i-1} \xi_j^{m_j}(\lceil \frac{R_i}{T_j} \rceil) \tag{3.2}$$

where $\xi_j^{m_j}(\lceil \frac{R_i}{T_j} \rceil)$ is the cumulative function of the critical frame of τ_j as defined by Equation (3.1).

Proof: As R_i is the worst case response time of the task τ_i, then for each multiframe task whose priority is higher than the priority of τ_i (i.e. $\tau_j : j = 1..i-1$); the number of invocations of τ_j within R_i is given by $\lceil \frac{R_i}{T_j} \rceil$ assuming the simultaneous release of the critical frames of τ_j and τ_i. So, when τ_j is released with its critical frame, the amount of interference that τ_j generates within R_i is given by: $\xi_j^{m_j}(\lceil \frac{R_i}{T_j} \rceil)$. In addition, as the critical instance of τ_i is the simultaneous release of the critical frames of all τ_j; for $j = 1,..i-1$, the maximum amount of interference that all τ_j generate within R_i is given by adding all interference that is generated by the higher priority AM multiframe tasks (i.e. $\sum_{j=1}^{i-1} \xi_j(\lceil \frac{R_i}{T_j} \rceil)$).

In addition, the maximum amount of time τ_i takes for execution is represented by $C_i^{m_i}$. So the response time of τ_i is given by Equation (3.2); which presents the execution of both the AM multiframe task τ_i itself as well as interference from all higher priority AM multiframe tasks.□

Equation (3.2) can be solved by a recurrence relation as in Equation (3.3).

$$R_i^{l+1} = C_i^{m_i} + \sum_{j=1}^{i-1} \xi_j^{m_j}(\lceil \frac{R_i^l}{T_j} \rceil) \tag{3.3}$$

where $R_i^0 = C_i^{m_i}$ and $l = 0, 1, 2, ..$ until $R_i^{l+1} = R_i^l$. However, if R_i^{l+1} becomes greater than the relative deadline, τ_i is not guaranteed to meet its deadline. In other words, if $R_i^{l+1} > D_i$ then τ_i is unschedulable.

Equation (3.2) calculates an exact worst case response time of an AM multiframe task assuming exact attributes of the system. On the other hand, a schedulable real

time system is the system whose all tasks can be scheduled on time. In other words, a schedulable real time system is the system whose all tasks meet their relative deadlines. Also, a task, in its turn, meets its deadline when its worst case response time is less than or equal to its relative deadline. So, the scheduling test, of a system with AM multiframe tasks, is presented as follows: a system with AM multiframe tasks is schedulable if and only if all its multiframe tasks meet their relative deadlines. Where the AM multiframe task meets its deadline if its worst case response time, that is calculated by Equation (3.2), is less than or equal to its relative deadline. The following example illustrates this test.

Example

Table 3.1 presents an example of two AM multiframe tasks, τ_1 and τ_2. To analyze

task	C	D	T
τ_1	$(4,3,1,8)$	9	10
τ_2	$(2,7,2)$	20	20

Table 3.1: Example System

the schedulability of the system, we first identify the location of the critical frames (i.e. m_i). As there is only one peak frame per MF task, the critical frame is the peak frame[4], so, $m_1 = 3$ and $m_2 = 1$.

Because τ_1 is the highest priority MF task in the system, its worst case response time is $R_1 = C_1^{m_1} = 8 < D_1$. To analyze the worst case response time of τ_2, we apply Equation (3.3) for $i = 2$ and $R_2^0 = C_2^{m_2} = 7$, so we get
$R_2^{l+1} = C_2^{m_2} + \sum_{j=1}^{2-1} \xi_j^{m_j}(\lceil \frac{R_2^l}{T_j} \rceil)$,
$l = 0$, $R_2^1 = 7 + \xi_1^{m_1}(\lceil \frac{R_2^0}{T_1} \rceil)$,
$R_2^1 = 7 + \xi_1^3(\lceil \frac{7}{10} \rceil)$,
$R_2^1 = 7 + 8 = 15$.
Similarly, we find $R_2^2 = 19$ for $l = 1$ and $R_2^3 = 19$ for $l = 2$. As $R_2^3 = R_2^2$, the worst case response time of τ_2 is $R_2 = 19 < D_2$. Therefore, τ_2 is schedulable.

As τ_1 and τ_2 are schedulable, the whole system is schedulable.

[4] If the MF task has more than one peak frame, then we apply Equation (2.3) for all peak frames and choose the frame that satisfies this equation as the critical frame.

3.2 Adding Blocking Time to the Response Time Analysis

As mentioned earlier in Chapter 2, blocking of a task is when this task is waiting for lower priority tasks to complete some execution. So, when we have a system of multiframe tasks, we expect more than one blocking value for the execution of τ_i from each lower priority MF task that shares the same resource with τ_i. That is because also all lower priority tasks are multiframe tasks and therefore could have different execution times. However, using priority ceiling protocols [66, 60] allows the task to be blocked at most once during its execution, so we only add, to the worst case response time formula, the maximum of the expected blocking values which we symbolise as B_i. Thus, assuming that τ_i has a maximum blocking of B_i, the worst case response time formula, is presented by Equation (3.4) as a collection of three kinds of execution: maximum execution of the task itself $C_i^{m_i}$, maximum blocking time B_i and maximum interference from the higher priority multiframe tasks, $\sum_{j=1}^{i-1} \xi_j(\lceil \frac{R_i}{T_j} \rceil)$.

$$R_i = C_i^{m_i} + B_i + \sum_{j=1}^{i-1} \xi_j^{m_j}(\lceil \frac{R_i}{T_j} \rceil) \tag{3.4}$$

Similar to above, Equation (3.4) is solved using a recurrence relation given by Equation (3.5); where $r_i^0 = C_i^{m_i}$ and $l = 0, 1, 2, \ldots$ until $R_i^{l+1} = R_i^l$. The worst case response time of τ_i is obtained when it is found that $R_i^{l+1} = R_i^l$ $(= R_i$ for the smallest value of $l)$. However, when R_i^{l+1} becomes greater than the deadline of the task, τ_i is not guaranteed to meet its deadline, so we say that the task is unschedulable.

$$R_i^{l+1} = C_i^{m_i} + B_i + \sum_{j=1}^{i-1} \xi_j^{m_j}(\lceil \frac{R_i^l}{T_j} \rceil) \tag{3.5}$$

This response time scheduling analysis is an efficient scheduling test, better than the utilisation test that is given by Lu et. al [55], from three points of view. Firstly, the response time test is a sufficient and necessary test when B_i is exact, which means that the response time test is an exact test. Secondly, it is applicable to the system model when the tasks have deadlines less than their relative periods. Thirdly, the

response time test does not depend on the priority assignment scheme of the tasks in the system. For example, the response time test is still applicable to the system model where priorities are assigned according to RM, DM or any other fixed priority assignment scheme; while the utilisation based test is not.

For more illustration of the efficiency of the response time test, we compare this analysis with the most recent published scheduling test (i.e. Lu's test [55]); in two steps. In the first step we give in the following section two numeric examples, the first one illustrates the worst case response time analysis that is presented in this section. The second example is a modified example of the first one; this example illustrates the analysis of Lu's test and at the same time shows the insufficiency of Lu's test. In the second step we give, in a following section, an evaluation of the comparison between the worst case response time analysis and Lu's analysis.

3.3 Numeric Examples

Table 3.2 represents an example task set of 5 AM multiframe tasks with their parameters and their worst case response times according to RM priority assignment (the smaller period the task has the higher priority it is assigned). To simplify the example, we assume that all deadlines are identical to their relative periods and all blocking terms are zero.

$task$	C	$T = D$	R
τ_1	(1)	3	1
τ_2	(2)	9	3
τ_3	$(3,1)$	18	8
τ_4	$(2,1)$	20	14
τ_5	$(6,3)$	60	32

Table 3.2: Example System1

Lu et al. [55] noted that the schedulability of this task set is unknown using Kuo's [44] method[5], while response time analysis shows that the task set is schedulable as

[5]Details of applying Lu's test is given in Section 2.2.3.

explained below, so the worst case response time test is better than Kuo's test [44]. Also, the analysis gives an exact value of the worst case response time of each AM multiframe task in the system. For example, to find the worst case response time of τ_4, we solve Equation (3.4) for $i = 4$ by applying Equation (3.5) so we get

$$R_4^{l+1} = C_4^{m_4} + \sum_{j=1}^{2} \xi_j^{m_j}(\lceil \frac{R_4^l}{T_j} \rceil);$$

where $C_4^{m_4} = 2, R_4^0 = 2$.
$l = 0$, $R_4^1 = 2 + \xi_1^{m_1}(\lceil \frac{R_4^0}{T_1} \rceil) + \xi_2^{m_2}(\lceil \frac{R_4^0}{T_2} \rceil) + \xi_3^{m_3}(\lceil \frac{R_4^0}{T_3} \rceil)$
$R_4^1 = 2 + \xi_1^{m_1}(\lceil \frac{2}{3} \rceil) + \xi_2^{m_2}(\lceil \frac{2}{9} \rceil) + \xi_3^{m_3}(\lceil \frac{2}{18} \rceil) = 2 + 1 + 2 + 3 = 8.$
Similarly, we find R_4^{l+1}, for $l = 1, 2, 3, 4, 5$, so we get $R_4^2 = 10, R_4^3 = 13, R_4^4 = 14, R_4^5 = 14$ respectively. As $R_4^4 = R_4^5$, we stop increasing l and the worst case response time of τ_4 is 14 which is less than the deadline of τ_4, so τ_4 is schedulable.

Similarly, we find all worst case response times of all AM multiframe tasks τ_1, τ_2, τ_3, τ_4, τ_5 as given in Table 3.2 (i.e. the R column). As all of the worst case response times are less than their relative deadlines, all multiframe tasks in the system are schedulable. So, the system is schedulable.

However, if we modify the execution times of the task τ_4 to be $(3,2)$ instead of $(2,1)$ and keep all other parameters as in Table 3.2 (see Table 3.3); we find that the schedulability of the system is unknown using Lu's method but it is schedulable using our response time analysis. The following is the explanation.

task	C	T
τ_1	(1)	3
τ_2	(2)	9
τ_3	$(3,1)$	18
τ_4	$(3,2)$	20
τ_5	$(6,3)$	60

Table 3.3: Example System2

task	$\hat{C}$	$\hat{T}$
$\hat{\tau}_1$	$(7,5)$	18
$\hat{\tau}_2$	$(34,30)$	60

Table 3.4: Merged System

Using Lu's approach[6], τ_1, τ_4, and τ_5 are merged using Kuo's method [44] to $\hat{\tau}_2$ with

[6]Further details can be found in Section 2.2.3.

a period equals to the maximum period of T_1, T_4 and T_5; which is 60 in this example. $\hat{\tau}_2$ has number of execution times equals to the least common multiple of n_1, n_4 and n_5; which is 2 in this example. Values of $\hat{\tau}_2$'s execution times are found by applying

$$\hat{C_2^j} = (\sum_{k=0}^{(\frac{60}{T_1})-1} C_1^{(j(\frac{60}{T_1})+k) \ mod \ n_1}) + (\sum_{k=0}^{(\frac{60}{T_4})-1} C_4^{(j(\frac{60}{T_4})+k) \ mod \ n_4}) + (\sum_{k=0}^{(\frac{60}{T_5})-1} C_5^{(j(\frac{60}{T_5})+k) \ mod \ n_5})$$

$$for \ j = 0, 1.$$

So, $\hat{C_2^0} = 34$ and $\hat{C_2^1} = 30$.

Also, τ_2 and τ_3 are merged, using Kuo's method, to $\hat{\tau}_1$ with the number of execution time equal to $\hat{n_1} = 2$ and execution time values $\hat{C_1} = (7,5)$ and a period of 18. Table 3.4 represents the attributes of the merged tasks.

Once the merged tasks are identified, the scheduling test is to check if the total peak utilisation, U, is less than or equal to a pre defined conditional bound, CB. U, is the summation of all peak utilisations of the multiframe tasks in the system; and it is given by

$$U = \sum_{i=1}^{2} \max_{0 \le j \le \hat{n_i}-1} \{\frac{\hat{C_i^j}}{T_i}\} = \frac{7}{18} + \frac{34}{60} = 0.95556.$$

CB is found depending on two parameters r and z.
r is given as

$$r = \min_{1 \le i \le \hat{N}} \{r_i\};$$

where r_i is the ratio of the first two execution times of $\hat{\tau}_i$ and is defined by

$$r_i = \frac{\hat{C_i^0}}{\hat{C_i^1}}, so \ r_1 = \frac{7}{5}, r_2 = \frac{34}{30}.$$

Therefore, $r = \min\{\frac{7}{5}, \frac{34}{30}\} = \frac{34}{30} = 1.133333$.
z is given by

$$z = \max\{\min_{1 \le i \le \hat{N}-1} \{\frac{V_i}{T_{\hat{N}}}\}, \frac{r}{1+r}\};$$

where V_i is called a virtual period and is given by

$$V_i = \lfloor \frac{\hat{T}_{\hat{N}}}{\hat{T}_i} \rfloor \hat{T}_i = \lfloor \frac{60}{\hat{T}_i} \rfloor \hat{T}_i.$$

So, $V_1 = \lfloor \frac{60}{18} \rfloor 18 = 54$.

Therefore, $z = \max\{\frac{54}{60}, \frac{1.13333}{1+1.13333}\} = \max\{0.9, 0.53125\} = 0.9$.

Once r and z are identified, $CB(r,z)$ is given by

$$\begin{array}{rcccccc} CB(r,z) & = & z & + & r(z-1) & + & r(\hat{N}-1)((\frac{1}{z})^{\frac{1}{\hat{N}-1}}-1) \\ & = & 0.9 & + & 1.13333(0.9-1) & + & 1.13333(2-1)((\frac{1}{0.9})^{\frac{1}{2-1}}-1) \\ & = & 0.91259. & & & & \end{array}$$

Therefore, the conditional bound function (CB) of the merged tasks is less than the total peak utilisation of the system (i.e. $CB < U$) which means using Lu's test that the schedulability of the original system that is given in Table 3.3 is unknown. However, the exact response time analysis that is given in this chapter shows that the system is schedulable because:

$R_1 = 1 < 3$,
$R_2 = 3 < 9$,
$R_3 = 8 < 18$,
$R_4 = 15 < 20$,
$R_5 = 35 < 60$.

The example in Table 3.3 illustrates how the worst case response time analysis is better than Lu's analysis, in the sense, that the schedulbility status of the example system is not known using Lu's test but is found using worst case response time analysis. In the following section, we investigate the performance of both worst case response time analysis and Lu's analysis and then we make a comparison between both of them over randomly generated AM multiframe tasks.

3.4 Evaluating Exact Response Time Scheduling Analysis for MF Tasks

We show in this section how the worst case response time test is a clear improvement, compared to the most recent scheduling test that is represented by Lu et. al [55]. Comparison in this section requires the generation of real-time systems to check their schedulability status under each approach (i.e. each of the response time and Lu's approaches) and then evaluate the performance of each of these two approaches to determine to what extent the worst case response time test is better than Lu's test. This evaluation is presented as experiments that are explained in three steps, the first step shows how each experiment is constructed, the second step illustrates how each experiment is run, and the third step shows the results of the experiments.

3.4.1 Experimental Setup

The generation of the real-time system means the generation of the size of the system as well as the generation of the multiframe tasks that form the system. From the system size point of view, we assign the number of tasks in the system for each experiment to be one of the values $\{5, 20, 100\}$. While from the multiframe task's generation point of view, we require the generation of four parameters for each multiframe task, τ_i, (i.e. n_i, T_i, D_i, C_i; which are respectively: number of frames, Period, Deadline, and the execution time sequence).

The four parameters of a multiframe task are generated, in summary, as follows. The first parameter that is the number of frames of the multiframe task is assumed as fixed for all multiframe tasks in the system and is chosen, for each experiment, as one of the values $\{3, 7, 13, 23\}$. The values are chosen to be prime numbers so the execution time sequence is guaranteed to be in its shortest form. The second and third parameters, which are the period and deadline of the multiframe task, are assumed to be identical to each other for each multiframe task and are randomly generated in the range of $[1, 2500]$ using the uniform distribution. Once the deadlines are assigned to each task, the priorities of the tasks are also assigned according to DM (which is equal to RM in our experiments) priority assignment.

The sequence of the execution times, which is the fourth parameter, is generated in two steps. In the first step we generate the utilisation for each frame of the multiframe task, while in the second step we assign the execution time of this frame by multiplying its utilisation by its period. The following is the full details of the generation scheme for the execution times.

First of all, we give an overall utilisation of the system and then we distribute this utilisation to all multiframe tasks in the system. Bini et al. [20] introduced an efficient algorithm called UUniFast algorithm; which is used to randomly distribute the overall utilisation of the system to all tasks in the system. The algorithm is summarised by the pseudocode that is given by Algorithm 1; where *Average_Uti* represents the vector of the average utilisation portions for the MF tasks in the system.

Algorithm 1 Uunifast Pseudocode

Inputs: Overall_Utilisation, Tasks_Number.
Outputs: Array Average_Uti.

$Sum\_Uti \Leftarrow Overall\_Utilisation$
$N \Leftarrow Tasks\_Number$
for i =1 to N-1 **do**
  $nextSumU \Leftarrow Sum\_Uti.rand^{\frac{1}{N-i}}$
  $Average\_Uti(i) \Leftarrow Sum\_Uti - nextSumU$
  $Sum\_Uti \Leftarrow nextSumU$
end for
$Average\_Uti(N) \Leftarrow Remaining\_Uti$

We consider each portion of the utilisation for each multiframe task as the mean utilisation of this multiframe task, and we multiply this mean by the number of frames, then we again apply the UUnifast algorithm to the results of the multiplication. In this case, we get the utilisation of each frame in the multiframe task and therefore the execution time of this frame is the multiplication of its utilisation by its period. Algorithm 2 represents the descriptions of the way that is used in generating the execution times of each MF task. Once we get the execution time sequence we re-arrange it to be AM using Mok's algorithm [57].

For each experiment, we modify one and fix two of the three attributes of the analysed system: utilisation, number of frames and number of tasks. All experiments

Algorithm 2 Generating Execution Time Vectors

Inputs: Overall_Utilisation, Tasks_Number, Frames_Number, Array Period.
Outputs: Matrix of Execution_Time.

Array Averag_Utilisation $\Leftarrow Uunifast(Overall\_Utilisation, Tasks\_Number)$
for i = 1 to Tasks_Number **do**
  $Sum\_Uti\_MF \Leftarrow Averag\_Utilisation(i)$. $Frames\_Number$
  Array Frame_Utilisation $\Leftarrow Uunifast(Sum\_Uti\_MF, Frames\_Number)$
  for j = 0 to Frames_Number-1 **do**
    $Execution\_Time(i, j) \Leftarrow Frame\_Utilisation(j)$. $Period(i)$
  end for
end for

show, as expected, that the number of schedulable systems when the exact response time test is applied is always greater than when Lu's test is applied.

3.4.2 Scope of Running the Experiments

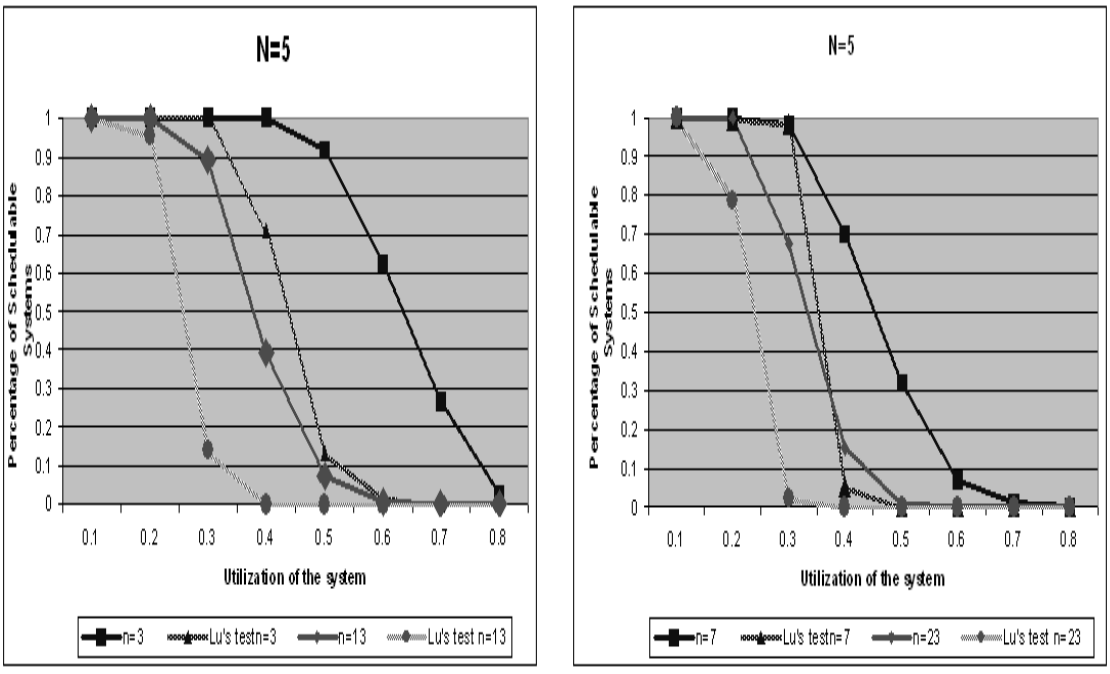
We run each experiment 1000 times, for each chosen number of frames, in four steps as following. Firstly, we generate the parameters of the experiment (i.e. number of frames, periods, deadlines, and execution time sequences) as previously explained. Secondly, we check the worst case response time of each task, using Equation (3.2), whether it is less than the relative deadline. In other words, we check the schedulability of the system by checking if the worst case response times of all multiframe tasks in this system are within their relative deadlines. Thirdly, for the same parameters of the system we check the schedulability of the same generated system using Lu's test. Lastly, for each of the two tests, we count the percentage of the number of schedulable systems out of the 1000 ones that are randomly generated.

3.4.3 Results of the Experiments

From the utilisation point of view, we investigate the values of the utilisations that are in $(0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8)$. Figures 3.1, 3.2, 3.3 show the percentage of the schedulable systems versus the overall utilisation of the systems regarding two parameters: number of tasks, N, and number of frames, n. Each line in each graph in

Figures 3.1, 3.2, 3.3 shows the results of the schedulability percentages for a value of n and a value of N. To simplify the presentation of the results, we present only two values of n in each graph. So, each graph has four lines, each two lines have the same values of parameters and present the results of both the response time test and Lu's test. For example, graph (a1) in Figure 3.1 shows the results for 5 number of tasks and two values of n, that are 3 and 13; and likewise all graphs of Figures 3.1, 3.2, 3.3 show the results for different values of the number of tasks and number of frames.

Figures 3.1, 3.2, 3.3 show that when the overall utilisation of the system is very low, 0.1, both of the response time and Lu's tests give the same performance of 100% schedulable systems. While when the utilisation is very high, greater than 0.6, although the exact test is better than Lu's one, the success of both tests is very low (as these systems are indeed unschedulable). So, we emphasise the range $[0.2, 0.6]$ of the overall utilisation to show how much the exact response time test is better than Lu's test.

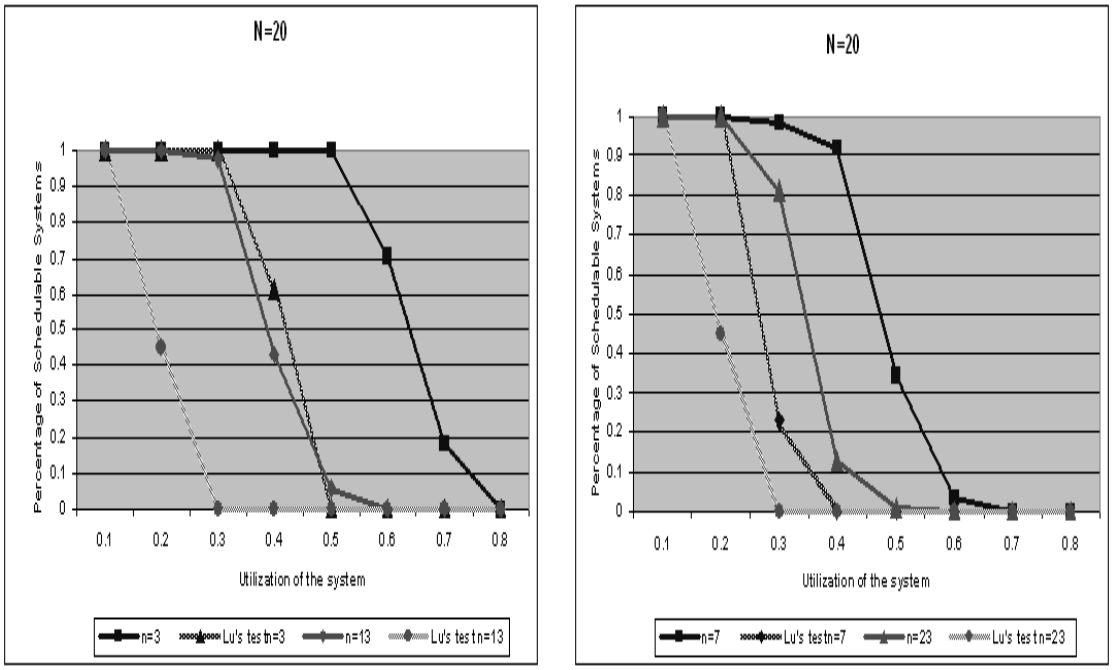


(a1) (a2)

Figure 3.1: Percentage of Schedulable Systems Regarding the Overall Utilisation of the System after Applying Response Time and Lu's Tests (N=5)

Graph (a1) in Figure 3.1 shows that there is less than 10% better performance of the exact test than Lu's test; when the overall utilisation of the system is 0.2, for 5 tasks in the system, and number of frames equal to 13. While this standard of performance rises to 20% in graph (a2) (i.e. percentage of the number of schedulable systems is 100%, according to the exact test, while this percentage is 80%, according to Lu's test), when the number of frames is 23 for the same other parameters.

The performance of the response time test becomes even better by increasing the number of tasks and number of frames. For example, graphs (b1) and (b2) in Figures 3.2 show that there is 55% better performance of the exact test than Lu's test; when the overall utilisation of the system is 0.2, for 20 tasks in the system, and number of frames is 13 or 23. While this standard of performance rises to 95% in graph (c2),

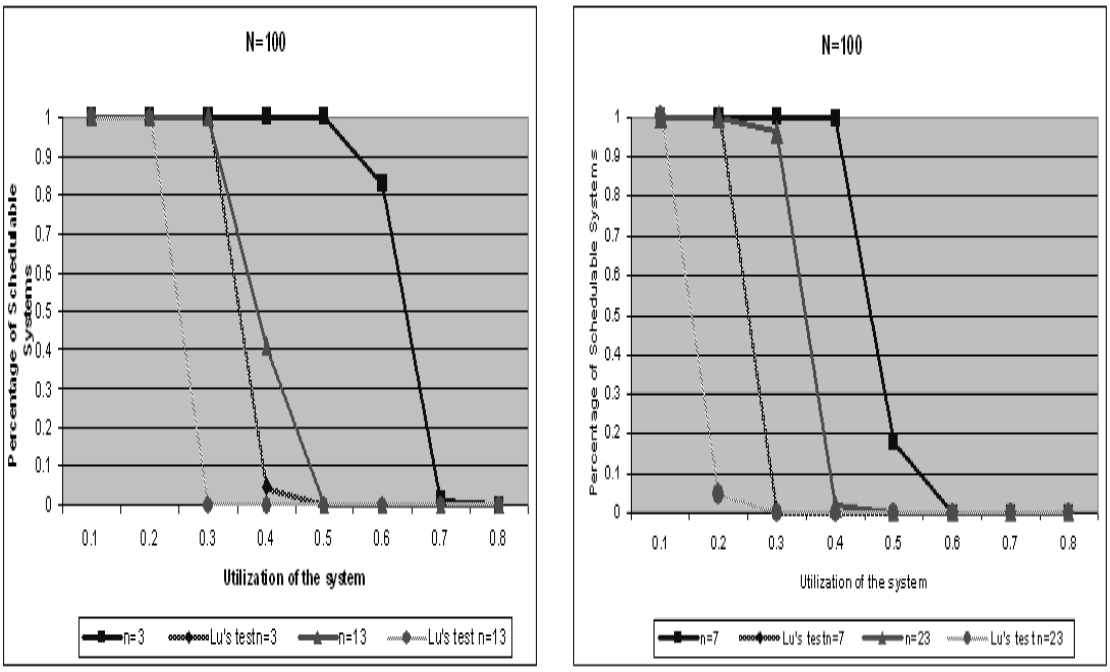


(b1) (b2)

Figure 3.2: Percentage of Schedulable Systems Regarding the Overall Utilisation of the System after Applying Response Time and Lu's Tests (N=20)

Figure 3.3 (i.e. percentage of the number of schedulable systems is 100%, according to the exact test, while this percentage is 5%, according to Lu's test); when the number

of tasks is 100 and the number of frames becomes 23 for the utilisation 0.2.



(c1) (c2)

Figure 3.3: Percentage of Schedulable Systems Regarding the Overall Utilisation of the System after Applying Response Time and Lu's Tests (N=100)

All graphs apart from (a2), in Figure 3.2, show that when the overall utilisation of the system increases up to 0.4 (and sometimes 0.5 as in graphs (b1) and (c1)) and the number of frames is 3, *or* 7; the performance of the exact test stays higher than 90% for all studied number of tasks (i.e. 5, 20, *and* 100) while at the same time, graph (b2) shows that the performance of Lu's test decreases to about 22% when the utilisation is 0.3, number of tasks is 20 and number of frames is 7. Also, from graph (c2), there is around 97% better performance of of the exact than Lu's test; when the overall utilisation of the system is 0.3, for 100 tasks in the system, and number of frames is 23.

In addition, graph (b1) shows that there is about 42% better performance of the exact test when the overall utilisation of the system is 0.4, the number of frames is 13 and the number of tasks is 20. While graph (b2) shows that there is 80% better performance

of the exact when the overall utilisation of the system is 0.3, the number of frames is 23 and the number of tasks is 20; where 80% of the number of the random tasks are schedulable by the response time test but none of them were schedulable using Lu's test.

So, the percentage of the schedulability performance of the exact response time test is much better than Lu's test and some times reach around 100% better performance. For example, graph (c1) shows that 100% of the random systems are schedulable using exact test while non of the systems are schedulable using Lu's test; when the overall utilisation is 0.3, the number of tasks is 100 and number of frames is 13. Similarly, graph (c2) shows that when the overall utilisation is 0.3, the number of tasks is 100 and number of frames is 23; the percentage of the schedulable systems using exact test is about 97% while 0% of the systems are schedulable using Lu's test.

3.5 Summary

In this chapter, we present an exact scheduling test for a system of AM multiframe tasks in terms of worst case response time analysis. The test shows a clear improvement in the scheduling performance from three points of view, firstly, the test is exact and tractable. Secondly, the test is applicable to the system model when deadlines of the tasks are less than their relative periods and regardless of the scheme for priority assignment. For example, response time test is still applicable to the system model where priorities are assigned according to RM, DM or even any other priority assignment scheme

Thirdly, evaluations show that this exact response time test has better performance than the most effective utilisation-base scheduling test for AM multiframe tasks. This improvement could reach 100% for some system parameters.

4 Extensions of the Exact Scheduling Analysis of AM Multiframe Tasks

This chapter[1] extends the basic system model that was given in the previous chapter (i.e. Chapter 3) and presents the worst case response time analysis that copes with the extended model. The extension of the basic model is achieved in two directions relating to release jitter and arbitrary deadlines. In the first direction we assume that each AM multiframe task, τ_j, has a maximum release jitter, J_j, but its deadline is less than its relative minimum release times. In the second direction we assume that each AM multiframe task, τ_i, has a deadline could be greater than its relative period, so an AM multiframe task, τ_i, could have interference from its previous frames during its execution, but no release jitter is permitted in this stage of extension. However, a combination of having release jitter and arbitrary deadlines is also given later on.

This chapter is organised as follows: the next section provides an exact worst case response time analysis of AM multiframe tasks assuming that these tasks are subjected to release jitter but no interference from the analysed task itself is permitted. Section 4.2 gives an exact worst case response time analysis of AM multiframe tasks assuming no task in the system has release jitter and, also, the analysed task could have arbitrary deadline so there could be interference from its previous frames during its execution. Section 4.3 analyses the worst case response time of AM multiframe tasks when these tasks have release jitter and arbitrary deadlines at the same time. A summary of the chapter is given in Section 4.5.

[1] Material based on Sections 4.1 and 4.2 have been published in [78].

4.1 Analysis of AM Multiframe Tasks with Release Jitter

When a task τ_j is subjected to release jitter, J_j, this task is not released as soon as it arrives in the system; where the maximum time from when it arrives in the system and being released is J_j. So, release jitter of a task τ_j could increase the number of interference that τ_j provides within the execution of a lower priority task, in the sense that τ_j could be released within less than its minimum inter arrival time, T_j. So, T_j is not purely constant for all jobs of τ_j; which means that the number of interference that τ_j provides within R_i (i.e. the worst case response time of a lower priority task τ_i) can not be purely presented as $\lceil \frac{R_i}{T_j} \rceil$. Therefore, the basic worst case response time formula of AM multiframe tasks (i.e. Equation (3.4)) requires a relative modification to cope with the release jitter model. This section presents full details of this modification assuming no interference from the analysed MF task.

To formulate the release jitter situation mathematically, assume $s_j^{m_j+k}$ is the time when the frame that follows τ_j's critical frame by k steps is released $(k = 0,1,2,..)$ (this implies that $s_j^{m_j}$ is the time when τ_j's critical frame is released). As the critical frame of an AM multiframe task, τ_j, always generates the maximum amount of interference within the execution of a lower priority task, τ_i, for all number of τ_j's invocations, we assume that τ_j's critical frame is released first in the execution of the AM multiframe task τ_j. So, when τ_j is subjected to release jitter, $s_j^{m_j}$ takes its place within a time interval of length $J_j < T_j$ whilst $s_j^{m_j+k}$ take their places after k periods. Equation (4.1) represents mathematically release jitter situation of τ_j.

$$kT_j + x \leq s_j^{m_j+k} \leq kT_j + y; \ \forall k \in \mathrm{Z}(i.e.\ k = 0,1,2,..) \tag{4.1}$$

$$where\ J_j = y - x.$$

In fact, τ_j indeed preempts τ_i the most when $s_j^{m_j}$ takes place rightmost in its release jitter interval (i.e. J_j) whilst $s_j^{m_j+k}$ take place leftmost in their release jitter interval, $(\forall k = 1,2,..)$. From Equation (4.1) $s_j^{m_j} = y$ and $s_j^{m_j+k} = x + kT_j;\ (k = 1,2,...)$. In addition, the maximum execution of the lower priority MF task τ_i is presented by the

peak frame of τ_i[2]. Therefore, the worst case preemption scenario of τ_i is when τ_i's peak frame is released simultaneously with the critical frames of all higher priority AM multiframe tasks. Assume s is the time when τ_i's peak frame is released, Figure 4.1 illustrates the worst case execution scenario of τ_i having only two AM multiframe tasks, a low priority one τ_i and a high priority one τ_j.

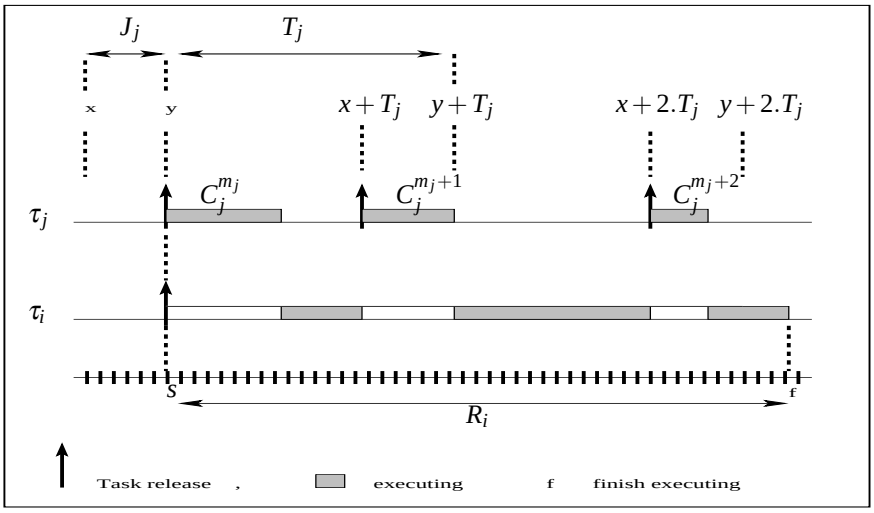


Figure 4.1: Illustration of Release Jitter Problem

As τ_i does not have interference from its previous frames and τ_i's peak frame provides the maximum amount of execution of τ_i, analysing the peak frame of τ_i is enough to determine the schedulability status of τ_i. We call the situation that leads τ_i to execute for the longest time, *the critical instance of* τ_i. The following definition illustrates this critical instance of τ_i, for the system model in this section.

Definition 4 *The* **critical instance of an AM multiframe task, τ_i, in a system subjected to release jitter** *is the simultaneous release of τ_i's peak frame and the critical frames[3] of the higher priority AM multiframe tasks; taking into account that the critical frames are released at the very end of their relative release jitter interval (after their relative arrival times) whilst next frames are released at the very beginning of their relative release jitter interval (so, they are released as soon as they arrive).*

[2]Or the critical frame of τ_i as the critical frame of AM multiframe task is a peak frame.
[3]Remember that an AM multiframe task has only one critical frame.

So, the worst case response time of an AM multiframe task τ_i is found by finding the worst case response time of τ_i's peak frame, assuming the critical instance of τ_i that is given by Definition 4. Finding this worst case response time needs the worst case interference from all higher priority AM multiframe tasks. The following lemma proves the worst interference from a higher priority AM multiframe task τ_j.

Lemma 1 *For a real-time system whose tasks are AM multiframe tasks τ_j; $j = 1,2,..,N$. Each multiframe task τ_j has a maximum release jitter equals J_j, and its critical frame is at position m_j. Assuming Definition 4, $\tilde{I}_j$ is given by Equation (4.2); where $\tilde{I}_j$ stands for the maximum interference from a higher priority multiframe task τ_j in R_i; where R_i is a period of execution of τ_i at its critical instant.*

$$\tilde{I}_j = \xi_j^{m_j}(\lceil \frac{R_i + J_j}{T_j} \rceil). \tag{4.2}$$

Proof

We divide $\tilde{I}_j$ into two parts $\tilde{I}_j = C_j^{m_j} + \tilde{I}_j^{rest}$; where $C_j^{m_j}$ is the first interference that τ_j provides within $(T_j - J_j)$ while $\tilde{I}_j^{rest}$ is the amount of interference that τ_j provides within $R_i - (T_j - J_j)$ starting from the release that follows the critical one. So, $\tilde{I}_j^{rest}$ is given by: $\tilde{I}_j^{rest} = \xi_j^{m_j+1}(\lceil \frac{R_i-(T_j-J_j)}{T_j} \rceil)$.

Therefore,

$\tilde{I}_j = C_j^{m_j} + \xi_j^{m_j+1}(\lceil \frac{R_i-(T_j-J_j)}{T_j} \rceil)$

$\tilde{I}_j = \xi_j^{m_j}(\lceil \frac{R_i-(T_j-J_j)}{T_j} \rceil + 1)$ because the cumulative function, $\xi_j^{m_j}(\lceil \frac{R_i-(T_j-J_j)}{T_j} \rceil + 1)$, starts from the release that is immediately previous to $(m_j + 1)$ so an extra interference has been added to ξ_j whilst the relative release is subtracted by one to be m_j instead of $m_j + 1$.

$\tilde{I}_j = \xi_j^{m_j}(\lceil \frac{R_i-(T_j-J_j)}{T_j} + 1 \rceil)$ because we add an integer to the ceiling function so we can move this integer into the ceiling function

$\tilde{I}_j = \xi_j^{m_j}(\lceil \frac{R_i-(T_j-J_j)}{T_j} + \frac{T_j}{T_j} \rceil) \quad = \xi_j^{m_j}(\lceil \frac{R_i+J_j}{T_j} \rceil).\square$

Using Lemma 1, the following theorem proves the worst case response time formula of an AM multiframe task assuming release jitter scenario.

Theorem 2 *Given a real-time system consisting of N multiframe tasks τ_j; $j = 1,2,..,N$ that satisfy the AM restriction, each multiframe task τ_j has a maximum release jitter*

equals J_j, and its critical frame is at position m_j; the worst case response time of the multiframe task τ_i is given by the smallest non-negative solution to Equation (4.3) assuming the priority ceiling protocols [66, 60]:

$$R_i = C_i^{m_i} + B_i + \sum_{j=1}^{i-1} \xi_j^{m_j}(\lceil \frac{R_i + J_j}{T_j} \rceil) \tag{4.3}$$

where $\xi_j^{m_j}(\lceil \frac{R_i+J_j}{T_j} \rceil)$ is the cumulative function of the critical frame of τ_j as defined by Equation (2.1) and B_i is the maximum expected blocking time of τ_i.

Proof

Assume $\tilde{I}$ is the maximum interference from tasks whose priorities are higher than the multiframe task τ_i. Definition 4 introduces τ_i's critical instance as the simultaneous release of all higher priority tasks, so $\tilde{I}$ can be presented by a summation of all $\tilde{I}_j$; where $\tilde{I}_j$ is the maximum interference from τ_j:

$$\tilde{I} = \sum_{j=1}^{i-1} \tilde{I}_j.$$

Assuming Lemma 1, the maximum amount of interference from all AM multiframe tasks that have higher priority than τ_i is given by

$$\tilde{I} = \sum_{j=1}^{i-1} \xi_j^{m_j}(\lceil \frac{R_i + J_j}{T_j} \rceil).$$

On the other hand, using priority ceiling protocols [66, 60] allows the task to be blocked at most once during its execution, so we only add, to the worst case response time formula, the maximum of the expected blocking values which we symbolise as B_i.

Thus, the worst case response time formula of τ_i, is presented as a collection of three kinds of execution: maximum execution of the task itself $C_i^{m_i}$, maximum blocking time B_i and maximum interference from the higher priority multiframe tasks; which is identical to Equation (4.3).□

Solving Equation (4.3) is given by a recurrence relation as in Equation (4.4).

$$R_i^{l+1} = C_i^{m_i} + B_i + \sum_{j=1}^{i-1} \xi_j^{m_j}(\lceil \frac{R_i^l + J_j}{T_j} \rceil); \tag{4.4}$$

where $R_i^0 = C_i^{m_i}$ and $l = 0, 1, ..$ until we get $R_i^{l+1} = R_i^l = R_i$. However, if $R_i^{l+1} + J_i$ becomes greater than the deadline, we say that the system is unschedulable. This is because the deadline of the task is relative to its arrival time[4] whilst the response time of the task is relative to its release time. Hence, the scheduling test for a task τ_i with release jitter J_i is: τ_i is schedulable if $R_i + J_i \leq D_i$; where R_i is found by applying Equation (4.4).

Example

As an illustration of the presented analysis in this section, Table 4.1 represents a simple system example of two tasks: τ_1 and τ_2. Priorities of the tasks are assigned according to $(D-J)-$*monotonic* priority assignment that is presented by Theorem 16 in Section 2.3.3. To simplify the example, we assume all blocking times are zero. To find the worst case response time of τ_2 we apply Equation (4.4) to get:

Task	C	D	T	J	Priority
τ_1	$(5,4,3)$	10	12	2	1
τ_2	$(6,4)$	20	20	0	2

Table 4.1: Example System Attributes

$R_2^{l+1} = C_2^0 + \xi_1^0(\lceil \frac{R_2^l + J_1}{T_1} \rceil)$; $R_2^0 = C_2^0 = 6$.
$l = 0,\ R_2^1 = 6 + \xi_1^0(\lceil \frac{6+2}{12} \rceil) = 6 + 5 = 11$,
$l = 1,\ R_2^2 = 6 + \xi_1^0(\lceil \frac{11+2}{12} \rceil) = 6 + 9 = 15$,
$l = 2,\ R_2^3 = 6 + \xi_1^0(\lceil \frac{15+2}{12} \rceil) = 6 + 9 = 15 = R_2^2$. So, $R_2 = 15$.
$R_2 + J_2 < D_2$; which is 20 in this example. Therefore, τ_2 is schedulable, also τ_1 is schedulable because $R_1 = 5$. $R_1 + J_1 < D_1$ as $5 + 2 < 10$.
Hence the whole example system is schedulable.

[4]The arrival times of the AM multiframe task τ_j in Figure 4.1 is presented by the term $x + kT_j$.

4.2 Analysis of AM Multiframe Tasks with Arbitrary Deadlines

This section extends the basic response time analysis that is given in Chapter 3 to be applicable to the AM multiframe task whose deadline is arbitrary and could be greater than its relative period. So there could be a situation where an AM multiframe task could suffer from interference from its previous frames during its execution. Analysis in this section does not permit any release jitter for any AM multiframe tasks.

To start with, we modify Definition 4 of the critical instance of an AM multiframe task to cope with the arbitrary deadlines model. As the AM multiframe task may suffer from interference from its previous frames and the critical frame of the AM multiframe task always provides the maximum amount of interference, for any possible number of its invocations (i.e. interference); we define the critical instance of the AM multiframe task as the simultaneous release of the critical frames of the analysed task and all higher priority AM multiframe tasks as in Definition 5.

Definition 5 *The* **critical instance of an AM multiframe task τ_i with arbitrary deadlines** *is the simultaneous release of the critical frame of τ_i with the critical frames of the higher priority multiframe tasks, taking into account that all τ_i's frames are released as soon as they arrive.*

Assuming this critical instance, the first step of the worst case response time analysis of τ_i is to introduce the term *busy period* of a frame of a MF task τ_i as the time from when this frame is released until it finishes its execution. So, the worst case response time of τ_i is the maximum of all busy periods of τ_i. We symbolise the busy period of the q^{th} frame[5] of the MF task τ_i as $w_i(q)$; $q = 1, ...$

The restriction of having deadlines less than their relative periods leads all busy periods of a schedulable MF task not to extend beyond its period. However, having arbitrary deadlines could lead the busy periods of a task to extend beyond its period and therefore its response time would include extra interference from the analysed task itself. So, the analysis in this scenario is concerned with analysing the interference

[5]Although q's values are $1, 2, ..$, we say q^{th} to simplify the presentation.

from the analysed AM multiframe task itself as well as interference from other tasks in the system.

To identify the amount of interference from the analysed task itself that should be considered in its response time analysis, we have to identify the relative number of invocations (i.e. interference) this task experiences within its busy period. To illustrate the problem of arbitrary deadlines more, Table 4.2 gives a simple numerical example system consisting of two tasks: a high priority task τ_1 and a low priority task τ_2. For simplicity and clarity we assume that none of the MF tasks has blocking; and τ_1 has one frame whilst only τ_2 is AM multiframe task with 4 frames.

Task	C	D	T
τ_1	5	10	10
τ_2	$(10,6,8,4)$	25	15

Table 4.2: Example of Arbitrary Deadline

Figure 4.2 shows four invocations of τ_2 starting from the execution of its critical

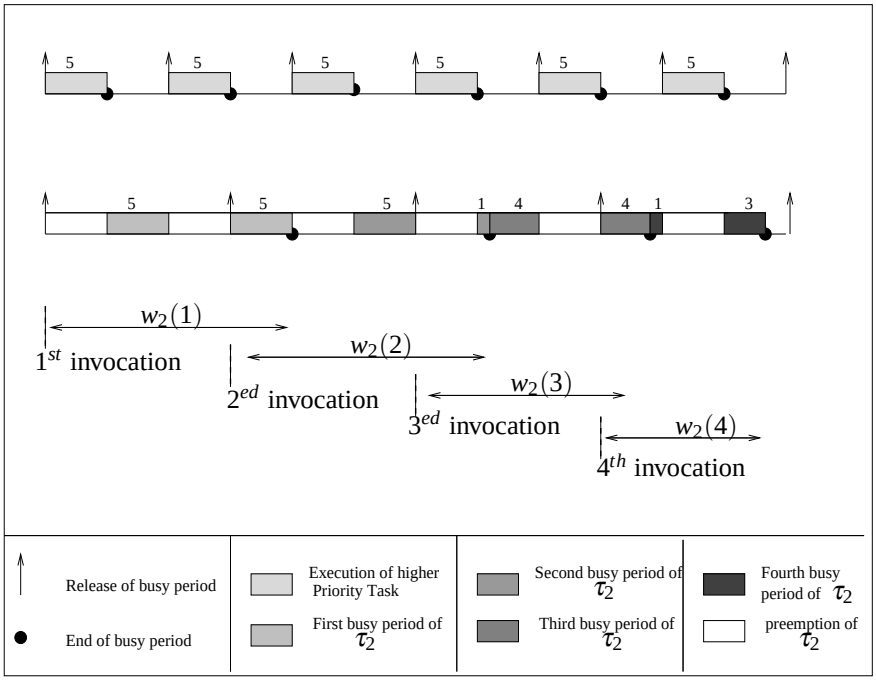


Figure 4.2: Illustration of Arbitrary Deadline Scenario- Timeline Diagram

frame and, also, shows how last three invocations (i.e. 2^{nd} invocation, 3^{rd} invocation, and 4^{th} invocation) of τ_2 include interference from previous frames of τ_i itself. In other words, Figure 4.2 shows four busy periods of τ_2 (i.e. $w_2(q); q = 1, .., 4$). As the worst case response time of a task is the maximum busy period that this task can experience, the worst case response time of τ_2 is $w_2(2)$; which equals 21 in this example (full details of the analysis and calculations are provided in the example at the end of this section).

As can be seen from the above example, to find the worst case response time of an AM multiframe task in the arbitrary deadline scenario, we have to check all its busy periods that include interference from the analysed task itself; and then take the maximum of them. However, to find the busy period of the q^{th} frame of τ_i we first find $r_i(q)$ that represents the time from when τ_i's critical frame is released until the q^{th} frame has finished its execution; then we subtract the execution that is related to the previous frames. The following theorem gives a formula for finding the q^{th} busy period of τ_i (i.e. $w_i(q)$).

Theorem 3 *Having a system of AM multiframe tasks, each task τ_i has an arbitrary deadline D_i, the q^{th} busy period of τ_i (i.e. $w_i(q)$) is given by Equation (4.5) assuming the priority ceiling protocols [66, 60].*

$$w_i(q) = r_i(q) - (q-1)T_i; \tag{4.5}$$

where $r_i(q)$ is found by the smallest non-negative solution to Equation (4.6).

$$r_i(q) = \xi_i^0(q) + B_i + \sum_{j=1}^{i-1} \xi_j^0(\lceil \frac{r_i(q)}{T_j} \rceil). \tag{4.6}$$

where $\xi_i^0(q)$ is introduced by Definition 1 and B_i is the maximum blocking time of τ_i.

Proof

The busy period of a task τ_i represents two kinds of invocations: one of them belongs to τ_i itself whilst the other belongs to the tasks other than τ_i. Within preemptive fixed priority scheduling, invocations of other tasks represent two kinds of invocations one

of them is interference from tasks whose priorities are higher than τ_i and the other one is blocking from tasks whose priorities are lower than τ_i.

The term that represents the interference from higher priority tasks in this scenario is $\sum_{j=1}^{i-1} \xi_j^{m_j}(\lceil \frac{w_i(q)}{T_j} \rceil)$; as long as two factors are considered. The first one is the previous critical instance (as in Definition 5); and the second one is that the q^{th} busy period of the analysed multiframe task τ_i, $w_i(q)$, is the time from when its q^{th} execution is started until this execution is finished. In addition, using priority ceiling protocols [66, 60] allows the task to be blocked at most once during its execution. However, in this model, we are analysing continuous busy periods of the same priority due to the interference from the MF task itself. So there is only one opportunity for a lower priority task to gain access to a shared resource and cause blocking. We therefore have the single term B_i, that is the maximum expected blocking. So, what is left to analyze is the interference from τ_i itself.

To analyze the interference from the analysed task itself, we consider q as the number of invocations of τ_i, so the amount of execution that τ_i provides starting from its critical frame is given by $\xi_i^{m_i}(q)$; $q = 1, 2, ...$ Therefore, the time from when the critical frame of τ_i starts its execution until achieving the q^{th} execution, $r_i(q)$, is given as a collection of three terms: the maximum blocking B_i, the interference from the higher priority AM multiframe tasks within $r_i(q)$ (i.e. $\sum_{j=1}^{i-1} \xi_j^{m_j}(\lceil \frac{r_i(q)}{T_j} \rceil)$) and the amount of execution of τ_i itself (i.e. $\xi_i^{m_i}(q)$). So, $r_i(q)$ is given by Equation (4.6).

Both $r_i(q)$ and $w_i(q)$ have same end time but different start times where the difference between the two start times is $(q-1)T_i$ having $w_i(q)$ starts at $(q-1)T_i$ after $r_i(q)$. So to find $w_i(q)$, we subtract $(q-1)T_i$ from $r_i(q)$; which is identical to Equation (4.5).□

Solving Equation (4.6) requires a recurrence relation as in Equation (4.7).

$$r_i^{l+1}(q) = \xi_i^{m_i}(q) + B_i + \sum_{j=1}^{i-1} \xi_j^{m_j}(\lceil \frac{r_i^l(q)}{T_j} \rceil); \tag{4.7}$$

Where, $r_i^0(q) = \xi_i^{m_i}(q)$ and $l = 0, 1, ..$ until finding $r_i^{l+1}(q) = r_i^l(q) = r_i(q)$. However, if $r_i^{l+1}(q)$ becomes greater than $(q-1)T_i + D_i$ we say that τ_i is unschedulable.

Theorem 3 represents a formula for finding the q^{th} busy period of τ_i. Now, we have

q	$r_2(q)$	$w_2(q)$
1	20	$20 > T_2$
2	36	$21 > T_2$
3	49	$19 > T_2$
4	58	$13 < T_2$

Table 4.3: Possible Values of the Busy Periods

to identify how many busy periods we have to consider. In other words, how many values of q we have to consider for the analysis. As the analysis is mainly interested in the interference from the analysed task itself, we will consecutively analyze the busy periods until no interference from the frames of τ_i itself occurs; which means that the busy period is finished within the same period it is released in. Therefore, q takes values as $q = 1, 2, ..$ until $w_i(q) \leq T_i$ is satisfied.

Once all needed busy periods are identified, the final step of the analysis is to find the maximum busy period which represents the worst case response time of τ_i, R_i. Symbolically, R_i is found by maximising $w_i(q)$ over all possible values of q as in the following equation $R_i = \max_{q=1,2,..} w_i(q)$.

Example

In this example, we apply the response time analysis that is presented in this section to check the schedulability of τ_2 in the example system that is given by Table 4.2. To begin with, we give a starting values for $r_2^0(q)$ as $r_2^0(q) = \xi_2^0(q)$; then, we give values to q starting from 1. So, when $q = 1$, $r_2^0(1) = \xi_2^0(1) = 10$. Then we apply Equation (4.7) for $l = 0, 1, 2$ so we get

$l = 0, \quad r_2^1(1) = \xi_2^0(1) + \xi_1^0(\lceil \frac{r_2^0(1)}{T_1} \rceil) = 10 + \xi_1^0(\lceil \frac{10}{10} \rceil) = 15,$

$l = 1, \quad r_2^2(1) = \xi_2^0(1) + \xi_1^0(\lceil \frac{r_2^1(1)}{T_1} \rceil) = 10 + 10 = 20,$

$l = 2, \quad r_2^3(1) = 20 = r_2^2(1)$. So, $r_2(1) = 20$.

Now, we find the busy period of the first frame of τ_i by applying Equation (4.5):

$w_2(1) = 20 - 0(15) = 20 > T_2$. So we increase q to be 2 and similarly we apply Equation (4.7) and (4.5) to get all possible values of $r_2(q)$ and $w_2(q)$ as in Table 4.3. As we get $w_2(4) < T_2$, we stop increasing q.

To get the worst case response time of τ_2, R_2, we now maximise over all possible busy periods in Table 4.3. Therefore, $R_2 = max\{20, 21, 19, 13\} = 21 < D_2$, so τ_2 is

schedulable[6]. Also τ_1 is schedulable because $R_1 = 5 < D_1$. Hence, the whole system example is schedulable.

4.3 Combined Analysis of Release Jitter and Arbitrary Deadlines

This section combines the two models of Sections 4.1 and 4.2 within one model and presents an exact worst case response time analysis of AM multiframe tasks that are subjected to both release jitter and arbitrary deadlines at the same time. So, each AM multiframe task τ_i has a sequence of execution times C_i, a maximum release jitter J_i, a deadline D_i, and a period T_i. In fact, when τ_i is subjected to release jitter, there could be a situation where the minimum time between two successive frames of its frames is $T_i - J_i$ instead of T_i, so having D_i greater than $T_i - J_i$ means that there could be a situation where τ_i is released more than once during its execution and therefore an interference from the analysed task τ_i itself could happen during an execution of one of its frames. So, analysis of the worst case response time of τ_i must take into account interference from τ_i itself as well as interference from other tasks in the system taking into account the situation of having two consecutive frames of a task τ_j $(j = 1,..i)$ within time interval $T_j - J_j$ instead of T_j. Without lose of generality, we assume that the first frame of each AM multiframe task τ_i is its critical frame, so $m_i = 0; \forall i = 1,..,N$.

As all MF tasks in the system satisfy the AM restriction, the situation that leads to the worst case response time of τ_i is when its critical frame is released simultaneously with the critical frames of all higher priority AM multiframe tasks. That is because the critical frame of an AM multiframe task always provides the maximum interference in the execution of the same or lower priority tasks. So, when τ_i has interference from previous invocations of its frames, the maximum generated interference from τ_i comes from when its critical frame is released.

Also, due to release jitter situation, all τ_j $(j = 1,..,i)$ could be released up to J_j units

[6]Note how the worst case response time of τ_2 does not fall into the busy period of its critical frame, but in the busy period of its second frame (i.e. the frame whose execution time is 6). .

after they arrive. So, from the preemption point of view, τ_i is preempted the most by a higher priority AM multiframe task τ_j when the critical frames of both τ_i and τ_j are simultaneously released rightmost in their release jitter interval whilst next frames are released leftmost in their release jitter interval as explained by Figure 4.1 in Section 4.1.

For more illustration, Figure 4.3 shows the execution behaviour of the example system in Table 4.4 whose both AM multiframe tasks are subjected to release jitter and arbitrary deadlines where τ_2 has deadline greater than its period. Figure 4.3

Task	C	D	T	J
τ_1	$(2,1)$	5	5	1
τ_2	$(4,3,1)$	10	6	2

Table 4.4: Example of Arbitrary Deadlines and Release Jitter

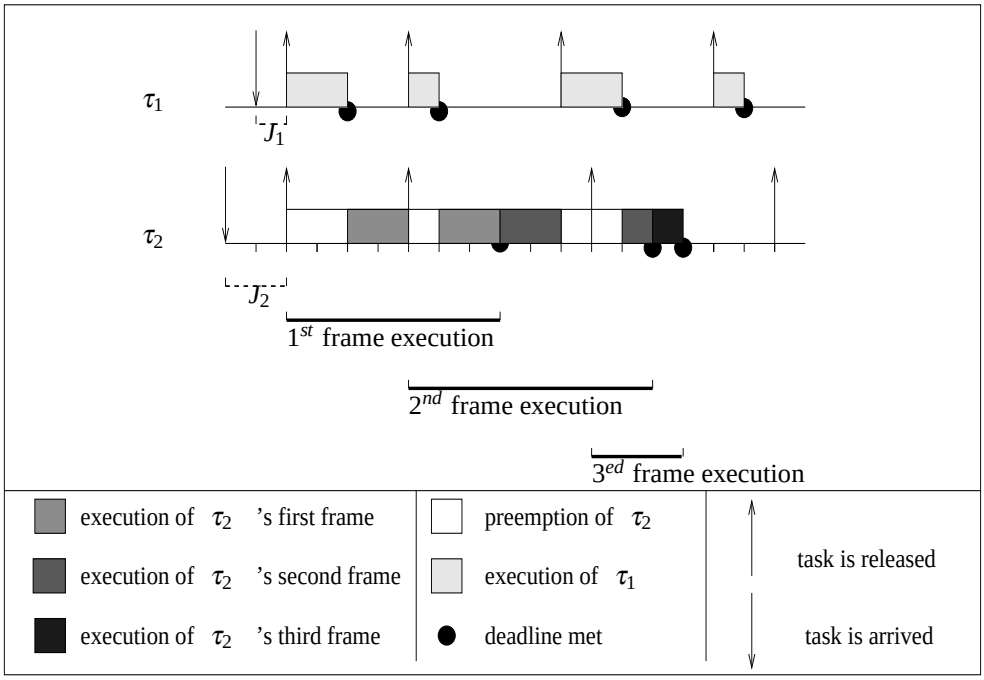


Figure 4.3: Execution of the Tasks in the Example

shows the worst case preemption of τ_2's peak frame where this preemption situation lets two frames of τ_2 to interfere with the execution of the following frames. So, τ_i's

critical instance that leads to its worst case response time can be modeled by Definition 6.

Definition 6 *The* **critical instance of an AM multiframe task τ_i, in a system subjected to release jitter and arbitrary deadlines,** *is the simultaneous release of the critical frame of τ_i with the critical frames of the higher priority multiframe tasks, taking into account that the critical frames are released at the very end of their relative release jitter interval (after their relative arrival times) whilst next frames are released at the very beginning of their relative release jitter interval (so, they are released as soon as they arrive).*

Note the differences between Definitions 4, 5 and 6. In Definition 4, the critical instance of τ_i is characterised, from the i^{th} level point of view, by its peak frame whilst in Definition 6 this critical instance is characterised by its critical frame. In addition, Definition 5 assumes that the arrival time of τ_i is the same as its release time for all τ_i's frames whilst Definition 6 assumes that release time of τ_i's critical frame is after its arrival time.

Analysis in this section considers Definition 6 to analyze τ_i's worst case response time. As a first step of the worst case response time analysis of τ_i, we define the busy period of a frame of τ_i as the time from when this frame is released until it finishes its execution. So, the worst case response time of τ_i is the maximum busy period of τ_i over all τ_i's frames that include interference from τ_i itself. Assume q is the number of invocations of τ_i $(q = 1, 2, ..)$, to find the busy period of the q^{th} frame of τ_i we follow two steps: first we find $r_i(q)$ that represents the time from when τ_i's critical frame is released until the q^{th} frame has finished its execution; then we subtract the execution that is related to the previous frames. The following theorem proves the technique that is used to find the q^{th} busy period of τ_i.

Theorem 4 *Having a system of AM multiframe tasks, each task τ_i has an arbitrary deadline D_i and is subjected to release jitter J_i, the q^{th} busy period of τ_i (i.e. $w_i(q)$) is given by Equation (4.8) assuming the priority ceiling protocols [66, 60].*

$$\begin{aligned} w_i(q) &= r_i(q); && for\ q = 1, \\ &= r_i(q) - (q-1)T_i + J_i; && for\ q > 1. \end{aligned} \tag{4.8}$$

where $r_i(q)$ is found by the smallest non-negative solution to Equation (4.9).

$$r_i(q) = \xi_i^0(q) + B_i + \sum_{j=1}^{i-1} \xi_j^0(\lceil \frac{r_i(q) + J_j}{T_j} \rceil). \tag{4.9}$$

where $\xi_i^0(q)$ is introduced by Definition 1 and B_i is the maximum blocking time of τ_i.

Proof

$r_i(q)$ represents two kinds of execution; one is related to the execution of τ_i and the other is related to MF tasks other than τ_i. The execution that is related to τ_i is represented by its cumulative function $\xi_i^0(q)$ and the execution that is related to the MF tasks other than τ_i is represented by blocking from lower priority tasks and interference from higher priority tasks.

As priority ceiling protocols allow the task to be blocked at most once during its execution and as $r_i(q)$ is a continuous execution of the same priority MF task, the blocking term from lower priority tasks is represented by the maximum expected blocking time B_i. Furthermore, as we assume the simultaneous release of τ_i and higher priority tasks (Definition 6 of the critical instance of τ_i), the interference from the MF tasks whose priorities are higher than τ_i is presented by a summation of all interference from those tasks.

Assume $\tilde{I}_j$ is the interference from a higher priority AM multiframe task τ_j in $r_i(q)$, applying Lemma 1 leads to $\tilde{I}_j$ being presented by $\xi_j^{m_j}(\lceil \frac{r_i(q)+J_j}{T_j} \rceil$. So, the maximum interference from the MF tasks whose priorities are higher than τ_i's is presented by $\sum_{j=1}^{i-1} \xi_j^0(\lceil \frac{r_i(q)+J_j}{T_j} \rceil)$. Therefore, $r_i(q)$ is a collection of $\xi_i^0(q)$, B_i and $\sum_{j=1}^{i-1} \xi_j^0(\lceil \frac{r_i(q)+J_j}{T_j} \rceil)$; which is identical to Equation (4.9).

$r_i(q)$ consists of q number of τ_i's execution starting from τ_i's critical frame. So, the first busy period of τ_i is the busy period of τ_i's critical frame. In addition, $w_i(q)$ starts from when the q^{th} frame of τ_i is released whilst $r_i(q)$ starts from when the first frame is released; also, both of $w_i(q)$ and $r_i(q)$ have the same end duration. So, when $q = 1$ both of $w_i(1)$ and $r_i(1)$ have the same start and end duration; which means that $w_i(1) = r_i(1)$. However, when $q > 1$, $w_i(q)$ and $r_i(q)$ have different starts where the first frame starts its execution at J_i and the q^{th} frame starts its execution at

$(q-1)T_i - J_i$. So, $w_i(q) = r_i(q) - ((q-1)T_i - J_i) = r_i(q) - (q-1)T_i + J_i$ which is identical to Equation (4.8).□

Equation (4.9) is solved by a recurrence relationship given by Equation (4.10); where $r_i^0(q) = \xi_i^0(q)$ and $l = 0,1,2,...$ $r_i(q)$ is found once $r_i^{l+1}(q) = r_i^l(q)$ is satisfied. However, if $r_i^{l+1}(q) > (q-1)T_i - J_i + D_i$ we say that the AM multiframe task τ_i is unschedulable because one of τ_i's frames could miss its deadline in this case.

$$r_i^{l+1}(q) = \xi_i^0(q) + B_i + \sum_{j=1}^{i-1} \xi_j^0(\lceil \frac{r_i^l(q) + J_j}{T_j} \rceil). \tag{4.10}$$

Final step of the worst case response time analysis of τ_i in this section is to identify the upper bound of q that we have to consider in the response time analysis. In other words, how many invocations of τ_i we have to consider in the analysis. Actually, q takes values from 1 until no interference from τ_i occurs; which happens when the relative busy period falls in the same period that τ_i is released in. In other words, $q = 1,2,...$ until we get $w_i(q) < T_i - J_i$ for $q = 1$ or $w_i(q) < T_i$ for $q > 1$. Therefore, the worst case response time of τ_i, R_i, is the maximum busy period over all values of q. Symbolically,

$$R_i = \max_{q=1,2,..} \{w_i(q)\}.$$

As the deadline of a task is relative to the arrival time of the task while the response time is relative to the release time, the scheduling test of the model in this section is the following: τ_i is schedulable if its worst case response time R_i is less than or equal to $D_i - J_i$.

As this section generalises the analysis of both analyses in Sections 4.1 and 4.2, the following section presents an example that applies all details of the worst case response time analysis that is presented in this section.

4.4 Example

Assume the system in Table 4.4, with no blocking assumed. To find the worst case response time of τ_2, we first find the busy periods $w_i(q)$ depending on $r_i(q)$ by applying

Equations (4.10) and (4.8) for $i = 2$ and $r_2^0(q) = \xi_2^0(q)$, so we get:
$r_2^{l+1}(q) = \xi_2^0(q) + B_i + \sum_{j=1}^{1} \xi_j^0(\lceil \frac{r_2^l(q)+J_j}{T_j} \rceil)$.

$q = 1, \quad r_2^{l+1}(1) = \xi_2^0(1) + \xi_1^0(\lceil \frac{r_2^l(1)+J_1}{T_1} \rceil)$. To solve this equation,

$$
\begin{aligned}
l = 0, \quad r_2^1(1) &= \xi_2^0(1) + \xi_1^0(\lceil \frac{r_2^0(1)+J_1}{T_1} \rceil) \\
&= 4 + \xi_1^0(\lceil \frac{4+1}{5} \rceil) \\
&= 4 + 2 = 6. \\
l = 1, \quad r_2^2(1) &= \xi_2^0(1) + \xi_1^0(\lceil \frac{r_2^1(1)+J_1}{T_1} \rceil) \\
&= 4 + \xi_1^0(\lceil \frac{6+1}{5} \rceil) \\
&= 4 + 3 = 7. \\
l = 2, \quad r_2^3(1) &= \xi_2^0(1) + \xi_1^0(\lceil \frac{r_2^2(1)+J_1}{T_1} \rceil) \\
&= 4 + \xi_1^0(\lceil \frac{7+1}{5} \rceil) \\
&= 4 + 3 = 7 = r_2^2(1).
\end{aligned}
$$

So, $r_2(1) = 7$, therefore $w_2(1) = r_2(1) = 7$.
$w_2(1) > T_2 - J_2$, so we increase q to 2 and apply Equations (4.10) and (4.8) for $i = 2$, $q = 2$ and $r_2^0(2) = \xi_2^0(2) = 7$, so we get

$q = 2, \quad r_2^{l+1}(2) = \xi_2^0(2) + \xi_1^0(\lceil \frac{r_2^l(2)+J_1}{T_1} \rceil)$. To solve this equation,

$$
\begin{aligned}
l = 0, \quad r_2^1(2) &= \xi_2^0(2) + \xi_1^0(\lceil \frac{r_2^0(2)+J_1}{T_1} \rceil) \\
&= 7 + \xi_1^0(\lceil \frac{7+1}{5} \rceil) \\
&= 7 + 3 = 10. \\
l = 1, \quad r_2^2(2) &= \xi_2^0(2) + \xi_1^0(\lceil \frac{r_2^1(2)+J_1}{T_1} \rceil) \\
&= 7 + \xi_1^0(\lceil \frac{10+1}{5} \rceil)
\end{aligned}
$$

$$= 7+5 = 12.$$

$$l = 2, \quad r_2^3(2) = \xi_2^0(2) + \xi_1^0(\lceil \frac{r_2^2(2)+J_1}{T_1} \rceil)$$
$$= 7 + \xi_1^0(\lceil \frac{12+1}{5} \rceil)$$
$$= 7+5 = 12 = r_2^2(2).$$

So, $r_2(2) = 12$, therefore $w_2(2) = r_2(1) - T_2 + J_2 = 12 - 6 + 2 = 8$. $w_2(1) > T_2$, so we increase q to 3 and again apply Equations (4.10) and (4.8) for $i = 2$, $q = 3$ and $r_2^0(3) = \xi_2^0(3) = 8$, so we get

$q = 3, \quad r_2^{l+1}(3) = \xi_2^0(3) + \xi_1^0(\lceil \frac{r_2^l(3)+J_1}{T_1} \rceil)$. To solve this equation,

$$l = 0, \quad r_2^1(3) = \xi_2^0(3) + \xi_1^0(\lceil \frac{r_2^0(3)+J_1}{T_1} \rceil)$$
$$= 8 + \xi_1^0(\lceil \frac{8+1}{5} \rceil)$$
$$= 8+3 = 11.$$
$$l = 1, \quad r_2^2(3) = \xi_2^0(3) + \xi_1^0(\lceil \frac{r_2^1(3)+J_1}{T_1} \rceil)$$
$$= 8 + \xi_1^0(\lceil \frac{11+1}{5} \rceil)$$
$$= 8+5 = 13.$$
$$l = 2, \quad r_2^3(3) = \xi_2^0(3) + \xi_1^0(\lceil \frac{r_2^2(3)+J_1}{T_1} \rceil) = 8 + \xi_1^0(\lceil \frac{13+1}{5} \rceil) = 13 = r_2^2(3).$$

So, $r_2(3) = 13$, therefore $w_2(3) = r_2(3) - 2T_2 + J_2 = 13 - 12 + 2 = 3$. $w_2(1) < T_2$, so we stop increasing q. Hence, the worst case response time of τ_2 is the maximum of the busy periods we have got

$$R_2 = \max\{7, 8, 3\} = 8.$$

As a result $R_2 \leq D_2 - J_2$, so τ_2 is schedulable[7], also τ_1 is schedulable because $R_1 = 2 < D_1 - J_1$. Therefore, the whole example system that is given in Table 4.4 is schedulable.

[7] Note how the worst case response time of τ_2 is fallen in the busy period of its second frame.

4.5 Summary

This chapter has shown that the worst case response time analysis of AM multiframe tasks is tractable and flexible enough to be extended in two directions. Firstly, the worst case response time analysis is applicable to the system model whose AM multiframe tasks are subjected to release jitter. Secondly the worst case response time analysis is applicable to the system model whose AM multiframe tasks have arbitrary deadlines. Furthermore, this chapter gives full details of the exact worst case response time analysis of AM multiframe tasks that are subjected to release jitter and arbitrary deadlines at the same time.

5 Exact Scheduling Analysis of Non-AM Multiframe Tasks

The previous two chapters presented the worst case response time analysis of multiframe tasks that are restricted to satisfy the AM restriction. In this chapter[1], the restriction of having AM multiframe tasks is relaxed, so, the assumption that having only one critical frame per MF task is not satisfied any more; which affects the response time analysis of the MF tasks. Initially, the worst case response time analysis of the general MF task τ_i requires checking all possible combinations of all frames of the MF tasks whose priorities are higher than τ_i; which means we have to consider $\prod_{j=1}^{i-1} n_j$ different combinations[2] of the frames [69]. However, having introduced the critical frame concept (see Section 2.1) leads to the requirement of only checking the critical frames of the MF tasks whose priorities are higher than τ_i's. Evaluation shows that this usage of the critical frames reduces the number of required combinations for finding the worst case response time of τ_i.

This chapter is organised as follows: Section 5.1 introduces a criterion to identify the critical frames per MF task. Using critical frames, Section 5.2 presents the exact response time formula of general MF tasks. Section 5.3 explains the application of the response time analysis of general MF tasks by a numeric example. Section 5.4 gives a formal evaluation of the number of critical frames in practice.

[1] Material based on Sections 5.1 and 5.2 in this chapter was published in [79].

[2] remember that $\prod_{j=1}^{i-1} n_j$ means the product of n_j for $j = 1,..,i-1$. $\prod_{j=1}^{i-1} n_j = n_1.n_2.n_3. \ldots n_{i-1}$.

5.1 Identifying the Critical Frames

Recall Definition 2 in Section 2.1, a frame of location x is considered critical when this frame provides a maximum interference for at least one number of its invocations. However, this definition is not enough for non-AM multiframe tasks, for example Table 5.1 shows all possible interference from a MF task τ_j with an execution time sequence $(8,5,7,6,8,5)$. Applying Definition 2 on τ_j leads to having only one critical frame whose execution time is 6. However, this criterion does not cover all critical frames of a non-AM multiframe task as the critical frame, whose execution time is 6, is not the only critical frame because it does not provide the maximum interference in the case of one, three and five interference from τ_j. So, Definition 2 does not identify all critical frames of a non-AM multiframe task. This is because there could be a frame, of a MF task τ_j, that does not satisfy Equation (2.2) but is critical; because, from one side, it provides the maximum interference for a specific number of τ_j's invocations, and from another side, the frames that satisfy Equation (2.2) do not provide more interference than it does. This section presents a criterion for identifying the set of critical frames for a non-AM multiframe task.

Location of Released Frame	exe. seq.	1 inv.	2 inv.	3 inv.	4 inv.	5 inv.	6 inv.
0	(8,5,7,6,8,5)	8	13	20	26	34	39
1	(5,7,6,8,5,8)	5	12	18	26	34	39
2	(7,6,8,5,8,5)	7	13	21	26	34	39
3	(6,8,5,8,5,7)	6	**14**	19	**27**	32	39
4	(8,5,8,5,7,6)	8	13	21	26	33	39
5	(5,8,5,7,6,8)	5	13	18	25	31	39

Table 5.1: Possible Interference From τ_j

To identify the critical frames of a MF task, we follow a reversing scenario where we first identify the non-critical frames then consider the remaining frames of this MF task as critical. To identify the non-criticality of a frame whose execution time is C_j^y of the MF task τ_j, we invert Definition 2 in Section 2.1. So, we say that the frame whose execution time is C_j^y is not critical if there is another frame of τ_j whose execution time is C_j^x; where the amount of interference that this frame provides is always greater

than or equal to the amount of interference that the frame whose execution time is C_j^y provides for all number of τ_j's invocations. In other words, the cumulative function of the frame whose execution time is C_j^x is always greater than or equal to the cumulative function of the frame whose execution time is C_j^y for all number of τ_j's invocations. However, we sufficiently consider only $n_j - 1$ invocations of τ_j because the amount of the generated interference from any of τ_j's frames increases with a fixed rate after $n_j - 1$ invocations.

To symbolise the definition of the non-critical frame of a MF task τ_j having n_j execution times $(C_j^0, C_j^1, .., C_j^{(n_j-1)})$ within its shortest form, we consider the frame whose execution time is C_j^y is definitely not critical if $\exists x = 0, .., n_j - 1$ *and* $x \neq y$; where Equation (5.1) is satisfied $\forall k = 1, 2, .., n_j - 1$

$$\xi_j^x(k) \geq \xi_j^y(k). \tag{5.1}$$

The non-criticality criterion that is represented by Equation (5.1) means that the amount of interference the frame whose execution time is C_j^y generates is never more than the amount of interference the frame whose execution time is C_j^x generates, so the frame whose execution time is C_j^y is never critical. We call the frame whose execution time is C_j^y in this case the *dominated frame*. So, applying this criterion on all frames of a MF task judges the non-critical frames, and therefore the remaining frames of the MF task are critical.

In fact, although this criterion of identifying critical frames is safe, it does not provide an optimal set of critical frames of a MF task. This is because there could be a frame, in the generated set, that is dominated by more than one other frames in the same generated set. However, finding the minimum set of critical frames is equivalent to the set-covering problem[4] and is known to be NP-complete, so we apply the non-criticality criterion which is tractable. One successful application of this criterion results in the frames whose execution time is the minimum are never critical, the following theorem proves this.

Theorem 5 *Given a MF task τ_i whose execution time is in its shortest form, with n_i frames where $n_i > 1$, a minimum frame[3] is never a critical frame.*

[3]The minimum frame is the frame whose execution time is the minimum value of the execution times.

Proof

The cumulative functions of the two frames whose execution times are respectively the minimum and next to the minimum are respectively given by Equations (5.2) and (5.3)

$$\xi_i^{min}(k) = \sum_{j=min}^{min+k-1} C_i^{(j \bmod n_i)}, \tag{5.2}$$

$$\xi_i^{min+1}(k) = \sum_{j=min+1}^{min+k} C_i^{(j \bmod n_i)}; \tag{5.3}$$

where *min* is the location of the minimum execution time in its sequence.
For each $k = 1, 2, ..$, we subtract Equation (5.2) from Equation (5.3), so we get

$$\xi_i^{min+1}(k) - \xi_i^{min}(k) = C_i^{((min+k) \bmod n_i)} - C_i^{(min \bmod n_i)}.$$

As C_i^{min} is the minimum execution time of all frames, the right side of the equation is never negative so the left side of the equation is also never negative. So, $\xi_i^{min+1}(k) \geq \xi_i^{min}(k); \forall k = 1, 2, ..$; which means that each frame with the minimum execution time is always dominated by the frame it is followed by. □

Corollary 1 *When a MF task has more than one minimum in its sequence of execution times, then all minimum frames are not critical.*

Proof

Followed directly from Theorem 5 where each minimum frame (i.e. the frame with the minimum execution time) is dominated by the followed frame, which means that all minimum frames are not critical.□

For example, Table 5.2 presents all possible interference from a MF task τ_j with an execution time sequence $(8, 3, 8, 3, 3, 4)$. We can see from this table that, for all number of interference, the amount of interference the minimum frame provides is always less than or equal to the amount of interference the followed frame provides. So, each minimum frame is dominated by the frame that is followed by and therefore all minimum frames are non-critical.

Theorem 5 shows that in the worst case, when there is only one minimum frame of τ_i and there is no dominated frames other than one minimum, there is a maximum

Location of Released Frame	exe. seq.	1 inv.	2 inv.	3 inv.	4 inv.	5 inv.	6 inv.
0	(8,3,8,3,3,4)	8	11	19	22	25	29
1	**(3,8,3,3,4,8)**	3	**11**	**14**	**17**	**21**	**29**
2	(8,3,3,4,8,3)	8	11	14	18	26	29
3	**(3,3,4,8,3,8)**	**3**	**6**	**10**	**18**	**21**	**29**
4	**(3,4,8,3,8,3)**	**3**	**7**	**15**	**18**	**26**	**29**
5	(4,8,3,8,3,3)	4	12	15	23	26	29

Table 5.2: Possible Interference From τ_j

of $n_i - 1$ critical frames as only this minimum is definitely non-critical. So, for a MF task with at least two different execution times, the frames that have to be checked for the critical criterion are the frames whose execution times are not minimum. So, in the worst case, the maximum number of enumeration that is needed in evaluating the response time of a MF task τ_i is $\prod_{j=1}^{i-1}(n_j - 1)$ and not as previously claimed $\prod_{j=1}^{i-1} n_j$ [69].

Moreover, dominated frames, of which the minimums are one example, are never critical while the remaining frames are critical. So, the number of enumeration that is needed in evaluating the response time of a MF task τ_i could be even less than $\prod_{j=1}^{i-1}(n_j - 1)$ when the dominated frames are discard from the criticality criterion.

Once the critical frame set for each MF task is identified, the worst case response time analysis uses the combinations of the critical frame sets of higher priority MF tasks to find the worst case response time of a lower priority MF task. The critical frame set is represented by the locations of the relative critical frames in the MF task, so the combinations of the critical frame sets are relatively represented by the combinations of the indices of the critical frames. So, assume $\hat{L}_j$ is the set of the critical frame locations of the MF task τ_j. Then, from $\hat{L}_j$ we define $\hat{V}_i$ to represent the combinations of the critical frames of higher priority MF tasks as the cartesian product of all sets of the critical frame locations for all tasks whose priorities are higher than τ_i. This cartesian product $\hat{V}_i$ is defined as follow:
Let $\hat{V}_1 = \{\}$, $\hat{V}_2 = \hat{L_1}$ and for $i > 2$ define $\hat{V}_i$ to be the cartesian product of $\hat{L_1}$, ..., $\hat{L_{i-1}}$.

In other words,

$$\hat{V}_i = \hat{L_1} \times \hat{L_2} \times .. \times \hat{L_{i-1}}.$$

For example, assume we have $\hat{L_1} = \{1,2,4\}$, $\hat{L_2} = \{0,1\}$ and $\hat{L_3} = \{3,6\}$. Then $\hat{V}_1$, $\hat{V}_2$ and $\hat{V}_3$ are found as follows
$\hat{V}_1 = \{\}$,
$\hat{V}_2 = \hat{L_1} = \{(1),(2),(4)\}$,
$\hat{V}_3 = \hat{L_1} \times \hat{L_2} = \{(1,0),(1,1),(2,0),(2,1),(4,0),(4,1)\}$.
The following section uses $\hat{V}_i$ in presenting the worst case response time analysis.

5.2 Exact Response Time Analysis of Non-AM Multiframe Tasks

This section presents the response time analysis of multiframe tasks that do not satisfy the AM restriction. The system model in this section assumes the basic MF model that is introduced in Section 2.1. In this model, all MF tasks are not subjected to release jitter and no interference from the analysed task is permitted. However, sharing resources is permitted and is represented for each MF task τ_i by the maximum blocking time B_i.

Usually, the first step of analysing the worst case response time of τ_i is to identify the situation that leads to this worst case response time. This situation is called the critical instance of τ_i. From the preemption point of view, τ_i's response is the worst when τ_i is preempted the most. In addition, τ_i is preempted the most when the amount of interference from the higher priority MF tasks is the maximum. As the critical frames of a MF task are the only frames that provide the maximum interference in the execution of lower priority MF tasks, we now identify the critical instance of a MF task τ_i as in Definition 7; where the peak frame of τ_i is the frame that generates the worst case execution amount of τ_i assuming no interference from τ_i itself.

Definition 7 . The critical instance of a MF task τ_i *is the simultaneous release of the peak frame of* τ_i *with the critical frames of the higher priority MF tasks, that lead to the worst case response time of* τ_i.

Assuming the critical instance in Definition 7, the response time analysis of τ_i considers its peak frame and the previous reduced set of critical frames for each MF task whose priority is higher than the priority of τ_i. So, to find the worst case response time of τ_i we have to maximise its response time over all critical frame combinations of the higher priority MF tasks. Symbolically, the worst case response time of τ_i has to be maximised over all values in $\hat{V}_i$; which is given by Equation (5.4).

$$R_i = \max_{\tilde{v} \in \hat{V}_i} \{R_{i,\tilde{v}}\}; \tag{5.4}$$

where $\{R_{i,\tilde{v}}\}$ is the response time of τ_i that is relative to the simultaneous release, of critical frames of higher priority MF tasks, that is presented by the combination $\tilde{v}$ from the cartesian product $\hat{V}_i$ and is found by Equation (5.5) as in the following theorem.

Theorem 6 *$R_{i,\tilde{v}}$ is the worst case response time, of a non-AM multiframe task τ_i, that is relative to $\tilde{v}$ which represents one of the simultaneous releases of the critical frames of higher priority MF tasks. Assuming Definition 7, $R_{i,\tilde{v}}$ is given by Equation (5.5) assuming priority ceiling protocols [66, 60].*

$$R_{i,\tilde{v}} = C_i^{m_i} + B_i + \sum_{j=1}^{i-1} \xi_j^{\tilde{v}_j} \left(\lceil \frac{R_{i,\tilde{v}}}{T_j} \rceil\right); \tag{5.5}$$

where m_i is a location of a peak frame of the MF task τ_i. $\tilde{v}_j$ is the j^{th} element of the vector $\tilde{v}$ (i.e. the index of τ_j's critical frame that is relative to the combination $\tilde{v}$).

Proof As we are assuming a simultaneous release of both τ_i and higher priority MF tasks, the worst case response time of τ_i can be presented by a summation of the worst case execution of τ_i, maximum interference from higher priority MF tasks within this execution, and maximum blocking from lower priority tasks B_i as priority ceiling protocols let the task to be blocked at most once during its execution. The worst case execution of τ_i is represented by the execution time of its peak frame (i.e. $C_i^{m_i}$).

On the other hand, the interference from the higher priority MF tasks are presented by a summation of the interference from each higher priority MF task. Assume $\tilde{I}$ is the amount of interference that is generated by the MF tasks whose priorities are higher

than τ_i's,

$$\tilde{I} = \sum_{j=1}^{i-1} \tilde{I}_j; \tag{5.6}$$

where $\tilde{I}_j$ is the amount of interference that is generated by the MF task τ_j.

We already know that $R_{i,\tilde{v}}$ starts from when τ_i is released; and τ_i is released simultaneously with τ_j which is released every period T_j; so the number of interference from τ_j within $R_{i,\tilde{v}}$ is

$$\lceil \frac{R_{i,\tilde{v}}}{T_j} \rceil.$$

However, τ_j is first released having an execution time $C_j^{\tilde{v}_j}$, so the amount of interference that is generated by τ_j is given by:

$$\tilde{I}_j = \xi_j^{\tilde{v}_j} \, (\lceil \frac{R_{i,\tilde{v}}}{T_j} \rceil).$$

Therefore, substituting $\tilde{I}_j$ in Equation (5.6) ends up with Equation (5.5).□

Equation (5.5) is solved by forming a recurrence relation given by Equation (5.7).

$$R_{i,\tilde{v}}^{l+1} = C_i^{m_i} + B_i + \sum_{j=1}^{i-1} \xi_j^{\tilde{v}_j} \, (\lceil \frac{R_{i,\tilde{v}}^{l}}{T_j} \rceil) \tag{5.7}$$

where[4] $R_{i,\tilde{v}}^0 = C_i^{m_i}$ and $l = 0, 1, ..$ until $R_{i,\tilde{v}}^{l+1} = R_{i,\tilde{v}}^l$. However, if $R_{i,\tilde{v}}^{l+1}$ becomes greater than D_i, we say that τ_i is not schedulable.

5.3 Numeric Example

This section presents a simple example system to illustrate the application of the presented exact response time analysis of the non-AM multiframe tasks. Table 5.3 represents the parameters of this example that consists of three MF tasks τ_1, τ_2 and τ_3.

This example shows how using the critical frames in the analysis reduces the num-

[4]To reduce the number of iterations over calculations for the response time of a MF task τ_i, an alternative value of $R_{i,\tilde{v}}^0$ can be found as the minimum interference within the execution of τ_i's peak frame (i.e. $R_{i,\tilde{v}}^0 = \sum_{j=1}^{i-1} \min_{x \in L_j} \{\xi_j^x (\lceil \frac{C_i^{m_i}}{T_j} \rceil)\}$.

task	C	$T = D$	priority
τ_1	3, 4, 6, 8, 7, 5	10	high
τ_2	5, 6, 10, 7	40	medium
τ_3	1, 2, 3	60	low

Table 5.3: Example System

ber of combinations that are needed for the response time analysis. For example, previously before presenting the critical frame concept, we had to evaluate the response time of τ_3 over all possible combinations of the frames of τ_1 and τ_2; which means we have to do 24 evaluations (because τ_1 has 6 frames and τ_2 has 4 frames so the number of combinations is $6 \times 4 = 24$). However, as minimum frames are not critical, the number of evaluations reduces, in the first step, to $5 \times 3 = 15$. Also, as dominated frames are never critical, so considering only the critical frames of both τ_1 and τ_2 reduces the number of needed evaluations to only 6 as explained below.

To find the critical frames of τ_1 and τ_2, we first find the cumulative functions for each frame. Tables 5.4 and 5.5 show the amount of cumulative functions each frame of each MF task τ_1 and τ_2 generates; which are represented by the function ξ, (1 inv. means $k = 1$ for $\xi_j^x(k)$, and so on).

frame location	1 inv.	2 inv.	3 inv.	4 inv.	5 inv.
0	3	7	13	21	28
1	**4**	**10**	**18**	**25**	**30**
2	**6**	**14**	**21**	**26**	**29**
3	**8**	**15**	**20**	**23**	**27**
4	7	12	15	19	25
5	5	8	12	18	26

Table 5.4: Cumulative Functions of τ_1

Once all cumulative functions are found, we apply Equation (5.1) to each frame to identify the critical ones[5]. So, τ_1 and τ_2 have less than $n_j - 1$ critical frames; for $j = 1, 2$; where applying Equation (5.1) to τ_1 shows that the frame with the execution time 8 dominates both frames with the execution times 7 and 5. So, both frames

[5]Note how the minimum frame is always dominated by the frame that it is followed by.

frame location	1 inv.	2 inv.	3 inv.
0	5	11	21
1	**6**	**16**	**23**
2	**10**	**17**	**22**
3	7	12	18

Table 5.5: Cumulative Functions of τ_2

with the execution times 7 and 5 are never critical. The same argument is applied to the frame with the execution time 10 in τ_2; where it dominates the frame with the execution time 7. So, the critical frame locations of τ_1 and τ_2 which are presented by $\hat{L_1}$ and $\hat{L_2}$ are $\hat{L_1} = \{1,2,3\}$ and $\hat{L_2} = \{1,2\}$. As a result,
$\hat{V_1} = \{\}$
$\hat{V_2} = \{(1),(2),(3)\}$
$\hat{V_3} = \hat{L_1} \times \hat{L_2} = \{(1,1),(1,2),(2,1),(2,2),(3,1),(3,2)\}$

Therefore, to find the worst case response time of τ_2 and τ_3 we have to evaluate their response time over the critical frames of τ_1 and τ_2 while τ_1 is the highest priority MF task, its worst case response time is $R_1 = 8 < D_1$. So, τ_1 is already schedulable. For τ_2 and τ_3 we apply Equation (5.7) for the relative $\tilde{v_i}$.

For the worst case response time of τ_2,
$R^1_{2,(1)} = 10 + \sum_{j=1}^{1} \xi_j^{\tilde{v_j}} (\lceil \frac{10}{T_j} \rceil) = 10 + \xi_1^1 (\lceil \frac{10}{10} \rceil) = 10 + 4 = 14,$
$R^2_{2,(1)} = 10 + \sum_{j=1}^{1} \xi_j^{\tilde{v_j}} (\lceil \frac{14}{T_j} \rceil) = 10 + 10 = 20,$
$R^3_{2,(1)} = 10 + \sum_{j=1}^{1} \xi_j^{\tilde{v_j}} (\lceil \frac{20}{T_j} \rceil) = 10 + 10 = 20 = R^2_{2,(1)}$. So, $R_{2,(1)} = 20$
Similarly, we find that $R_{2,(2)} = 36$ and $R_{2,(3)} = 30$. So, $R_2 = \max\{20,30,36\} = 36$.
$R_2 < D_2$ so τ_2 is schedulable.

To find the worst case response time of τ_3, for each combination $\tilde{v} \in \hat{V_3}$, we find the relative response time of τ_3 by applying Equation (5.7). For example, to find $R_{3,(1,1)}$, we do the following $R^0_{3,(1,1)} = 3$,

$R^1_{3,(1,1)} = 3 + \sum_{j=1}^{2} \xi_j^{\tilde{v_j}} (\lceil \frac{3}{T_j} \rceil) = 3 + 4 + 6 = 13,$

$R^2_{3(1,1)} = 3 + \sum_{j=1}^{2} \xi_j^{\tilde{v_j}} (\lceil \frac{13}{T_j} \rceil) = 19$

$R^3_{3,(1,1)} = 19 = r^2_{3(1,1)}$. So, $R_{3,(1,1)} = 19$.

Similarly we find all $R_{3,\tilde{v}}$ for all elements in $\hat{V}_3$ to get the values in Table 5.6. Therefore[6], $R_3 = \max\{19,30,29,38,39,36\} = 39$. $R_3 < D_3$, so τ_3 is schedulable.

frame location	0	1	2	3	4	5
0	-	-	-	-	-	-
1	-	19	30	29	-	-
2	-	38	**39**	36	-	-
3	-	-	-	-	-	-

Table 5.6: Possible Response Times of τ_3

As τ_1, τ_2 and τ_3 are schedulable, the whole example system is schedulable.

Although evaluating the exact worst case response time is still formally an intractable problem as in the worst case there could be a maximum of $\prod_j (n_j - 1)$ evaluations, the exact analysis can be applied to many non-AM multiframe tasks. The following section investigates the number of critical frames likely to occur in practice.

5.4 Evaluating the Number of Critical Frames

In this section, we evaluate the number of critical frames that are likely to occur to see how often we could optimise the response time analysis in general. This is done in summary by generating a set of random execution time sequences; which represent the execution time sequence of the MF tasks. Then, for each execution time sequence we find its relative number of critical frames. The following two sections show the scope of the experiments (i.e. choosing parameters and how each of the experiments is running) while the last section presents the results of the experiments.

[6]Note the maximum is only over 6 values instead of 24.

5.4.1 Experimental Setup

The experiments in this chapter require the generation of a multiframe task to find its relative number of critical frames. Generating the multiframe task, in its turn, requires generating two parameters, first one is the number of frames of the multiframe task and second one is the shortest form of the execution time sequence.

To guarantee the execution time sequence to be in its shortest form (as we request in the system model, see Section 2.1 for details), the experiments are done for all prime numbers in the range $[3,29]$ to be as the number of frames. That is because a sequence with at least two different values and of a prime size can not consist of repetitive sub sequences. After identifying the number of frames of the MF task, we randomly generate the execution times using a uniform distribution. Values of the execution time sequences are randomly generated within two ranges $[1,10]$ and $[100,200]$. This is to try different ratios of the execution time values; where the ratio of the first range is 5 times the second one.

5.4.2 Scope of Running the Experiments

The experiments are done in two sessions: one session is when execution times are generated within range $[1,10]$ and the other is when execution times are generated within range $[100,200]$. Within each session we run the experiment 10000 times for each chosen number of frames in four steps as following. Firstly, we construct a multiframe task by generating the parameters of the experiment (i.e. number of frames and execution time sequence) as explained in the previous subsection (i.e. Section 5.4.1). Secondly, we count the number of critical frames of the generated parameters (i.e. the generated MF task) by checking Equation (5.1) for each of its frames. Algorithm 3 represents the psodocode of the algorithm used for finding the indeces and the number of critical frames. Thirdly, we repeat this experiment for the same parameters 10000 times and then calculate the mean number of critical frames. Lastly, to present clearer overview on the generated data with the knowledge that the number of critical frames does not exceed $n-1$ where n is the number of frames, we find the most frequent number of critical frames that appears the most within the 10000 set for the

same generated parameters.

Each experiment is done for each prime number of frames within range $[3,29]$ and, as can be seen from Figure 5.1, each graph (i.e. $[1,10]$ and $[100,200]$) has three lines two of them represent the two functions: the mean number of critical frames and the most frequent number of critical frames; and the third line represents the lowest bound of the percentage that is greater than the two mentioned functions of the critical frames. Consequently, each point in Figure 5.1 represents one of two options (i.e. regarding to the line that it belongs to). The first option is the mean number of critical frames for the relative number of frames out of 10000 set of randomly generated execution times. The second option is the number of critical frames that is appeared the most, for the same relative number of frames, in the same 10000 set of generated execution times.

5.4.3 Algorithms of the experiment

To clarify the experiment steps', we present here all algorithms that are related to the experiments. To start with, Algorithm 3 represents the required steps for finding the number of critical frames.

Algorithm 3 Finding Number of Critical Frames

Inputs: Array Execution_Times.
Outputs: Array Critical_Indices, Integer Size_Criticals.

```
if Frames_Number = 1 then
  Critical_Indices ⇐ 0
  Size_Criticals ⇐ 1
else
  Cumulat_Matrix ⇐ cumulative matrix of Execution_Times
  Size_Criticals ⇐ Frames_Number − 1
  for i = 0 to Frames_Number - 1 do
    for j = 0 to Frames_Number - 1 do
      Counter ⇐ 0
      if i ≠ j then
        for k = 0 to Frames_Number - 1 do
          if Cumulat_Matrix(i,k) ≤ Cumulat_Matrix(j,k) then
            Counter ⇐ Counter + 1
          end if
        end for
        if Counter=Frames_Number - 1 then
          Execution_Times ⇐ − 1
          Size_Criticals ⇐ Size_Criticals − 1
        end if
      end if
    end for
  end for
  Counter2 ⇐ 0
  for i = 0 to Frames_Number - 1 do
    if Execution_Times(i) ≠ −1 then
      Critical_Indices(Counter2) ⇐ i
      Counter2 ⇐ Counter2 + 1
    end if
  end for
end if
```

Algorithm 4 illustrates the steps that are followed to find the combinations of the critical frames that are presented by the cartesian product V_i in Section 5.1.

For more illustration of the steps in Algorithm 4, we present the following numeric example. Assume we have the inputs that are given in Table 5.7. Applying the steps in Algorithm 4 leads to the parameters in Table 5.8; which leads to the needed combinations.

Algorithm 4 Finding Combinations of Critical Frames

Inputs: Counter_Cartesian, Task_Level, Critical_Indices.
Outputs: Array Locations_Sync_Release.

Require: $Task\_Level \neq 1$
  if $Task\_Level = 2$ **then**
    $Locations\_Sync\_Release \Leftarrow Critical\_Indices(1)$
  else
    $Multiple : array\ of\ size\ Task\_Level - 2 \Leftarrow 1$
    for i = Task_Level-2 to 1 **do**
      $Multiple(i) \Leftarrow Multiple(i+1)\ .\ Size\_Criticals(i+1)$
    end for
    $Locations\_Sync\_Release(Task\_Level - 1) \Leftarrow$
    $Critical\_Indices(Counter\_Cartesian\ mod\ Size\_Criticals(Task\_Level - 1))$
    for j = Task_Level-2 to 1 **do**
      $Locations\_Sync\_Release(j) \Leftarrow$
      $Critical\_Indices(\lfloor \frac{Counter\_Cartesian}{Multiple(j)} \rfloor\ mod\ Size\_Criticals(j))$
    end for
  end if

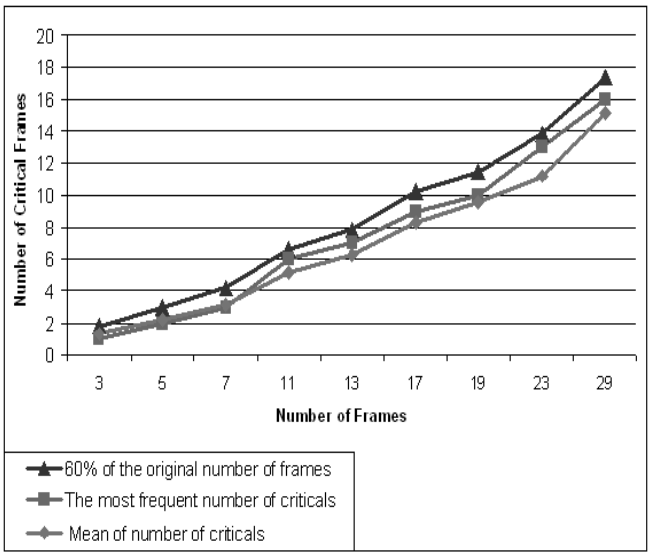
Task	*Size_Criticals*	*Critical_Indices*	*Multiple*
τ_1	3	$(0,1,2)$	2
τ_2	2	$(0,1)$	-
τ_3	1	(0)	-

Table 5.7: Numeric Example to Illustrate Algorithm 4

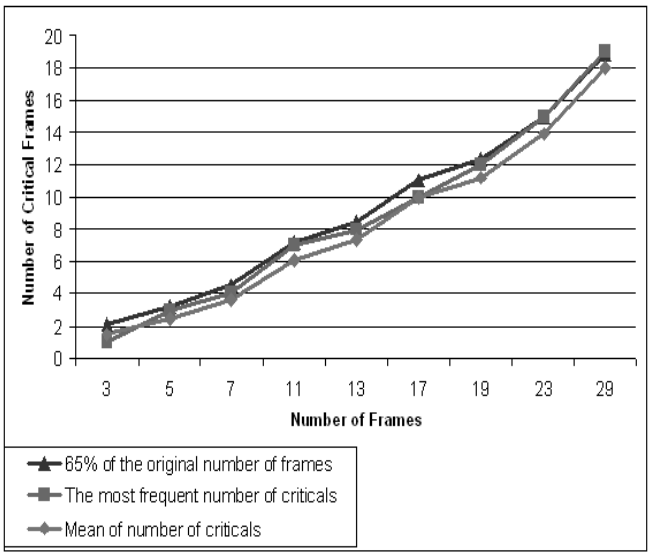
Counter	first element	$Task\_Level - 1$ element	*Locations_Sync_Release*
0	$Critical\_Index(\lfloor \frac{0}{2} \rfloor\ mod\ 3) = 0$	$Critical\_Index(0\ mod\ 2) = 0$	$(0,0)$
1	0	1	$(0,1)$
2	1	0	$(1,0)$
3	1	1	$(1,1)$
4	2	0	$(2,0)$
5	2	1	$(2,1)$

Table 5.8: Values of the Parameter: Locations_Sync_Release

5.4.4 Results of the Experiments



([1, 10])



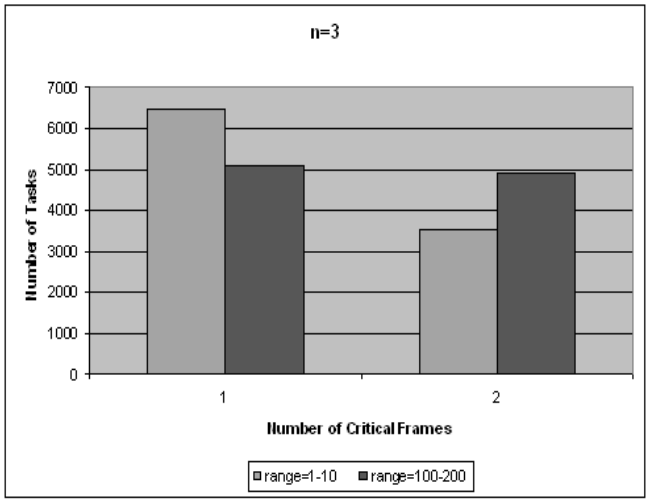
([100, 200])

Figure 5.1: Mean and Most Frequent Number of Critical Frames When the Range of Execution Times is [1,10] and [100,200]

Figure 5.1 presents the evaluation of the number of critical frames over 10000 MF tasks with randomly generated execution time values. This evaluation is presented by

the two functions: the mean number of critical frames and the most frequent number of critical frames over the 10000 randomly chosen multiframe tasks. Figure 5.1 shows that both functions of the mean and the most frequent number of critical frames are less than 60% of the original number of frames for the range of the execution times $[1,10]$; whilst these functions are less than 65% of the original number of frames for the range of the execution times $[100,200]$. This implies that the number of critical frames in practice is likely to be significantly less than $(n-1)$. In addition, Figure 5.1 demonstrates a linear relationship between the number of frames and the number of critical frames; which implies that these conclusions can be extrapolated to tasks with more than 29 frames.

To give a better coverage of the generated data, Figures 5.2 - 5.6 give details of where each point in Figure 5.1 comes from. In other words, Figures 5.2 - 5.6 show the distribution of the number of critical frames, over the 10000 randomly chosen multiframe tasks for each of n where n represents the number of frames and has one of the values $\{3,5,7,11,13,17,19,23,29\}$. We can see from all mentioned figures



(**3 frames**)

Figure 5.2: Number of Schedulable Tasks Versus Number of Critical Frames When $n = 3$ (10000 Tasks in Total)

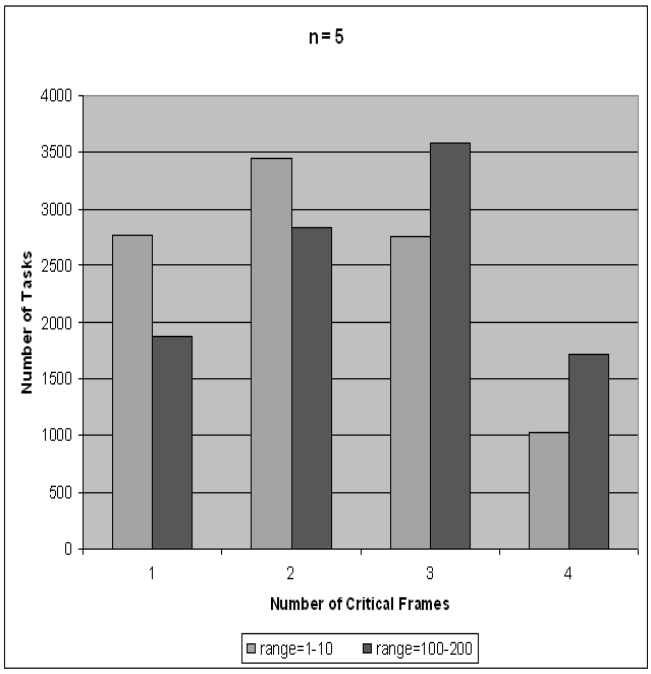
that reaching the peak when the range of execution times is $[1,10]$ is faster than when the range is $[100,200]$. To illustrate more, we can deduce from the above figures that

the bigger ratio the execution times have the less number of critical frames they could contain.

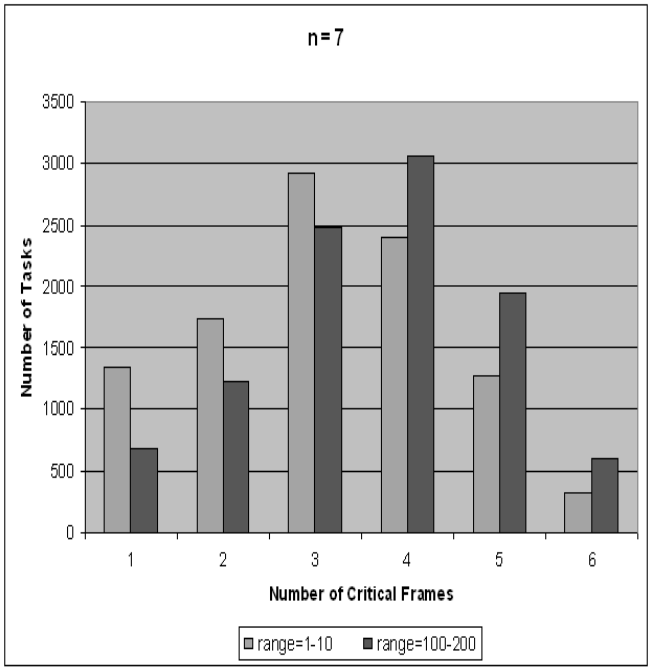
Although we can see a similar behaviour for all graphs in Figures 5.2 - 5.6, an example is given here to support and illustrate the previous mentioned deduction. Figure 5.6 (29 frames) shows that when the range of the execution times is $[1,10]$ (i.e. ratio is 10), the maximum number of tasks (i.e. about 1150 out of the 10000) have 16 critical frames while about 800 tasks have the same number of critical frames when range of the execution times is $[100,200]$ (i.e. ratio is 2). On the other hand, for the same generated set of data, when the ratio is 2, similar number of tasks (i.e. nearly 1150, 1200 and 1150 tasks out of 10000 tasks) have the number of critical frames equal $18,19,20$ critical frames respectively; while about 950, 750 and 500 tasks have the same number of critical frames (i.e. $18,19,20$ respectively) when the ratio is 10; which supports the idea of the bigger ratio the execution times have, the lower number of critical frames the MF task could get.

5.5 Summary

This chapter is concerned with the basic response time analysis of MF tasks when the AM restriction is relaxed. The analysis is done in two main steps and then evaluated. Firstly we introduce the critical frame concept, secondly we use this concept to give the basic response time formula of non-AM multiframe tasks. Evaluating the critical frame concept is also given to show how this concept improves the response time analysis by reducing the number of combinations that need to be examined to determine the worst case response time of a MF task. Although we proved that number of critical frames could reach in the worst case to $n-1$ where n is the original number of frames of the MF task, evaluation shows that number of critical frames are mostly less than 65% of the original number of the frames.

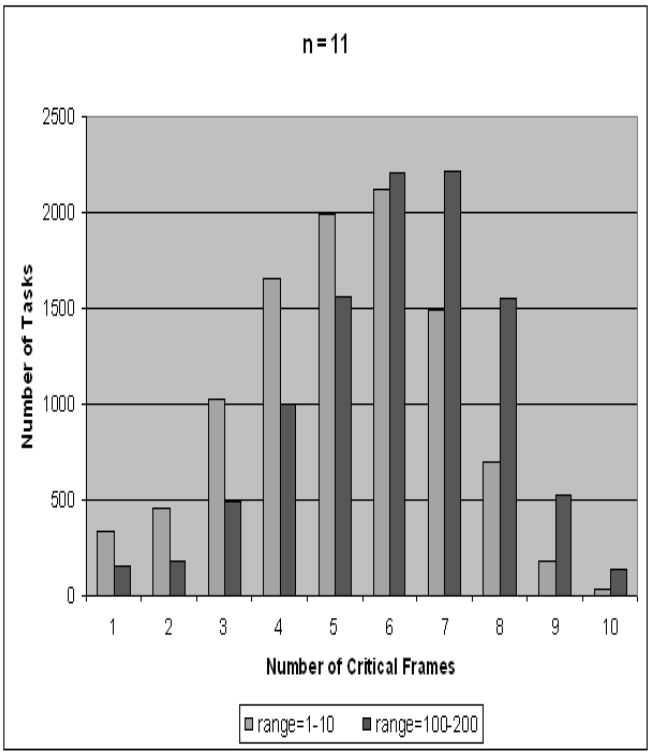


(**5 frames**)

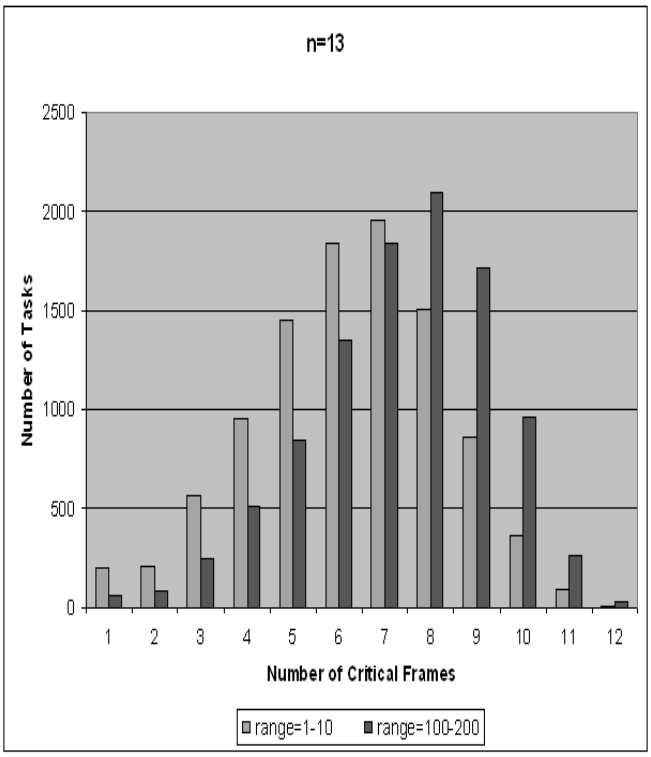


(**7 frames**)

Figure 5.3: Number of Schedulable Tasks Versus Number of Critical Frames When $n = 5$ *and* 7 (10000 Tasks in Total)

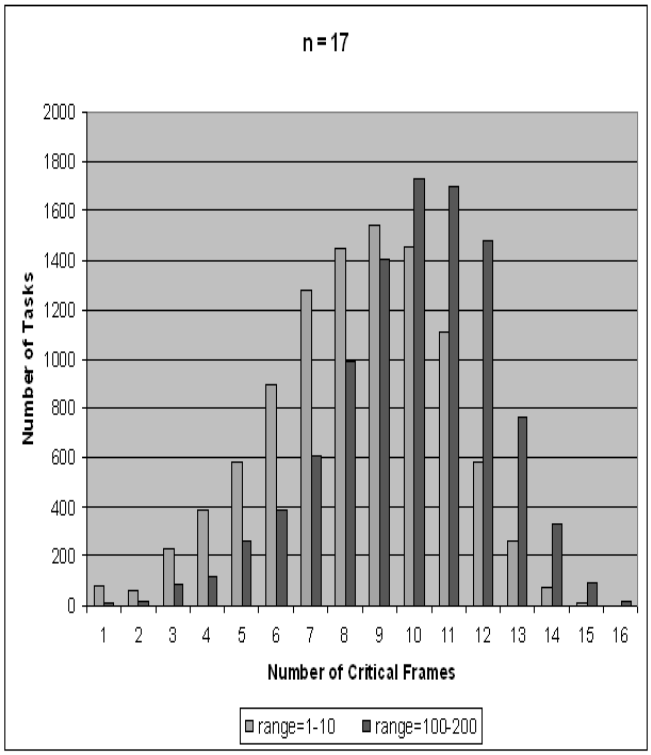


(**11 frames**)

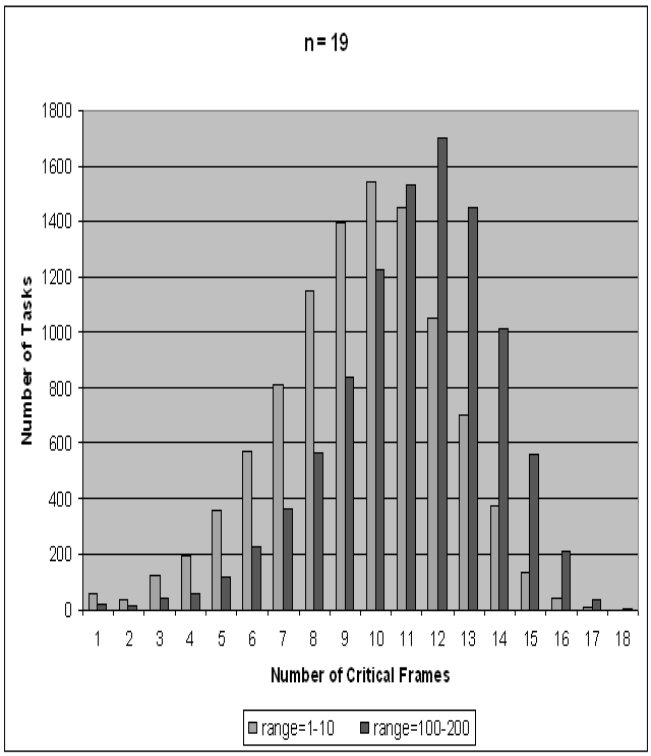


(**13 frames**)

Figure 5.4: Number of Schedulable Tasks Versus Number of Critical Frames When $n = 11$ *and* 13 (10000 Tasks in Total)

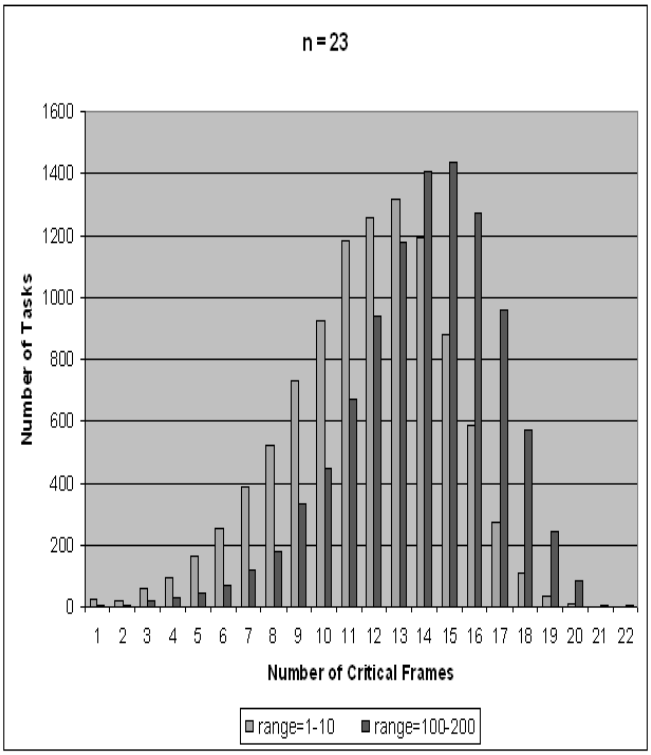


(17 frames)

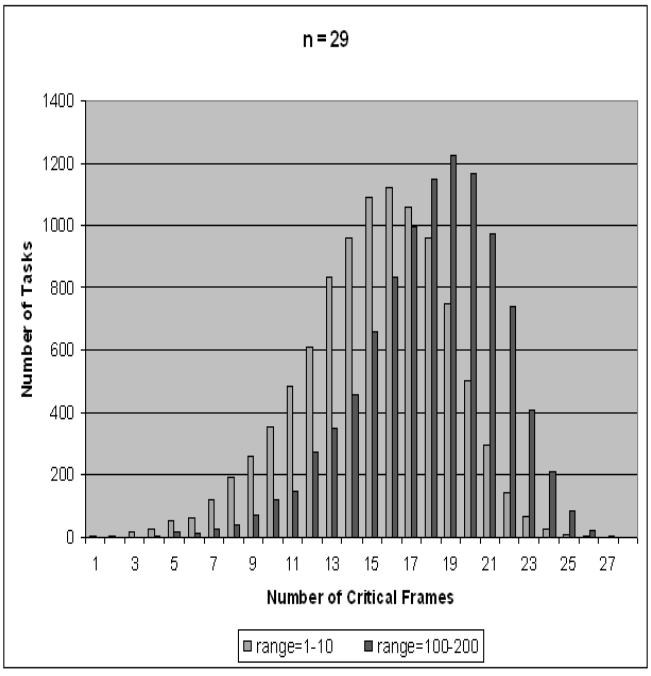


(19 frames)

Figure 5.5: Number of Schedulable Tasks Versus Number of Critical Frames When $n = 17$ *and* 19(10000 Tasks in Total)



(**23 frames**)



(**29 frames**)

Figure 5.6: Number of Schedulable Tasks Versus Number of Critical Frames When $n = 23$ *and* 29 (10000 Tasks in Total)

6 Extension of the Exact Scheduling Analysis of Non-AM Multiframe Tasks

This chapter[1] extends the system model that was given in Chapter 5 and presents the worst case response time analysis of each relative extended model. This extension is firstly done in two directions relative to release jitter and arbitrary deadlines. In the first direction we assume that each MF task τ_j has a maximum release jitter J_j but no interference from the analysed MF task itself is allowed. In the second direction, the analysed MF task, τ_i, is permitted to have a deadline greater than its period so τ_i could have interference from previous frames during its execution. Then, the two models of release jitter and arbitrary deadline are combined and the relative exact response time analysis is presented.

This chapter is organised as follows: the next section presents the worst case response time analysis of MF tasks that are subjected to release jitter. The worst case response time analysis of MF tasks whose deadlines are arbitrary is presented in Section 6.2. Section 6.3 presents an example to illustrate the analysis of the arbitrary deadlines senario. Section 6.4 presents the worst case response time analysis of MF tasks whose deadlines are arbitrary and are subjected to release jitter. In Section 6.5 we present an example to illustrate the analysis of the combined model of release jitter and arbitrary deadlines .

[1] Parts of Sections 6.1 and 6.2 in this chapter are published in [79].

6.1 Analysis of MF Tasks with Release Jitter

Section 4.1 in Chapter 4 explained how release jitter affects the periodicity of the tasks. However, we presented in Chapter 5 the worst case response time analysis of purely periodic non-AM multiframe tasks. In this section, we cover the extension of this analysis when non-AM multiframe tasks are subjected to release jitter. The analysis is presented in a self contained manner rather than as an extension to the AM analysis of Section 4.1.

When a task τ_j is subjected to release jitter, its release time takes place somewhere after its arrival time in an interval of length equals to the maximum release jitter, J_j. To symbolise the release jitter problem mathematically, let a_j^k and s_j^k be the times when the $(k+1)^{th}$ frame[2] of τ_j arrives and is released respectively. a_j^k and s_j^k must satisfy Equations (6.1) and (6.2) as τ_j arrives periodically and has to be released after its arrival time within a maximum interval of time equals to J_j.

$$a_j^k = x + kT_j; \tag{6.1}$$

$$a_j^k \leq s_j^k \leq y + kT_j; \tag{6.2}$$

$$y - x = J_j \text{ and } k = 0, 1, 2, ..$$

As release jitter affects the periodicity of the release times of τ_j, the worst case situation of a lower priority task τ_i is when τ_j is released the most during τ_i's execution because, in this case, τ_j provides the maximum number of interference within τ_i's execution. However, the maximum number of releases that τ_j practices is when its release times are close to each other as much as possible. The following lemma explains this situation.

Lemma 2 *Having τ_j subjected to release jitter, J_j, τ_j is released the most when its first frame is released rightmost in its release jitter interval while subsequent frames are released leftmost in this release jitter interval.*

Proof

[2] Although $k = 0, 1, ..$, for simplicity of the presentation we say $(k+1)^{th}$.

Substitute Equation (6.1) for Equation (6.2), so we get

$$x + kT_j \leq s_j^k.$$

Now, substitute $k+1$ for k to get the following inequality

$$x + (k+1)T_j \leq s_j^{k+1}.$$

Subtract s_j^k from both sides of this inequality, so

$$x + (k+1)T_j - s_j^k \leq s_j^{k+1} - s_j^k.$$

We already know, from Equation (6.2), that $-s_j^k \geq -(y + kT_j)$. So,

$$x + (k+1)T_j - (y + kT_j) \leq x + (k+1)T_j - s_j^k \leq s_j^{k+1} - s_j^k.$$

$$T_j - (y - x) \leq s_j^{k+1} - s_j^k.$$

Therefore,

$$s_j^{k+1} \geq s_j^k + T_j - J_j. \tag{6.3}$$

Equation (6.3) presents a relationship between release times of each two successive frames when τ_j is subjected to release jitter. Without lose of generality and to clasp the first two releases of τ_j's frames, we assume τ_j first arrives as early as possible (i.e. first arrival time is x) but is released as late as possible so release time of the first frame (i.e. $k = 0$) is $s_j^0 = y$ which is rightmost of τ_j's release jitter interval.

For $k > 0$, τ_j is released the most when release times of its successive frames are closest to each other. In other words, τ_j is released the most when s_j^{k+1} equals to the actual lower bound value of s_j^{k+1}. We already know that $s_j^k \geq a_j^k$ and $a_j^k = x + kT_j$, so Equation (6.3) becomes

$$s_j^{k+1} \geq x + kT_j + T_j - J_j,$$

$$s_j^{k+1} \geq x + (k+1)T_j - J_j.$$

As $a_j^{k+1} = x + (k+1)T_j$,

$$s_j^{k+1} \geq a_j^{k+1} - J_j.$$

However, $s_j^{k+1} \geq a_j^{k+1}$ as the release time of a task is always after its arrival time. So, s_j^{k+1} should not take values in the range $[a_j^{k+1} - J_j, a_j^{k+1})$. Therefore, the lowest value that s_j^{k+1} takes is a_j^{k+1}; which means that the latest time τ_j is released after the first release is as soon as it arrives. However, the arrival times are always at the beginning of release jitter interval; which is leftmost of release jitter interval. So τ_j is released the most when the first frame is released rightmost of its release jitter interval while next frames are released leftmost of its release jitter interval.□

Figure 6.1 illustrates the situation where τ_j is released the most in the interval $[s_i^1, f]$.

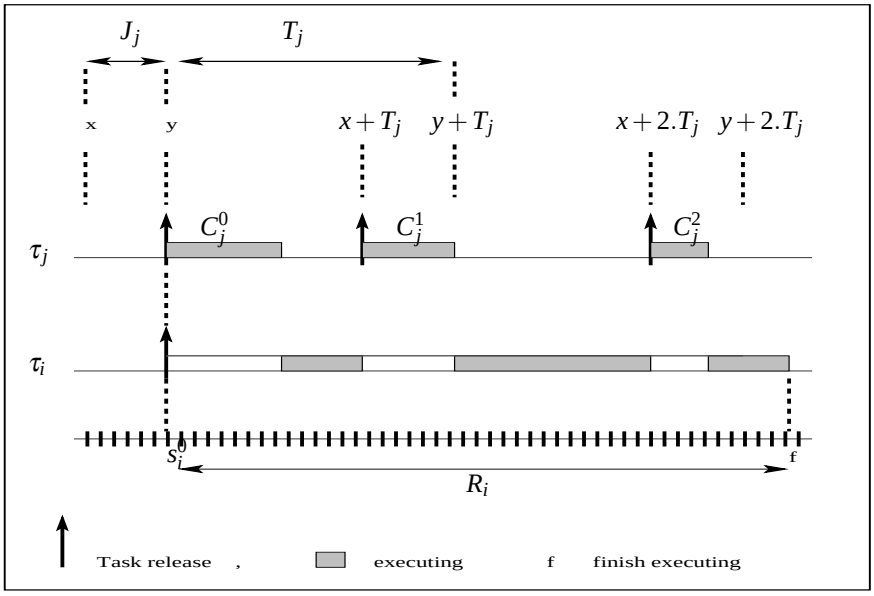


Figure 6.1: Illustration of Release Jitter Problem

Lemma 2 presents the situation where τ_j is invoked for the maximum number of times. As τ_j is a MF task, the amount of invocation (i.e. interference on lower priority tasks) that is relative to this maximum number is relatively different according to the first released frame of τ_j. However, we explained in Section 5.1 how critical frames of a MF task are the only frames that provide the maximum amount of interference in lower priority tasks. The following lemma proves that this critical frame set remains the same when τ_j is subjected to release jitter.

Lemma 3 *Having non-AM multiframe task τ_j subjected to release jitter, its critical frame set remains the same as when τ_j does not have release jitter.*

Proof

The maximum amount of interference from a MF task τ_j, for all number of interference (i.e. $\forall k = 1, 2, ..$), are generated by its critical frames. On the other hand, release jitter of τ_j could affect the number of interference τ_j generates on a lower priority task but does not affect the execution times of the frames. So, the frame that is relative to the maximum interference on this lower priority task could be different from the one that is relative to the maximum interference when τ_j was not subjected to release jitter. However, both frames are from the critical frame set because this set depends on the maximum amount of interference that τ_j generates for all its possible number of interference. So, even if the number of interference increases by release jitter, the relative critical frame will be one of the original critical frame set. Therefore, the critical frame set keeps the same as explained in Section 5.1.□

Example

The following example illustrate Lemma 3. Suppose a system with three MF tasks in Table 6.1, the critical frame locations of τ_1 and τ_2 are $\{1, 2, 3, 4\}$ and $\{1, 2, 3\}$ respectively.

task	*C*	*T*	priority
τ_1	3, 4, 6, 7, 8, 6, 8	10	high
τ_2	5, 6, 7, 10	40	medium
τ_3	1, 2, 3	60	low

Table 6.1: Example System

Firstly, we present all possible response times of τ_3 assuming there is no release jitter for any tasks in the system. Table 6.2 presents all possible response times that are relative to all critical frames of τ_1 and τ_2. So, we see from the table that the worst case response time of τ_3 is 50 and the relative critical frames of τ_1 and τ_2 are the execution times whose locations are 3 and 3 respectively; which represent the execution times 7 and 10 respectively.

Now, assume τ_1 has release jitter of 1 then this release jitter gives rise to an extra interference from τ_1 and therefore its critical frame is changed to be relative to the new number of interference. To find out how the critical frame is changed, Table 6.3

frame location	0	1	2	**3**	4	5	6
0	-	-	-	-	-	-	-
1	-	19	30	30	34	-	-
2	-	20	37	39	35	-	-
3	-	30	40	**50**	38	-	-

Table 6.2: Responses of τ_3 When No Release Jitter

presents all possible response times of τ_3 (calculated[3] by applying Equation (6.5)) that are relative to all critical frames of τ_1 and τ_2. We see from this table that the worst case response time of τ_3 is now 56 and critical frames of τ_1 and τ_2 are 6 and 10 respectively. At the same time, according to the frames of 7 and 10 of τ_1 and τ_2, the relative response time of τ_3 in the case $J_1 = 1$ is $R_3 = 54$. So, the specific critical frame of τ_1 has changed when release jitter exists but is still one member of the critical frame set.

frame location	0	1	**2**	3	4	5	6
0	-	-	-	-	-	-	-
1	-	19	36	38	34	-	-
2	-	27	37	39	35	-	-
3	-	38	**56**	54	38	-	-

Table 6.3: Responses of τ_3 When $J_1 = 1$

As a first step in any worst case response time analysis of a task τ_i we have to identify its critical instance that is considered as the worst case situation in τ_i's response time analysis. For preemptive real-time tasks under fixed priority scheduling, τ_i's response time is the worst when τ_i is preempted the most during its execution. A higher priority MF task τ_j preempts τ_i the most when τ_j provides as much interference as possible in τ_i's execution. Having τ_j and τ_i subjected to release jitter and considering Lemma 2, τ_j interferes τ_i the most when both τ_j and τ_i start their first executions simultaneously and τ_j's first frame is released rightmost in its release jitter interval while subsequent frames are released leftmost in this release jitter interval. On the other hand, as τ_j is a non-AM multiframe task, its critical frames provide the maxi-

[3] Full details of the calculation are given by the example at the end of this section.

mum amount of interference in lower priority tasks. Also, as τ_i does not preempt itself, studying the schedulability status of τ_i's peak frame is enough to decide its schedulability status. Therefore, we define the critical instance of a non-AM multiframe task τ_i as in the following definition.

Definition 8 Having a system whose MF tasks are subjected to release jitter, the critical instance of a non-AM multiframe task τ_i *is the simultaneous release of its peak frame and the critical frames, of the higher priority MF tasks, that lead to the worst case response time of* τ_i*; where the simultaneous release takes its place at the end (i.e. rightmost) of their release jitter intervals whilst subsequent releases of the frames that follow the critical frames take their place leftmost in their relative release jitter intervals.*

Figure 6.1 illustrates this critical instance by presenting the simultaneous release of two MF tasks; a higher priority MF task τ_j and a lower priority MF task τ_i.

Assuming Definition 8 of the critical instance and according to Lemma 3, analysis of the worst case response time of τ_i has to be maximised over all combinations of the critical frames of the higher priority MF tasks. However, assume $R_{i,\tilde{v}}$ is τ_i's response time that is relative to a specific combination, that is represented[4] by $\tilde{v}$, of the critical frames of the higher priority MF tasks; to find $R_{i,\tilde{v}}$ we have to find the amount of interference from the higher priority MF tasks within $R_{i,\tilde{v}}$. The following lemma proves a formula for finding this amount of interference.

Lemma 4 *Given a real-time system consisting of N non-AM multiframe tasks* τ_j*;* $j = 1,2,..,N$*, each MF task* τ_j *has a maximum release jitter equals* J_j*. Assuming Definition 8 where the simultaneous release of the critical frames of the higher priority MF tasks is represented by* $\tilde{v}$*,* $R_{i,\tilde{v}}$ *is the response time of* τ_i *that is relative to a specific* $\tilde{v}$*; the maximum amount of interference in* $R_{i,\tilde{v}}$ *from the tasks whose priorities are higher than* τ_i *is given by Equation (6.4).*

$$\sum_{j=1}^{i-1} \xi_j^{\tilde{v}_j}(\lceil \frac{R_{i,\tilde{v}} + J_j}{T_j} \rceil). \tag{6.4}$$

[4] $\tilde{v} \in \hat{V}_i$; where $\hat{V}_i$ is given in Section 5.1 as the cartesian product of the locations of the critical frames of τ_j, more details in Section 5.1.

Proof

Assume $\tilde{I}$ is the maximum amount of interference from tasks whose priorities are higher than τ_i, then we can present $\tilde{I}$ as a summation of all interference from higher priority tasks because of the simultaneous release of all higher priority MF tasks (Definition 8). So,

$$\tilde{I} = \sum_{j=1}^{i-1} \tilde{I}_i;$$

where $\tilde{I}_j$ is the maximum amount of interference from the higher priority MF task τ_j. According to Definition 8 we divide this amount into two parts

$$\tilde{I}_j = C_j^{\tilde{v}_j} + I_j^{rest};$$

where $C_j^{\tilde{v}_j}$ is the first interference that τ_j provides within $(T_j - J_j)$ while I_j^{rest} is the amount of interference that τ_j provides within $R_{i,\tilde{v}} - (T_j - J_j)$ starting from the release that follows the first one. So, I_j^{rest} is given by:

$$I_j^{rest} = \xi_j^{(\tilde{v}_j+1)}(\lceil \frac{R_{i,\tilde{v}} - (T_j - J_j)}{T_j} \rceil).$$

Therefore,

$\tilde{I}_j = C_j^{\tilde{v}_j} + \xi_j^{(\tilde{v}_j+1)}(\lceil \frac{R_{i,\tilde{v}} - (T_j - J_j)}{T_j} \rceil)$

$\tilde{I}_j = \xi_j^{\tilde{v}_j}(\lceil \frac{R_{i,\tilde{v}} - (T_j - J_j)}{T_j} \rceil + 1)$ because the cumulative function starts from a previous release so an extra interference has been added,

$\tilde{I}_j = \xi_j^{\tilde{v}_j}(\lceil \frac{R_{i,\tilde{v}} - (T_j - J_j)}{T_j} + 1 \rceil)$ because we add an integer to the ceiling function so we can move this integer into the ceiling function,

$\tilde{I}_j = \xi_j^{\tilde{v}_j}(\lceil \frac{R_{i,\tilde{v}} - (T_j - J_j)}{T_j} + \frac{T_j}{T_j} \rceil) = \xi_j^{\tilde{v}_j}(\lceil \frac{R_{i,\tilde{v}} + J_j}{T_j} \rceil)$.

So, the maximum amount of interference from tasks whose priorities are higher than the MF task τ_i, assuming $C_j^{\tilde{v}_j}$ is the first execution time of τ_j (remember that $\tilde{v}_j$ is a location of a critical frame, of τ_j, that is relative to the combination $\tilde{v}$), is given by Equation (6.4).□

Having a formula for the amount of interference from higher priority MF tasks, what is left in the response time analysis is to find the formula that represents $R_{i,\tilde{v}}$ for

a specific combination, $\tilde{v} \in \hat{V}_i$, of the critical frames of all higher priority MF tasks. The following theorem proves a formula of $R_{i,\tilde{v}}$.

Theorem 7 *Given a real-time system consisting of N non-AM multiframe tasks τ_j; $j = 1,2,..,N$, each MF task τ_j has a maximum release jitter equals J_j and has its peak frame at position m_j; assuming Definition 8, the worst case response time of τ_i, that is relative to $\tilde{v} \in \hat{V}_i$ is given by the smallest non-negative solution to Equation (6.5) assuming priority ceiling protocols [66, 60]. $\tilde{v}$ represents a combination vector of the critical frame locations of the MF tasks whose priorities are higher than τ_i.*

$$R_{i,\tilde{v}} = C_i^{m_i} + B_i + \sum_{j=1}^{i-1} \xi_j^{\tilde{v_j}} \left(\lceil \frac{R_{i,\tilde{v}} + J_j}{T_j} \rceil\right) \tag{6.5}$$

where $\tilde{v_j}$ is the j^{th} element of the vector $\tilde{v}$.

Proof

As τ_i does not preempt itself, the maximum amount of τ_i's execution is represented by its peak frame. Also, priority ceiling protocols let the task to be blocked at most once, so we just add the maximum blocking time to the response time formula. In addition, Lemma 4 presents a formula for the amount of interference from higher priority MF tasks. So as we are assuming the simultaneous release of τ_i and higher priority MF tasks, we can present the worst case response time of τ_i as a summation of its execution and interference from the higher priority MF tasks. Therefore, the worst case response time of the task τ_i is given by the smallest non negative solution to Equation (6.5).□

Solving Equation (6.5) is given by forming a recurrence equation given by Equation (6.6).

$$R_{i,\tilde{v}}^{l+1} = C_i^{m_i} + B_i + \sum_{j=1}^{i-1} \xi_j^{\tilde{v_j}} \left(\lceil \frac{R_{i,\tilde{v}}^{l} + J_j}{T_j} \rceil\right) \tag{6.6}$$

where $R_{i,\tilde{v}}^0 = C_i^{m_i}$ and $l = 0,1,..$ till $R_{i,\tilde{v}}^{l+1} = R_{i,\tilde{v}}^l = R_{i,\tilde{v}}$. However, if $R_{i,\tilde{v}}^{l+1} > D_i - J_i$, we say that τ_i is not schedulable.

Corollary 2 *The worst case response time of a non-AM multiframe task τ_i in a system*

subjected to release jitter is given by Equation (6.7)

$$R_i = \max_{\tilde{v} \in \hat{V}_i} \{R_{i,\tilde{v}}\} \tag{6.7}$$

Proof

For each combination of $\tilde{v} \in \hat{V}_i$, we find the worst case response time of τ_i that is relative to this combination $\tilde{v}$. Therefore, the worst case response time of τ_i is the maximum of all of them as in Equation (6.7).□

Schedulability Test

As the response time is calculated from the time when τ_i is released while deadline is scheduled from when τ_i arrives in the system, the response time scheduling test is given as following: τ_i is schedulable if $R_i \leq D_i - J_i$.

Example

Recall the example in Table 6.1 with no blockings and $J_1 = 1, J_2 = 0,$ *and* $J_3 = 0$, to find the worst case response time of τ_3 we apply Equation (6.6) for all $\tilde{v} \in \hat{V_3}$; where $\hat{V_3} = \{(1,1),(2,1),(3,1),(4,1),(1,2),(2,2),(3,2),(4,2),(1,3),(2,3),(3,3),(4,3)\}$. For example, when $\tilde{v} = (2,3)$ then

$R^{l+1}_{3,(2,3)} = C_3^{m_3} + \sum_{j=1}^{2} \xi_j^{\tilde{v}_j} (\lceil \frac{R^l_{3,(2,3)} + J_j}{T_j} \rceil)$

$R^{l+1}_{3,(2,3)} = 3 + \xi_1^2(\lceil \frac{R^l_{3,(2,3)}+1}{T_1} \rceil) + \xi_2^3(\lceil \frac{R^l_{3,(2,3)}}{T_2} \rceil)$

l=0, $R^0_{3,(2,3)} = 3$;

l=1, $R^1_{3,(2,3)} = 3 + \xi_1^2(1) + \xi_2^3(1) = 3 + 6 + 10 = 19.$

l=2, $R^2_{3,(2,3)} = 3 + \xi_1^2(\lceil \frac{19+1}{10} \rceil) + \xi_2^3(\lceil \frac{19}{40} \rceil) = 3 + 13 + 10 = 26.$

l=3, $R^3_{3,(2,3)} = 3 + \xi_1^2(\lceil \frac{26+1}{10} \rceil) + \xi_2^3(\lceil \frac{26}{40} \rceil) = 3 + 21 + 10 = 34.$

l=4, $R^4_{3,(2,3)} = 3 + \xi_1^2(\lceil \frac{34+1}{10} \rceil) + \xi_2^3(\lceil \frac{34}{40} \rceil) = 3 + 27 + 10 = 40.$

l=5, $R^5_{3,(2,3)} = 3 + 35 + 10 = 48.$

l=6, $R^6_{3,(2,3)} = 3 + 35 + 15 = 53.$

l=7, $R^7_{3,(2,3)} = 3 + 38 + 15 = 56.$

l=8, $R^8_{3,(2,3)} = 3 + 38 + 15 = 56.$

So, $R_{3,(2,3)} = 56$. Similarly, we find all $R_{3,\tilde{v}}$ for all $\tilde{v} \in \hat{V_3}$ to get values that were previously presented by Table 6.3. Therefore, R_3 is the maximum of all values in Table 6.3, so $R_3 = \max\{19,36,38,34,27,37,39,35,38,56,54,38\} = 56 \leq D_3$. So, τ_3 is schedulable.

6.2 Analysis of MF Tasks with Arbitrary Deadlines

The previous section presents an extension of the worst case response time analysis of the non-AM multiframe tasks from the point of view that tasks have release jitter but the analysed task does not have interference from its previous frames during its execution. This section presents another extension of the worst case response time analysis of the non-AM multiframe tasks; where the MF tasks have arbitrary deadlines but does not have release jitter. In other words, the deadline of the analysed task could be greater than its period, so analysis of the worst case response time has to take into account the interference from the analysed MF task itself as well as interference from higher priority MF tasks.

As a first issue we start by identifying the situation of the MF task τ_i that could lead to its worst case delay of its response time; which we call the critical instance of τ_i. As there is a possibility of having interference from the task itself within its execution as well as the interference from the higher priority MF tasks, to demonstrate the maximum amount of interference from τ_i, we have to consider its own critical frames besides the critical frames of the higher priority MF tasks. So, the arbitrary deadline scenario leads us to the situation of analysing all critical frames of the analysed MF task instead of analysing only its peak frame because its critical frames are the frames that generate the maximum amount of interference within the same or lower priority tasks. In other words, the critical instance of τ_i is presented by the following definition.

Definition 9 . The critical instance of a non-AM multiframe task τ_i whose deadline is arbitrary *is the simultaneous release, that leads to the worst case response time of τ_i, of the critical frames of both τ_i and all MF tasks whose priorities are higher than τ_i's.*

In the previous section, the simultaneous releases of the critical frames of the MF tasks whose priorities are higher than τ_i are represented by the cartesian product $\hat{V}_i$ of $\hat{L_j}$; where $j = 1, 2, .., i-1$. However, Definition 9 considers all simultaneous releases of the critical frames of the analysed MF task τ_i and the MF tasks whose priorities are higher than τ_i's. So, we represent the simultaneous releases in this section by the

cartesian product $\hat{V_i}$ of $\hat{L_j}$; where $j = 1,\ 2,\ ..,\ i$. Therefore, the response time of τ_i has to be analysed for all its critical frames whose locations are presented by $\tilde{v_i}$, which is the i^{th} element of the vector $\tilde{v} \in \hat{V_i}$, as well as critical frames of higher priority MF tasks, whose locations are presented by $\tilde{v_j};\ j = 1,2,..,i-1$.

To analyze the response time of τ_i that is relative to the combination of the critical frames $\tilde{v}$, the first step is to define the busy period of a frame of a MF task as the time from when this frame is released until it finishes its execution that is relative to this frame. So, assuming Definition 9, the worst case busy period of τ_i that is relative to the combination $\tilde{v}$ is the maximum busy period of τ_i taking into account that the busy period could include interference from τ_i itself.

Assume q is the number of invocations of τ_i $(q = 1,2,..)$, to find the worst case busy period of the q^{th} frame[5] of τ_i that is relative to the combination $\tilde{v}$ we follow two steps: first we find $r_{i,\tilde{v}}(q)$ that represents the time from when τ_i's critical frame whose location is $\tilde{v_i}$ is released until the q^{th} frame has finished its execution; then we find $w_{i,\tilde{v}}(q)$ that represents the q^{th} busy period of τ_i that is relative to the combination of the critical frames $\tilde{v}$ by subtracting the overlap invocations that are not related to the busy period of the q^{th} frame. The following theorem proves the technique that is used to find $w_{i,\tilde{v}}(q)$.

Theorem 8 *Having a system of non-AM multiframe tasks, each MF task τ_i has an arbitrary deadline D_i. Assuming Definition 9, the q^{th} busy period of τ_i that is relative to the combination $\tilde{v}$ (i.e. $w_{i,\tilde{v}}(q)$) is given by Equation (6.8) assuming the priority ceiling protocols [66, 60].*

$$\begin{aligned} w_{i,\tilde{v}}(q) &= r_{i,\tilde{v}}(q); && for\ q = 1, \\ &= r_{i,\tilde{v}}(q) - (q-1)T_i; && for\ q > 1. \end{aligned} \tag{6.8}$$

where $r_{i,\tilde{v}}(q)$ is found by the smallest non-negative solution to Equation (6.9).

$$r_{i,\tilde{v}}(q) = \xi_i^{\tilde{v_i}}(q) + B_i + \sum_{j=1}^{i-1} \xi_j^{\tilde{v_j}}(\lceil \frac{r_{i,\tilde{v}}(q)}{T_j} \rceil). \tag{6.9}$$

[5] With the knowledge that q's values start with $1,2,..$

where $\xi_i^{\tilde{v}_i}(q)$ is introduced by Definition 1 and B_i is the maximum blocking time of τ_i.

Proof

As we are assuming the simultaneous release of τ_i and higher priority MF tasks, $r_{i,\tilde{v}}(q)$ can be represented by a summation of two kinds of execution; one is related to the execution of τ_i and the other is related to MF tasks other than τ_i. The execution that is related to τ_i is represented by its cumulative function $\xi_i^{\tilde{v}_i}(q)$ and the execution that is related to the MF tasks other than τ_i is represented by blocking from lower priority tasks and interference from higher priority tasks.

As priority ceiling protocols allow the task to be blocked at most once during its execution and as $r_{i,\tilde{v}}(q)$ is a continuous execution of the same priority MF task, the blocking term from lower priority tasks is represented by the maximum expected blocking time B_i. Furthermore, as we assume the simultaneous release of τ_i and higher priority tasks (Definition 9 of the critical instance of τ_i), the interference from the MF tasks that have higher priority than τ_i is presented by a summation of all interference from those tasks.

Assume $\tilde{I}_j$ is the interference from a higher priority MF multiframe task τ_j in $r_{i,\tilde{v}}(q)$, applying Lemma 1 leads to $\tilde{I}_j$ being presented by $\xi_j^{\tilde{v}_j}(\lceil \frac{r_{i,\tilde{v}}(q)}{T_j} \rceil$ (with the assumption that $J_j = 0$). So, the maximum interference from the MF tasks whose priorities are higher than τ_i's is presented by $\sum_{j=1}^{i-1} \xi_j^{\tilde{v}_j}(\lceil \frac{r_{i,\tilde{v}}(q)}{T_j} \rceil)$. Therefore, $r_{i,\tilde{v}}(q)$ is a collection of $\xi_i^{\tilde{v}_i}(q)$, B_i and $\sum_{j=1}^{i-1} \xi_j^{\tilde{v}_j}(\lceil \frac{r_{i,\tilde{v}}(q)}{T_j} \rceil)$; which is identical to Equation (6.9).

$r_{i,\tilde{v}}(q)$ consists of q number of τ_i's execution starting from τ_i's critical frame whose location is $\tilde{v}_i$. So, the first busy period of τ_i is the busy period of the $\tilde{v}_i{}^{th}$ critical frame of τ_i. In addition, $w_{i,\tilde{v}}(q)$ starts from when the q^{th} frame of τ_i is released whilst $r_{i,\tilde{v}}(q)$ starts from when the first frame is released; also, both of $w_{i,\tilde{v}}(q)$ and $r_{i,\tilde{v}}(q)$ have the same end. So, when $q = 1$ both of $w_{i,\tilde{v}}(1)$ and $r_{i,\tilde{v}}(1)$ have the same start and end; which means that $w_{i,\tilde{v}}(1) = r_{i,\tilde{v}}(1)$. However, when $q > 1$, $w_{i,\tilde{v}}(q)$ and $r_{i,\tilde{v}}(q)$ have different starts where $r_{i,\tilde{v}}(q)$ starts at 0 and $w_{i,\tilde{v}}(q)$ starts at $(q-1)T_i$. So, $w_{i,\tilde{v}}(q) = w_{i,\tilde{v}}(q) - (q-1)T_i$ which is identical to Equation (6.8).□

Equation (6.9) is solved by forming a recurrence relationship as in Equation (6.10); where $l = 0, 1, ..$ until getting $r_{i,\tilde{v}}^{l+1}(q) = r_{i,\tilde{v}}^{l}(q)$. However if $r_{i,\tilde{v}}^{l+1}(q) > (q-1)T_i + D_i$

then τ_i is not schedulable as τ_i would have passed the q^{th} deadline in this case.

$$r_{i,\tilde{v}}^{l+1}(q) = \xi_i^{\tilde{v}_i}(q) + B_i + \sum_{j=1}^{i-1} \xi_j^{\tilde{v}_j}(\lceil \frac{r_{i,\tilde{v}}^{l}(q)}{T_j} \rceil). \tag{6.10}$$

Theorem 8 provides a way for finding the q^{th} busy period of τ_i that is relative to the combination $\tilde{v}$; for $q = 1, 2, \ldots$ To identify the worst busy period that is relative to the combination $\tilde{v}$, we have to maximise all relative busy periods that includes interference from τ_i. In other words, we have to maximise $w_{i,\tilde{v}}(q)$ over all q; where q takes values from 1 until τ_i stops interfering within its invocations. So, we keep increasing values of q and finding $w_{i,\tilde{v}}(q)$ until Equation (6.11) is satisfied.

$$w_{i,\tilde{v}}(q) \leq T_i. \tag{6.11}$$

That is because satisfying Equation (6.11) means that τ_i has finished its execution within the period it is released in; and no further interference from τ_i itself will occur. Therefore, the worst case busy period $w_{i,\tilde{v}}$ that is relative to the combination $\tilde{v} \in \hat{V}_i$ is the maximum busy period over all q that is bounded by Equation (6.11). Mathematically, $w_{i,\tilde{v}}$ is found by Equation (6.12).

$$w_{i,\tilde{v}} = \max_{q=1,2,..} \{w_{i,\tilde{v}}(q)\}. \tag{6.12}$$

Therefore, to find the worst case response time of τ_i, R_i, we have to maximise all worst case busy periods $w_{i,\tilde{v}}$ over all possible combinations, $\tilde{v}$. In other word, the worst case response time of τ_i is given by Equation (6.13).

$$R_i = \max_{\tilde{v} \in \hat{V}} \{w_{i,\tilde{v}}\} \tag{6.13}$$

Scheduling Test

The schedulability test of τ_i within the arbitrary deadline scenario is as follows: τ_i is schedulable if its worst case response time, that is calculated by Equation (6.13), is less than or equals to its deadline (i.e. $R_i \leq D_i$).

6.3 Example

Assume a system with three independent tasks τ_1, τ_2 and τ_3 with the parameters given in Table 6.4. For simplicity of the example we assume all blockings are 0. To identify

task	C	T	D	priority
τ_1	5, 3, 4, 6, 8, 7	10	10	1
τ_2	6, 10, 7, 5	40	40	2
τ_3	6, 7, 8	50	60	3

Table 6.4: Attributes of the Tasks in the System

the schedulability status of τ_3, we have to find its worst case response time. As $D_3 > T_3$, we need to evaluate the response time of τ_3 over all its critical frames of the MF tasks τ_1, τ_2 and τ_3.

Using analysis in Section 5.1, locations of the critical frames $\hat{L_j};\ j = 1,2,3$ are found as follows
$\hat{L_1} = \{2,3,4\}$,
$\hat{L_2} = \{0,1\}$ and
$\hat{L_3} = \{1,2\}$.
So, the cartesian product $\hat{V_3}$ of $\hat{L_j};\ j = 1,..3$ is found as
$\hat{V_3} = \{(2,0,1),(2,0,2),(2,1,1),(2,1,2),(3,0,1),(3,0,2),(3,1,1),(3,1,2),(4,0,1),$
$(4,0,2),(4,1,1),(4,1,2)\}$.
Now, we apply the response time analysis in this section in two steps. In the first step, we find $w_{i,\tilde{v}}$ using Theorem 8 and Equation (6.10) and in the second step, we find the worst case response time R_i by maximising $w_{i,\tilde{v}}$ over all $\tilde{v} \in \hat{V_3}$.

For example for the combination $\tilde{v} = (2,0,1)$, applying Theorem 8 leads to

$$r_{3,(2,0,1)}(q) = \xi_3^1(q) + \sum_{j=1}^{2} \xi_j^{\tilde{v}_j}(\lceil \frac{r_{3,(2,0,1)}(q)}{T_j} \rceil)$$

$q = 1$, then $r_{3,(2,0,1)}(1) = 7 + \sum_{j=1}^{2} \xi_j^{\tilde{v}_j}(\lceil \frac{r_{3,(2,0,1)}(1)}{T_j} \rceil)$

By solving this iterative equation, we find that $r_{3,(2,0,1)}(1) = 38$. So,

$w_{3,(2,0,1)}(1) = 38 \leq T_3$.

So restriction (6.11) is satisfied and therefore, no need to increase q's values any more. Thus, $w_{3,(2,0,1)} = 38$.
Similarly, we find $r_{3,(2,0,2)}(1) = 8 + \sum_{j=1}^{2} \xi_j^{\tilde{v}_j}(\lceil \frac{r_{3,(2,0,2)}(1)}{T_j} \rceil) = 39$. So, $w_{3,(2,0,2)}(1) = 39$. In the same way, we find all $w_{3,\tilde{v}}$ using $r_{3,\tilde{v}}(q)$ to get the results in Table 6.5. Where we calculate all possible worst case busy periods that are relative to all critical frames of τ_3 and higher priority MF tasks. Note from Table 6.5 that values of q increases to

$\tilde{v}$	q	$r_{3,\tilde{v}}(q)$	$w_{3,\tilde{v}}(q)$	$w_{3,\tilde{v}}$
$(2,0,1)$	1	38	38	38
$(2,0,2)$	1	39	39	39
$(2,1,1)$	1	57	$57 > T_3$	
$(2,1,1)$	2	69	19	$max\{57,19\} = 57$
$(2,1,2)$	1	58	$58 > T_3$	
$(2,1,2)$	2	68	18	$max\{58,18\} = 58$
$(3,0,1)$	1	39	39	39
$(3,0,2)$	1	40	40	40
$(3,1,1)$	1	46	46	46
$(3,1,2)$	1	47	47	47
$(4,0,1)$	1	36	36	36
$(4,0,2)$	1	37	37	37
$(4,1,1)$	1	40	40	40
$(4,1,2)$	1	58	$58 > T_3$	
$(4,1,2)$	2	79	29	$max\{58,29\} = 58$

Table 6.5: Possible Busy Periods

2 for the combinations $(2,1,1)$, $(2,1,2)$, and $(4,1,2)$ as the relative $w_{3,\tilde{v}}(1)$ is greater than T_3.

Once all worst case busy periods that are relative to all $\tilde{v} \in V_3$ are identified, the worst case response time of τ_3, R_3, is the maximum of all identified busy periods and found by applying Equation (6.13) (i.e. $w_{3,\tilde{v}}$'s column in Table 6.5). Thus, $R_3 = max\{38,39,57,58,39,40,46,47,36,37,40,58\} = 58 < D_3$, so τ_3 is schedulable.

6.4 Combined Analysis of Release Jitter and Arbitrary Deadlines

Section 6.1 restricts the system to have deadlines less than their relative periods and Section 6.2 restricts the system to have no release jitter. In this section we relax the two previous restrictions and present the worst case response time analysis of systems whose deadlines are arbitrary and the MF tasks are subjected to release jitter.

The first issue of the analysis is to identify the situation that leads to the worst case response time of the analysed MF task. Within the context of the analysis in this section, we consider the simultaneous release, of the critical frames for both the analysed MF task and higher priority MF tasks, is the situation that leads to the worst case response time of the analysed task. We call this situation the critical instance of a MF task; which is given by Definition 10.

Definition 10 . The critical instance of a non-AM multiframe task τ_i whose deadline is arbitrary and the MF tasks are subjected to release jitter *is the simultaneous release, that leads to the worst case response time of τ_i, of the critical frames of both τ_i and all MF tasks whose priorities are higher than τ_i's; where the simultaneous release takes its place rightmost in their release jitter interval whilst subsequent releases of the frames that follow the critical frames take their place leftmost in their relative release jitter intervals.*

Definition 10 considers the simultaneous releases of the critical frames of both τ_i and higher priority MF tasks. So, we use the presentation of the simultaneous releases as in the previous section; which is the cartesian product $\hat{V}_i$ of $\hat{L_j}$; where $\hat{L_j}$ is the critical frame locations of τ_j; $j = 1, 2, .., i$. Note that $\hat{V}_i$ takes into account the critical frames of the analysed MF task; where their locations are presented by $\hat{L_i}$. So, $\hat{V}_i$ is a set of vectors, each vector represents a combination of the critical frames of both τ_i and higher priority MF tasks.

Assuming Definition 10, we divide the response time analysis into two steps. Firstly, for each simultaneous release of τ_i and higher priority MF tasks (i.e. $\tilde{v} \in \hat{V}_i$), we find $w_{i,\tilde{v}}$ that is the worst case busy period of τ_i that is relative to $\tilde{v} \in \hat{V}_i$. Secondly, we

maximise all found worst case busy periods over all combinations $\tilde{v} \in \hat{V}_i$.

To find $w_{i,\tilde{v}}$, we have to consider all busy periods that could include interference from τ_i itself as well as higher priority MF tasks. So, assuming $w_{i,\tilde{v}}(q)$ is the q^{th} busy period, of τ_i, that is relative to[6] $\tilde{v}$, $w_{i,\tilde{v}}(q)$ is found by firstly finding $r_{i,\tilde{v}}(q)$. $r_{i,\tilde{v}}(q)$ is the response time of q frames starting from the frame that is synchronised with the higher priority MF tasks. The following lemma introduces a formula for finding $r_{i,\tilde{v}}(q)$.

Lemma 5 *Having a MF task τ_i in a system that is subjected to release jitter and arbitrary deadlines. $r_{i,\tilde{v}}(q)$ is τ_i's response time, that is relative to $\tilde{v}$ which represents a combination of the critical frames of both τ_i and the higher priority MF tasks, of q frames starting from the frame whose location is $\tilde{v}_i$. $r_{i,\tilde{v}}(q)$ is found by the smallest non-negative solution to Equation (6.14) assuming the priority ceiling protocols [66, 60].*

$$r_{i,\tilde{v}}(q) = \xi_i^{\tilde{v}_i}(q) + B_i + \sum_{j=1}^{i-1} \xi_j^{\tilde{v}_j}(\lceil \frac{r_{i,\tilde{v}}(q) + J_j}{T_j} \rceil). \tag{6.14}$$

where $\xi_i^{\tilde{v}_i}(q)$ is introduced by Definition 1 and B_i is the maximum blocking time of τ_i.

Proof

As we assume a simultaneous release of all higher priority MF tasks at the starting time of $r_{i,\tilde{v}}(q)$, we can represent $r_{i,\tilde{v}}(q)$ by a summation of the amount of execution of τ_i and the amount of interference from higher priority MF tasks. Using Lemma 4, the maximum interference within $r_{i,\tilde{v}}(q)$ from higher priority MF tasks and that is relative to the combination $\tilde{v}$ is given by

$$\sum_{j=1}^{i-1} \xi_j^{\tilde{v}_j}(\lceil \frac{r_{i,\tilde{v}}(q) + J_j}{T_j} \rceil).$$

Furthermore, the amount of execution of τ_i for q frames starting from the frame whose location is $\tilde{v}_i$ is given by $\xi_i^{\tilde{v}_i}(q)$. In addition, as $r_{i,\tilde{v}}(q)$ represents a continuous execution of a same priority MF task, the priority ceiling protocols [66, 60] would not allow this MF task to be blocked for more than once at most during the execution of $r_{i,\tilde{v}}(q)$. So, we just need to add the maximum blocking time to $r_{i,\tilde{v}}(q)$. Therefore, $r_{i,\tilde{v}}(q)$ is represented by the summation of the three terms as in Equation (6.14).□

[6] $\tilde{v}$ represents the simultaneous release of the critical frames of both τ_i and higher priority MF tasks.

Solving Equation (6.14) is done by forming a recurrence relationship as in Equation (6.15).

$$r_{i,\tilde{v}}^{l+1}(q) = \xi_i^{\tilde{v}_i}(q) + B_i + \sum_{j=1}^{i-1} \xi_j^{\tilde{v}_j}(\lceil \frac{r_{i,\tilde{v}}^{l}(q) + J_j}{T_j} \rceil); \tag{6.15}$$

where $l = 0,\ 1,\ 2,\ ..$ until $r_{i,\tilde{v}}^{l+1}(q) = r_{i,\tilde{v}}^{l}(q)$. However, if $r_{i,\tilde{v}}^{l+1}(q) > (q-1)T_i + D_i - J_i$ then τ_i is not schedulable.

Once $r_{i,\tilde{v}}(q)$ is calculated, $w_{i,\tilde{v}}(q)$ is found by taking out the overlapping execution that does not belong to the execution of the q^{th} frame. The following theorem proves a formula for finding $w_{i,\tilde{v}}(q)$.

Theorem 9 *Having a system of non-AM multiframe tasks, each task τ_i has an arbitrary deadline D_i and is subjected to release jitter J_i and a simultaneous release of the critical frames of τ_i and higher priority MF tasks that is presented by their locations $\tilde{v}$. The q^{th} busy period of τ_i that is relative to $\tilde{v}$ (i.e. $w_{i,\tilde{v}}(q)$) is given by Equation (6.16).*

$$\begin{aligned} w_{i,\tilde{v}}(q) &= r_{i,\tilde{v}}(q); && for\ q = 1, \\ &= r_{i,\tilde{v}}(q) - (q-1)T_i + J_i; && for\ q > 1. \end{aligned} \tag{6.16}$$

Proof

$w_{i,\tilde{v}}(q)$ starts from when the q^{th} frame (starting from the frame whose location is $\tilde{v}_i$) of τ_i is released; and ends by when this frame has finished its execution. $r_{i,\tilde{v}}(q)$ starts from when τ_i's frame whose location is $\tilde{v}_i$ is released and ends by when the q^{th} frame of τ_i has finished its execution. So, when $q = 1$, $r_{i,\tilde{v}}(q)$ and $w_{i,\tilde{v}}(q)$ start and end at the same time so $w_{i,\tilde{v}}(q) = r_{i,\tilde{v}}(q)$. However, when $q \geq 1$, $r_{i,\tilde{v}}(q)$ and $w_{i,\tilde{v}}(q)$ end at the same time but $r_{i,\tilde{v}}(q)$ starts earlier than $w_{i,\tilde{v}}(q)$ so the amount of $r_{i,\tilde{v}}(q)$ is greater than $w_{i,\tilde{v}}(q)$. To find $w_{i,\tilde{v}}(q)$ we subtract the start time of $w_{i,\tilde{v}}(q)$ (i.e. $\tilde{v}_i + (q-1)T_i - J_i$) from the start time of $r_{i,\tilde{v}}(q)$ (i.e. $\tilde{v}_i$) because $w_{i,\tilde{v}}(q)$ starts later than $r_{i,\tilde{v}}(q)$. In other words,

$$r_{i,\tilde{v}}(q) - w_{i,\tilde{v}}(q) = \tilde{v}_i + (q-1)T_i - J_i - \tilde{v}_i.$$

Therefore, $w_{i,\tilde{v}}(q) = r_{i,\tilde{v}}(q) - (q-1)T_i + J_i$; which is identical to Equation (6.16).□

Theorem 9 finds the q^{th} busy period, of τ_i, that is relative to $\tilde{v}$. To find the worst case busy period that is relative to $\tilde{v}$ we have to maximise the busy periods that are

relative to $\tilde{v}$ over all values of q as in the following Corollary.

Corollary 3 *$w_{i,\tilde{v}}$ is the worst busy period of a MF task τ_i that is relative to the simultaneous release, of the critical frames, that is represented by $\tilde{v}$. $w_{i,\tilde{v}}$ is given by Equation (6.17).*

$$w_{i,\tilde{v}} = \max_{q=1,2,..} \{w_{i,\tilde{v}}(q)\}; \tag{6.17}$$

where $q = 1,2,..$ until $w_{i,\tilde{v}}(q) \leq T_i - J_i$ for $q = 1$ and $w_{i,\tilde{v}}(q) \leq T_i$ for $q > 1$. This is because τ_i, in this case, stops interfering its execution when $w_{i,\tilde{v}}(q)$ falls in the same period it is released in.

Up to this point, for each $\tilde{v}$ we have identified the relative worst case busy period. So, to find the worst case response time of τ_i we have to maximise these $w_{i,\tilde{v}}$ over all possible combinations $\tilde{v}$ as in the following Corollary.

Corollary 4 *The worst case respons time of a MF τ_i, in a system that is subjected to release jitter and arbitrary deadlines, is the maximum worst case busy period of τ_i over all combinations of the critical releases of the higher priority MF tasks. This maximisation is presented by Equation (6.18).*

$$R_i = \max_{\tilde{v} \in \hat{V}_i} \{w_{i,\tilde{v}}\} \tag{6.18}$$

Schedulability Test

We already know that R_i is found from when τ_i is released while D_i is scheduled from when τ_i arrives in the system. So, the schedulability test is as the following: τ_i is schedulable if $R_i \leq D_i - J_i$; where R_i is found by applying Equation (6.18).

6.5 Example

To apply the analysis in this section, assume a simple example system that consists of two tasks; τ_1 with only one frame and τ_2 with three frames. To simplify the example we assume all blocking times are zero. To analyze the schedulability of τ_2 we

task	C	T	D	J
τ_1	3	5	5	1
τ_2	$(2,3,4)$	10	20	2

Table 6.6: Example System

have to maximise all its worst case busy periods over all $\tilde{v} \in V_2$ which represent the combinations of the critical frames of both τ_2 and higher priority MF tasks.

First of all, using policy in Section 5.1, we find the critical frame locations of τ_1 and τ_2 (i.e. L_1 and L_2 respectively). So, $L_1 = (0)$ and $L_2 = (1,2)$ and therefore,

$$V_2 = \{(0,1),(0,2)\}.$$

Now, for each $\tilde{v} \in V_2$ we find all busy periods that could include interference from τ_2. In other words, we apply Theorem 9 to find $w_{2,\tilde{v}}(q)$ for all $q = 1,\ 2,\ ..$ until $w_{2,\tilde{v}}(q) \leq T_2 - J_2$ for $q = 1$ or $w_{2,\tilde{v}}(q) \leq T_2$ for $q > 1$. So, for $\tilde{v} = (0,1)$, we have to find $w_{2,(0,1)}(q)$ which requires finding $r_{2,(0,1)}(q)$ by applying Equation (6.15) where $B_2 = 0$.
$q = 1, \quad r^{l+1}_{2,(0,1)}(1) = \xi^1_2(1) + \sum_{j=1}^{2-1} \xi^{\tilde{v}_j}_j(\lceil \frac{r^l_{2,(0,1)}(1)+J_j}{T_j} \rceil)$. To solve this equation,

$$\begin{aligned}
l = 0, \quad r^1_{2,(0,1)}(1) &= \xi^1_2(1) + \xi^0_1(\lceil \frac{r^0_{2,(0,1)}(1)+J_1}{T_1} \rceil). \\
&= 3 + \xi^0_1(\lceil \frac{3+1}{5} \rceil) \\
&= 3+3 = 6. \\
l = 1, \quad r^2_{2,(0,1)}(1) &= \xi^1_2(1) + \xi^1_1(\lceil \frac{r^1_2(1)+J_1}{T_1} \rceil) \\
&= 3+6 = 9. \\
l = 2, \quad r^3_{2,(0,1)}(1) &= 3+6 = 9 = r^2_2(1).
\end{aligned}$$

So, $r_{2,(0,1)}(1) = 9$, therefore $w_{2,(0,1)}(1) = r_{2,(0,1)}(1) = 9$. $w_{2,(0,1)}(1) > T_2 - J_2$, so we increase q to 2 and apply Equations (6.15) and (6.16) for $i = 2$, $q = 2$ and $r^0_{2,(0,1)}(2) = \xi^1_2(2) = 7$, so we get

$$r^{l+1}_{2,(0,1)}(2) = \xi^1_2(2) + \xi^0_1(\lceil \frac{r^l_{2,(0,1)}(2) + J_1}{T_1} \rceil).$$

By solving this equation we get, $r_{2,(0,1)}(2) = 19$, therefore $w_{2,(0,1)}(2) = r_2(1) - T_2 + J_2 = 19 - 10 + 2 = 11$. $w_{2,(0,1)}(2) > T_2$, so we increase q to 3 and again apply Equations (6.15) and (6.16) to get $r_{2,(0,1)}(3) = 24$, so $w_{2,(0,1)}(3) = 24 - 20 + 2 = 6$. $w_{2,(0,1)}(3) \leq T_2$, so we stop increasing q and finding more busy periods that are relative to $\tilde{v} = (0,1)$.

As all needed busy periods are identified, we now find $w_{2,(0,1)}$ by applying Corollary 3 (i.e. Equation (6.17)). Therefore,

$$w_{2,(0,1)} = \max\{9, 11, 6\} = 11.$$

Similarly, when $\tilde{v} = (0,2)$ we find $w_{2,(0,2)}(q)$ for $q = 1, 2, ..$ until $w_{2,(0,2)}(q) \leq T_2$, so we get the values in Table 6.7.

$\tilde{v}$	q	$r_{2,\tilde{v}}(q)$	$w_{3,\tilde{v}}(q)$	$w_{3,\tilde{v}}$
$(0,1)$	1	9	$9 > T_2$	
	2	19	$11 > T_2$	
	3	24	6	$max\{9,11,6\} = 11$
$(0,2)$	1	13	$13 > T_3$	
	2	18	$10 \leq T_3$	$max\{13,10\} = 13$

Table 6.7: Possible Busy Periods

Thus, the worst case response time of τ_2 is found by applying Equation (6.18)

$$R_2 = \max\{11, 13\} = 13 \leq D_2 - J_2.$$

So, τ_2 is schedulable.

6.6 Summary

This chapter has shown the flexibility of the response time scheduling analysis of non-AM multiframe tasks by extending the analysis in two ways. One is to include MF tasks that are subjected to release jitter, and the other is to include MF tasks whose deadlines are arbitrary so interference from the analysed MF task has been taken into account. Then, the two models have been combined and the exact response time analysis has been presented for the new combined model.

7 Exact Analysis of Frame Specific Deadlines

Up to this chapter, the response time analysis of MF tasks assumes that all frames of the MF task have the same deadline, so analysing the maximum response of the critical frames is enough to decide the schedulability of the MF task itself. In this chapter, we generalise the system model to the situation that is called the frame specific deadline model; where the MF task could have different deadlines relative to each of its frames. So, each MF task τ_i has n_i deadlines (D_i^k); for each $k = 0,..,n_i - 1$. The model in this chapter covers the arbitrary deadlines model but no blocking from lower priority tasks is allowed to simplify the presentation.

The frame specific deadline model rises an issue of how to optimise the priority assignment for the MF tasks in the system. This chapter suggests an optimal priority assignment that can be used in this model.

This chapter is organised as the following: the next section presents the worst case response time analysis of the model assuming that all deadlines of each MF task are less than their relative period, and that priorities have been allocated. Section 7.2 relaxes the restriction on the deadlines and presents the worst case response time analysis of the model assuming that all deadlines of each MF task are arbitrary, so interference from the analysed task itself has to be taken into account in the analysis. In Section 7.3, the analysis is practically illustrated by a numeric example. Section 7.4 covers the priority assignment that is used for the frame specific deadlines model.

7.1 Exact Response Time Analysis of MF Task with no Interference from the Analysed Task

In general, as τ_i has n_i deadlines relative to its frames, to test the schedulability of the MF task τ_i we have to find the worst case response time for each of its frames, $R_i(C_i^k); k = 0..n_i - 1$, and then check $R_i(C_i^k) \leq D_i^k$ for all values of k. However, when there is no interference from previous frames of the same task, there are some cases where there is no need to check the schedulability of all n_i frames. One of these cases is when the schedulability of the x^{th} frame implies the schedulability of the y^{th} frame, so no need to explicitly check the schedulability of the y^{th} frame. This argument leads to a concept of coverage.

Definition 11 . *Having two frames x and y of a MF task τ, we say that* **frame** *x* ***covers*** **frame** *y if the schedulability of x implies the schedulability of y.*

Applying Definition 11 reduces the number of frames that are needed for checking the schedulability status of the MF task; where only uncovered frames are required for testing the schedulability of the MF task. Within the following two subsections we first introduce a criterion for identifying the covered frames, then we introduce the response time analysis within the frame specific deadlines scenario assuming no interference from the analysed task itself.

7.1.1 Identifying Covered frames

To investigate a criterion for identifying covered frames, we first introduce a simple example to illustrate how the schedulability of an uncovered frame leads to the schedulability of the covered frame. Assume a MF task τ_i with two frames one frame has a worst-case execution time of 3 (i.e. $C_i^0 = 3$) and a deadline equals 10 (i.e. $D_i^0 = 10$); and another frame with $C_i^1 = 2$ and $D_i^1 = 12$. Then, the schedulability of the first frame leads to the schedulability of the second frame because $C_i^0 > C_i^1$ and $D_i^0 < D_i^1$. Informally, if 3 units of execution can be achieved in 10 units then, clearly, 2 units of execution are achievable in 12 units. Furthermore, if 3 units are executable in 10 units then 2 units are guaranteed to be also executable in 9 units. The following

lemma introduces a schedulability criterion for a frame of a MF task depending on the schedulability of another frame of the same MF task.

Lemma 6 *For a MF task τ_i whose execution times are C_i^k, deadlines are D_i^k; $k = 0,..,n_i-1$, and $R_i(C_i^x)$ is the worst case response time of an arbitrary frame whose execution time is C_i^x, then having $R_i(C_i^x) \leq D_i^x$ leads to $R_i(C_i^x - p) \leq D_i^x - p$; where p is an integer and $C_i^x \geq p \geq 0$.*

Proof As we assumed no interference from same priority tasks, finding $R_i(C_i^x)$ is found as a collection of two kinds of execution one is the execution of the x^{th} frame of τ_i which is represented by C_i^x, and the other is related to the interference on the execution of C_i^x. In other words,

$$R_i(C_i^x) \; = \; C_i^x \; + \; I(C_i^x)$$

Where $I(C_i^x)$ stands for the interference on the x^{th} execution of τ_i. So, having $R_i(C_i^x) \leq D_i^x$ means that $C_i^x \; + \; I(C_i^x) \leq \; D_i^x$ and therefore for any positive integer p that is less than C_i^x then

$$C_i^x \; - \; p \; + \; I(C_i^x) \leq \; D_i^x \; - \; p \tag{7.1}$$

similarly, $R_i(C_i^x - p)$ is found as

$$R_i(C_i^x - p) \; = \; C_i^x \; - \; p \; + \; I(C_i^x - p)$$

where,

$$C_i^x \; - \; p \; + \; I(C_i^x - p) \; \leq \; C_i^x \; - \; p \; + \; I(C_i^x) \tag{7.2}$$

because obviously $I(C_i^x - p) \; \leq \; I(C_i^x)$ for each simultaneous release of the frames whose execution times are $C_i^x - p$ and C_i^x with the higher priority MF tasks.

It is clear that the right side of inequality (7.2) is identical to the left side of inequality (7.1). Therefore, we can say that

$$R_i(C_i^x - p) \; \leq \; C_i^x \; - \; p \; + \; I(C_i^x) \leq \; D_i^x \; - \; p.$$

In other words, $R_i(C_i^x - p) \; \leq \; D_i^x \; - \; p.\square$

Having Lemma 6, the following theorem introduces a criterion for identifying covered frames of a MF task.

Theorem 10 *For a MF task τ_i whose execution times are C_i^k and deadlines are D_i^k; $k = 0,..,n_i-1$, the x^{th} frame of τ_i covers the y^{th} frame if $C_i^x \geq C_i^y$ and $D_i^x \leq D_i^y + (C_i^x - C_i^y)$.*

Proof: To prove the theorem, we assume that x^{th} frame is schedulable and then check the schedulability of the y^{th} frame. As the x^{th} frame is schedulable then, $R_i(C_i^x) \leq D_i^x$; where $R_i(C_i^x)$ is the response time of the x^{th} frame. Using Lemma 6, we find that

$$R_i(C_i^x) \leq D_i^x \Rightarrow R_i(C_i^x - p) \leq D_i^x - p \text{ where } C_i^x \geq p \geq 0.$$

Let $p = (C_i^x - C_i^y)$, so,

$$R_i(C_i^x - (C_i^x - C_i^y)) \leq D_i^x - (C_i^x - C_i^y)$$

Therefore,

$$R_i(C_i^y) \leq D_i^x - (C_i^x - C_i^y) \tag{7.3}$$

We already have

$$D_i^x \leq D_i^y + (C_i^x - C_i^y) \tag{7.4}$$

So, by substituting inequality (7.4) for inequality (7.3) we get $R_i(C_i^y) \leq D_i^y + (C_i^x - C_i^y) - (C_i^x - C_i^y)$. Hence, $R_i(C_i^y) \leq D_i^y$.

Therefore, the y^{th} frame is schedulable; which means that the schedulability of the x^{th} frame leads to the schedulability of the y^{th} frame. So, x^{th} frame covers y^{th} frame.□

Using Theorem 10 in the scheduling analysis of the MF task τ_i, whose frames have specific deadlines, reduces the number of frames that are required for the scheduling test of τ_i. This is because of the efficiency of only analysing the uncovered frames for the schedulability status. The following is the policy of analysing the response time of the uncovered frames.

7.1.2 Response Time Analysis

To analyze the schedulability of a MF task with frame specific deadlines, we just need to analyze the worst case response time of its uncovered frames. Once all its uncovered frames are schedulable we say that the MF task is schedulable.

To analyze the response time of an uncovered frame of a MF task, we apply the worst case response time analysis of non-AM multiframe tasks that is given in Section 5.2 substituting the execution time of the analysed uncovered frame for the execution time of the peak frame. For more clarification, to analyze the response time of the uncovered frame whose execution time is C_i^x we first find the worst case response time of this frame that is relative to the combination $\tilde{v} \in \hat{V}_i$ by applying Equation (7.5); which is an application of Equation (5.5) with $C_i^{m_i} = C_i^x$.

$$R_{i,\tilde{v}}(C_i^x) = C_i^x + \sum_{j=1}^{i-1} \xi_j^{\tilde{v}_j} \left(\lceil \frac{R_{i,\tilde{v}}(C_i^x)}{T_j} \rceil\right); \tag{7.5}$$

where $\tilde{v}$ represents the combination of the critical frames of MF tasks whose priorities are higher than τ_i.

Equation (7.5) can be solved by forming an iterative equation given by Equation (7.6).

$$R_{i,\tilde{v}}^{l+1}(C_i^x) = C_i^x + \sum_{j=1}^{i-1} \xi_j^{\tilde{v}_j} \left(\lceil \frac{R_{i,\tilde{v}}^{l}(C_i^x)}{T_j} \rceil\right); \tag{7.6}$$

$R_{i,\tilde{v}}^0(C_i^x) = C_i^x$ and $R_{i,\tilde{v}}(C_i^x)$ is found when $R_{i,\tilde{v}}^{l+1}(C_i^x) = R_{i,\tilde{v}}^{l}(C_i^x)$. However, if $R_{i,\tilde{v}}^{l+1}(C_i^x) > D_i^x$ then the frame whose execution time is C_i^x is not schedulable and therefore τ_i is not schedulable.

To find the worst case response time of the frame whose execution time is C_i^x, we maximise $R_{i,\tilde{v}}(C_i^x)$ over all $\tilde{v} \in \hat{V}_i$. In other words, we apply Equation (5.4) with the same $\hat{V}_i$ that is defined in Section 5.1.

Example

Assume a simple system with two MF tasks τ_1 with only one frame and τ_2 with 4 different frames as in Table 7.1. To analyze τ_2's response time we firstly have to identify its covered frames. To identify the covered frames of τ_2 we apply the criterion of Theorem 10 on τ_2's frames starting with its peak frame.

task	C	D	T	Priority
τ_1	3	6	10	high
τ_2	$(1,3,5,2)$	$(8,10,8,5)$	10	low

Table 7.1: Example System

Basically, the third frame of τ_2 (i.e. the frame whose execution time is 5) covers all other frame of τ_2. That is because
$5 > 1$ and $8 \leq 8+4$, so the frame whose execution time is 5 covers the frame whose execution time is 1.
$5 > 3$ and $8 \leq 10+2$, so the frame whose execution time is 5 covers the frame whose execution time is 3.
$5 > 2$ and $8 \leq 5+3$, so the frame whose execution time is 5 covers the frame whose execution time is 2.

Therefore, to check the schedulability status of τ_2 we just need to analyze the worst case response time of the frame whose execution time is 5 and its location is 2. For this reason, we first find $\hat{V}_2 = \{(0)\}$ because we only have one higher priority task with only one frame. Then we apply Equation (7.6) so we get.

$$R^{l+1}_{2,(0)}(C^2_2) = C^2_2 + \xi^0_1 \ (\lceil \frac{R^l_{2,(0)}(C^2_2)}{T_1} \rceil);$$

Solving this equation leads to $R_{2,(0)}(C^2_2) = 8$, so $R_2(C^2_2) = 8 \leq D^2_2 = 8$. So, the frame whose execution time is 5 is schedulable and therefore τ_2 is schedulable.

7.1.3 Improving the Efficiency of the Analysis

One way of improving the efficiency of response time analysis of the uncovered frames, that are obtained by Theorem 10, is to reduce the number of iterations that are used in the recurrence relations that solve the response time equations. An expeditious way of solving the response time equation (i.e. Equation (7.6)) is to first analyze the schedulability of the frame whose execution time is the minimum and once found

schedulable we then solve the recurrence relation of the response time of the frame whose execution time is immediately greater than the minimum and we start the solution with the response time of the frame whose execution time is the minimum. For example, if we are checking the schedulability status of three frames with the execution times and deadlines $(2,3,8)$ and $(10,15,30)$ respectively, the execution time value of 2 is used as a starting point of the recurrence relation of the response time equation. Once we get the worst case response time less than 10 (for example 8) then we check the frame with the greater execution time (i.e. the second frame with the execution time 3). The starting point of the recurrence relation of the response time equation is now 8 instead of the 3 (i.e. $R^0_{i,\tilde{v}}(3) = R_{i,\tilde{v}}(2)$), as the solution for the value 3 cannot be less than the solution of 2. Similarly, when the new response time is found less than 15 (e.g. 12) then we check the third frame with the execution time 8 with starting point of 12. In fact, this means that we do not re-run the solution process for each frame of the analysed MF task.

7.2 Exact Response Time Analysis of MF Tasks Having Deadlines Beyond the Period

The analysis in the previous section was based on analysing the interference from higher priority MF tasks and does not consider any interference from the analysed MF task itself. However, this section covers the worst case response time analysis of MF tasks whose deadlines are greater than their periods so interference from the analysed task itself has to be taken into account.

The coverage concept that is introduced in the previous section is not applicable any more when the MF task has arbitrary deadlines. This is because there could be two frames of a MF task τ_i whose execution times are C_i^x and C_i^y; where $C_i^x > C_i^y$ but the interference from τ_i within C_i^y is greater than the interference from τ_i within C_i^x; in the sense that results $R_i(C_i^x) < R_i(C_i^y)$. Therefore the schedulability of the frame whose execution time is C_i^x does not necessarily lead to the schedulability of the frame whose execution time is C_i^y. Therefore, to analyze the schedulability of the MF task τ_i we have to analyze the worst case response time of all its frames.

To analyze the response time of a frame of a MF task τ_i, we have to consider all simultaneous releases of all frames of τ_i with the higher priority MF tasks. This is because the simultaneous release of the higher priority tasks leads to the worst case preemption of a lower priority task. In addition, we analyze the simultaneous release of each frame of τ_i and critical frames of higher priority MF tasks to analyze the interference that could be generated by each frame of τ_i within the analysed frame. To clarify the policy of the analysis, assume we are analysing the frame whose location is q in the MF task τ_i, so we have to consider in the analysis all simultaneous releases of all frames of τ_i with the critical frames of higher priority MF tasks to check if the simultaneous release could lead τ_i to interfere with the frame whose location is q.

Assume f is the location of the frame of τ_i that is released simultaneously with the higher priority MF tasks, so values of f are $f = 0,1,..,n_i - 1$. For the purpose of the analysis, we recall the term busy period of a frame, that is the time from when this frame is released until it finishes its execution. The worst case response time of the frame whose location is q is the maximum busy period of this frame for all simultaneous releases of all frames whose locations are $f = 0,1,..,n_i - 1$ with the critical frames of the higher priority MF tasks. So, response time analysis also has to consider all combinations of the critical frames of the higher priority MF tasks. In other words, the worst case response time analysis has to consider all combinations of f and critical frames of higher priority MF tasks. We present this combination as $\tilde{v} \in \hat{V}_i$ where $\hat{V}_i$ is given by

$$\hat{V}_i = \hat{L_1} \times \hat{L_2} \times .. \times \hat{L_{i-1}};$$

where $\hat{L_j}; j = 1,\ 2,\ ,\ ..\ ,\ i-1$ is the set of locations of the critical frames of the MF task τ_j.

The following observation is pertinent to the situation when a frame of τ_i could interfere with another frame of the same MF task.

Observation 1 *Having MF task τ_i with n_i frames that are indexed from 0 to $n_i - 1$. When τ_i is released with the frame whose location is f, τ_i interferes with the frame*

whose location is q when the number of interference from τ_i is:

$$\begin{aligned} q-f+1; \quad & \text{when } q \geq f, \\ n-(f-q-1); \quad & \text{when } f > q. \end{aligned} \tag{7.7}$$

Basically, Observation 1 measures, on one direction, the number of frames that f has to enter to reach the frame whose location is q taking into account its own frame and the q frame. For example, when $n = 5$, $q = 0$, and $f = 2$; the frame whose location is f has to enter 4 frames to reach the frame whose location is q because f has to pass the frames whose locations are $2, 3, 4$ and 0.

As the worst case response time of a frame is the longest busy period that this frame can practice, we have to find a way that calculates the busy periods of this frame. However, finding the busy period has to take into account the interference from the MF task itself as well as the interference from higher priority MF tasks. Taking Observation 1 into account, the following theorem proves a formula for finding the busy period of the frame whose location is q of a MF task τ_i. This busy period is relative to the simultaneous release of the frame whose location is f of the MF task τ_i and the critical frames, whose locations are presented by $\tilde{v} \in \hat{V}_i$, of the higher priority MF tasks.

Theorem 11 *Having a system of MF tasks. $w_{i,\tilde{v},f}(C_i^q)$ is the busy period, of a frame of a MF task τ_i, with location q that is relative to the simultaneous release of the frame whose location is f from τ_i with the critical frames of the higher priority MF tasks whose locations are presented by $\tilde{v}$. $w_{i,\tilde{v},f}(C_i^q)$ is given by Equation (7.8) .*

$$w_{i,\tilde{v},f}(C_i^q) = r_{i,\tilde{v},f}(C_i^q) - (t-1)T_i; \tag{7.8}$$

where

$$r_{i,\tilde{v},f}(C_i^q) = \xi_i^f(t) + \sum_{j=1}^{i-1} \xi_j^{\tilde{v}_j}(\lceil \frac{r_{i,\tilde{v},f}(C_i^q)}{T_j} \rceil); \tag{7.9}$$

and where t is given by

$$\begin{aligned} t &= q - f + 1; \ \ \textit{when } q \geq f, \\ t &= n - (f - q - 1); \ \ \textit{when } f > q. \end{aligned} \tag{7.10}$$

Proof

To prove the theorem, we will assume that the simultaneous release of the frame whose location is f leads to continuous busy periods of τ_i's frames until interfering the frame whose location is q. So, according to Observation 1, τ_i is invoked for t number of times (t is given by Equation (7.10)) starting from the frame whose location is f. So, the amount of execution that τ_i has to perform is given by ${\xi_i}^f(t)$ and therefore, the time that is consumed for achieving this amount of execution is presented by $r_{i,\tilde{v},f}(C_i^q)$ and given by Equation (7.9).

The busy period of the frame whose location is q starts from when this frame is released until finishing its execution that is presented by C_i^q. On the other hand, $r_{i,\tilde{v},f}(C_i^q)$ starts from when the frame whose location is f is released until the frame whose location is q finishes its execution. So, both $r_{i,\tilde{v},f}(C_i^q)$ and $w_{i,\tilde{v},f}(C_i^q)$ have same end and different starting point. So, as the busy period of a frame is the time from when this frame is released until finishing its execution, the busy period of the frame whose location is q is given by Equation (7.8). □

Equation (7.9) is solved by forming a recurrence relationship as in Equation (7.11)

$$r_{i,\tilde{v},f}^{l+1}(C_i^q) = {\xi_i}^f(t) + \sum_{j=1}^{i-1} {\xi_j}^{\tilde{v}_j}(\lceil \frac{r_{i,\tilde{v},f}^{l}(C_i^q)}{T_j} \rceil) \tag{7.11}$$

where $r_{i,\tilde{v},f}^{0}(C_i^q) = {\xi_i}^f(t)$, and $l = 0, 1, 2, ..$ until $r_{i,\tilde{v},f}^{l+1}(C_i^q) = r_{i,\tilde{v},f}^{l}(C_i^q)$. However, if $r_{i,\tilde{v},f}^{l+1}(C_i^q) - (t-1)T_i > D_i^q$, τ_i is not schedulable.

Note that if one of the busy periods of τ_i extends beyond its deadlines, the frame will miss its deadline and will not be schedulable and therefore the whole MF task will not be schedulable. So, if $w_{i,\tilde{v},f}(C_i^q) > D_i^q$, then τ_i is unschedulable.

Corollary 5 *Having $w_{i,\tilde{v}}(C_i^q)$ as the worst case busy period, of the frame whose lo-*

cation is q and that is relative to the combinations of the critical frames of the higher priority MF tasks. $w_{i,\tilde{v}}(C_i^q)$ *is the maximum busy period over all simultaneous releases of* τ_i*'s frames. In other words,*

$$w_{i,\tilde{v}}(C_i^q) = \max_{f=0,1,..,n_i-1} \{w_{i,\tilde{v},f}(C_i^q)\} \tag{7.12}$$

Corollary 6 *The worst case response time of a frame whose location is q is given by Equation (7.13).*

$$R_i(C_i^q) = \max_{\tilde{v}\in\hat{V}_i} \{w_{i,\tilde{v}}(C_i^q)\} \tag{7.13}$$

Scheduling Test

The schedulability test of a MF task in the frame specific deadline scenario is the following: τ_i is schedulable if $R_i(C_i^q) \leq D_i^q;\ \forall q = 0,\ 2,\ ..n_i - 1$; where $R_i(C_i^q)$ is found by Equation (7.13).

7.3 Example

task	*C*	*D*	*T*	Priority
τ_1	3	6	5	high
τ_2	$(5,2,1,3)$	$(20,10,8,10)$	10	low

Table 7.2: Example System

Assume a simple system with two MF tasks, τ_1 with only one frame and τ_2 with 4 different frames as in Table 7.2. To analyze the schedulability of τ_2, we have to analyze all simultaneous releases of τ_2 and τ_1 and also we have to find $\hat{V}_2$ which is $\hat{L_1}$ because we only have two tasks.

$\hat{L_1} = \{0\}$. So, $\hat{V_2} = \{0\}$.

f belongs to the set of all frame locations of τ_2, so , $f \in \{0,\ 1,\ 2,\ 3\}$. Now, to analyze

the worst case response time of the frame whose location is q, we have to find its maximum busy period over all simultaneous releases of the frames whose locations are $f \in \{0,\ 1,\ 2,\ 3\}$ and for each $\tilde{v} \in \hat{V}_i$. So, for each f and q we first find the relative t by applying Equation (7.10) so we get values in Table 7.3. Then for each $\tilde{v} \in V_i$ we find the response time of t frames starting from the frame whose location is f and ending by the frame whose location is q; which is presented by $r_{i.\tilde{v},f}(C_i^q)$ and found by applying Equation (7.9). Therefore, the busy period of the frame whose location is q, $w_{i.\tilde{v},f}(C_i^q)$, that is relative to f and $\tilde{v}$ is found by applying equation (7.8).

f ⟶ / q ↓	0	1	2	3
0	1	4	3	2
1	2	1	4	3
2	3	2	1	4
3	4	3	2	1

Table 7.3: Values of t

f ⟶ / q ↓	0	1	2	3	$R_2(C_2^q)$
0	**14**	-1	4	10	**14**
1	**9**	5	-1	-10	**9**
2	0	-1	**4**	-1	**4**
3	-1	-5	0	**9**	**9**

Table 7.4: Values of $w_{2,\tilde{v},f}(C_2^q)$

For example, to find $w_{2,\tilde{v},0}(C_2^1)$ (i.e. $f = 0$ and $q = 1$) we first find $t = 2$. Then we find $r_{2,\tilde{v},0}(C_2^1)$ by applying Equation (7.9). $\tilde{v} = (0)$ as $\hat{V}_2$ has only one value that is (0). So,

$$r_{2,(0),0}(C_2^1) = \xi_2^0(2) + \sum_{j=1}^{1} \xi_j^{\tilde{v}_j}(\lceil \frac{r_{2,\tilde{v},0}(C_i^1)}{T_j} \rceil).$$

By solving this equation we get $\xi_2^0(2) = 7$, $r^1_{2,(0),0} = 7+6 = 13$.
$r^2_{2,(0),0}(C_2^1) = 7+9 = 16$.
$r^3_{2,(0),0}(C_2^1) = 7+12 = 19$.
$r^4_{2,(0),0}(C_2^1) = 7+12 = 19 = r^3_{2,(0),0}(C_2^1)$.
So, $r_{2,(0),0}(C_2^1) = 19$.
To find $w_{2,(0),0}(C_2^1)$ we apply Equation (7.8) to get
$w_{2,(0),0}(C_2^1) = r_{2,(0),0}(C_2^1) - T_2 = 19 - 10 = 9$

Similarly, we find all $w_{2,(0),f}(C_2^1)$ for all possible values of f so we get the values in the third line of Table 7.4. As there is only one value of $\tilde{v} = (0)$, there is only one combination of the critical frames of the higher priority tasks. So,
$w_{2,f}(C_2^1) = w_{2,0,f}(C_2^1)$. Therefore, to find the maximum busy period of the frame

whose location is 1, we maximise $w_{2,f}(C_2^1)$ over all values of f. In other words,
$R_2(C_2^1) = \max_{f\in\{0,1,2,3\}}\{w_{2,f}(C_2^1)\} = 9 < D_2^1$.
Similarly,
$w_{2,f}(C_2^q) = w_{2,0,f}(C_2^q);\ \forall q = 0,1,2,3$. So,
$R_2(C_2^0) = \max_{f\in\{0,1,2,3\}}\{w_{2,f}(C_2^0)\} = 14 < D_2^0$.
$R_2(C_2^2) = \max_{f\in\{0,1,2,3\}}\{w_{2,f}(C_2^2)\} = 4 < D_2^2$.
$R_2(C_2^3) = \max_{f\in\{0,1,2,3\}}\{w_{2,f}(C_2^3)\} = 9 < D_2^3$.
As all $R_2(C_2^q) \leq D_2^q\ \forall q = 0,1,2,3$, τ_2 is schedulable.

Release jitter could also be added to this analysis following the approaches that is given in Sections 4.3 and 6.4.

7.4 Policy of Assigning Priorities to the MF Tasks

All frames of a MF task have the same priority and also no blocking is allowed in the model, so the response time of each frame of τ_i is not dependent upon lower priority tasks and also does not increase when it is assigned a higher priority nor decrease when it is assigned a lower priority. In addition, the response time of each frame of τ_i is also not dependent upon the relative priority ordering of higher priority MF tasks because we check all combinations of the critical frames of both τ_i and higher priority MF tasks to check the schedulability of τ_i. So, the optimal priority assignment that is presented in [7, 5] and reviewed in Section 2.3.3 is applicable to our model; where the priority assignment scheme depends on finding the MF task that is schedulable at the lowest priority (i.e. priority of N) then the schedulable MF task that is relative to the priority $N-1$, and so on until we get all priorities assigned to the MF tasks whilst preserving schedulability. If we did not find a schedulable MF task at one level of the priorities then the system is unschedulable for any priority assignment.

Example

To illustrate the policy of the priority assignment, Table 7.5 presents a simple example of two MF tasks τ_A and τ_B; where τ_A has only one frame and τ_B has three frames with three deadlines. Clearly, DM priority assignment is not applicable to this example as the deadline of τ_A lies between the deadlines of τ_B.

task	*C*	*D*	*T*
τ_A	3	6	10
τ_B	$(1,3,4)$	$(5,10,8)$	5

Table 7.5: Example System

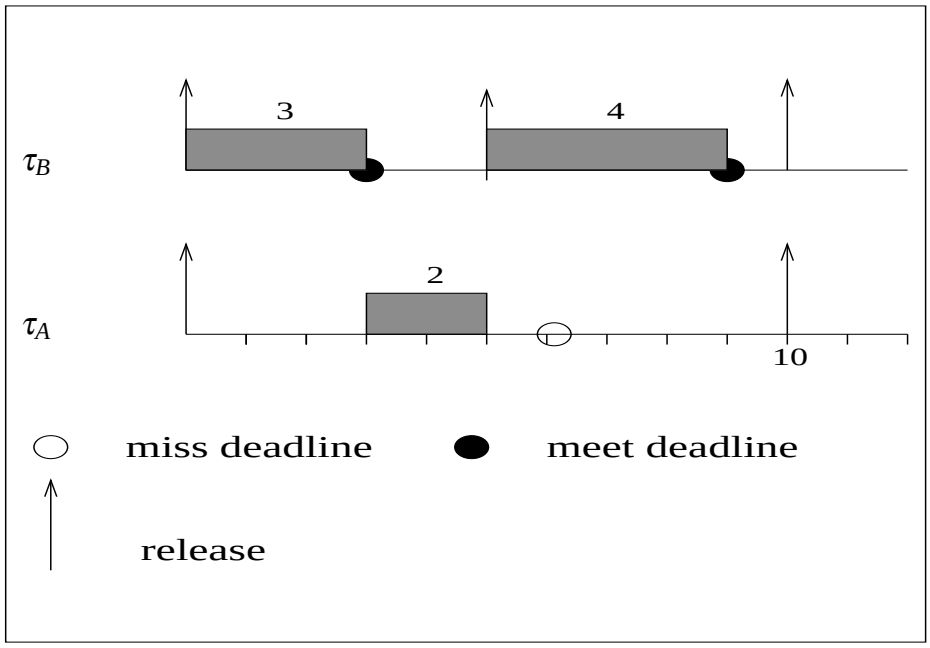
Furthermore, if we assign τ_A the lowest priority (i.e. 2), we find that τ_A is unschedulable when τ_B is released with the execution time of 3 or 4; whilst τ_A and τ_B are schedulable when τ_B is assigned priority 2. Figure 7.1 presents the timeline diagram to illustrate the execution of τ_A and τ_B when they are assigned different priorities. Figure 7.1 shows that in the worst case, the response times of τ_B when it is assigned the priority 2 are $(6,7,4)$.

7.5 Summary

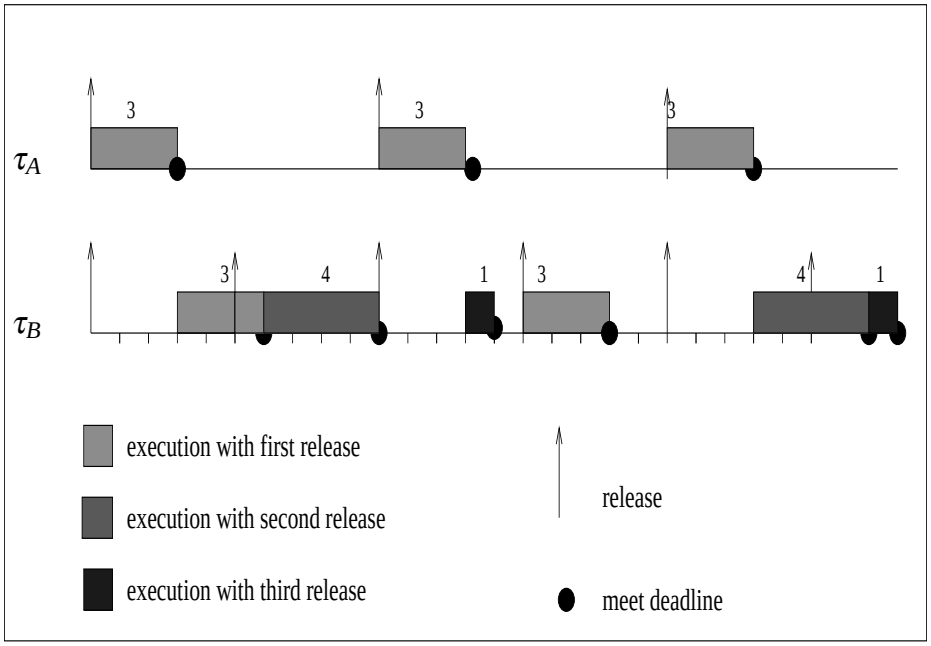
This chapter has presented exact worst case response time scheduling analysis for MF tasks whose frames could have different deadlines (i.e. frame specific deadlines). The analysis is presented in two steps regarding to the state of the MF's deadlines.

In the first step we restrict the deadlines to be less than or equal to the relative period, so no interference from the analysed task is considered. In this state, we introduced a coverage concept to reduce the number of frames, of the analysed task, that are needed for checking the schedulability status of the analysed MF task. This chapter has shown that we sufficiently need to analyze the uncovered frames of the analysed MF task to check its schedulability status. Further to the presentation of the basic response time analysis of frame specific deadlines, we have introduced a way to reduce the number of iterations used in finding the response time of a frame of a MF task.

In the second step we have relaxed the restriction of having deadlines less than the relative period and presented exact response time analysis. The coverage criterion that was presented in the first step is not applicable to MF tasks whose deadlines are arbitrary. Although the coverage criterion could be improved to cope with the arbitrary deadlines, we analysed all frames for checking the schedulability status of the analysed MF task.



(*execution of* τ_A and τ_B when τ_A's priority is the lowest)



(execution of τ_A and τ_B when τ_B's priority is the lowest)

Figure 7.1: Timeline Figure of τ_A and τ_B's execution

Finally, in this chapter we have considered a priority assignment for frame specific deadlines model. We have shown that the priority assignment that was presented by

Audsley [7, 5] is applicable to this model and we have explained the procedure of its application by a simple numeric example.

8 Approaches for Sufficient Scheduling Tests

Exact response time scheduling analysis becomes exhaustively intractable when the systems are respectively large. However, sufficient tractable approaches solve this problem; where a real-time system is exactly schedulable if it is schedulable using a specific approach. This chapter introduces and compares four sufficient approaches with the usage of the given response time analysis in this thesis. These approaches are called the maximum, the reordering, the complementary and the max accumulations approaches. The first three approaches depend on transforming all multiframe tasks in the system into AM tasks that have one critical frame, and then applying the exact response time formula on the transformed system. The fourth approach depends on pre-calculation of an upper bound interference from higher priority MF tasks within the deadline of the analysed task.

Comparisons between the approaches are done in two steps: in the first step we compare the results of the approaches with the exact results having small systems with 5 or 10 MF tasks; where the exact analysis is tractable. In the second step, we evaluate the comparison between the approaches, for big systems with $20, 40, 80,$ *and* 100 tasks, without taking the exact results into account so the comparison is done according to the approach that provides the best results.

The contents of this chapter is presented as the following: the first section introduces the maximum approach and proves the safety of this approach. Similarly, second, third and fourth sections cover the reordering, complementary and max accumulations approaches. In Section 8.5, we discuss the covering order of the approaches in the context of scheduling sufficiency. Section 8.6 compares, by evaluations, all mentioned

approaches. Summary of the chapter is given in the last section.

8.1 Maximum Approach

The major principle of the intractability problem of analysing the response times of non-AM multiframe tasks for big systems is the problem of analysing all simultaneous releases of all frames of the MF tasks. So, the first way to think of solving this intractability problem is to substitute the execution times of each multiframe task by its maximum execution time and then apply the basic original response time scheduling analysis[1] on the substituted tasks. We call the substituted task in this model *the maximum approximation*; where its period and deadline are identical to those of the original MF tasks while its execution time is constant and equals the maximum execution time of the original MF task. In other words, given a multiframe task τ_j having n_j frames with execution times (i.e. $C_j^k; k = 0..n_j - 1$); the **maximum approximation of τ_j is:** a task $\hat{\tau}_j$ that results by substituting τ_j's peak frame for all frames of τ_j. So, $\hat{\tau}_j$'s deadline and period are respectively $\hat{D}_j = D_j$ and $\hat{T}_j = T_j$ but the execution time, $\hat{C}_j$, is constant for all its jobs and equals to the maximum execution time of τ_j (i.e. $\hat{C}_j = C_j^{m_j}$)[2]. For example, the maximum approximation of the MF task τ_j whose execution times, deadline, and period are $< (3,7,4), 10, 15 >$ is the task $\hat{\tau}_j$ whose just mentioned attributes are $< 7, 10, 15 >$.

In the maximum approach, we transform all multiframe tasks in the system to their relative maximum approximations and then check the schedulability of the transformed system using basic response time test [40]. To be more accurate, checking the schedulability of a multiframe task relies on testing the schedulability of its peak frame assuming the maximum approximations for all higher priority MF tasks. The test assumes that having schedulable transformed system means that the original system is schedulable.

To consider the scheduling test using maximum approach as a sufficient scheduling test for a MF task, this approach has to be safe. The following theorem proves the

[1] We mean by the basic original response time scheduling analysis the response time analysis of the tasks whose execution times are constant for all of their jobs.

[2] Note that $C_j^{m_j}$ is the execution time of τ_j's peak frame.

safety of the maximum approach.

Theorem 12 *Given a system S with N multiframe tasks, $S = \{\tau_i;\ i = 1..N\}$. A multiframe task τ_i is definitely schedulable if its peak frame is schedulable using the maximum approach.*

Proof

The execution time of the maximum approximation is always greater than or equal to the execution times of the original MF task. In other words, $\hat{C}_j \geq C_j^l\ ;\ \forall l = 0,..,n_j - 1$. So, the cumulative functions of the maximum approximation is always greater than or equal to the cumulative functions of the original MF task for the same number of invocations and regardless of the releasing frame of the original MF task. Symbolically,

$$\hat{\xi_j}(k) \geq \xi_j^l(k)\ ;\forall k = 1,..,n_j, \forall l = 0,..,n_j - 1.$$

Therefore, the amount of interference the maximum approximation provides within lower priority task is always greater than or equal to the amount of interference the original MF task provides within this lower priority task; for each number of invocations (i.e. interference). So, the response time of the multiframe task τ_i under maximum approach is greater than or equals to the exact worst case response time of the original MF task (i.e. $\hat{R}_i \geq R_i$). Thus, having τ_i as a schedulable task under maximum approach means that it is exactly (i.e. definitely) schedulable.□

The following example illustrates the procedure of analysing a MF task using the maximum approach.

Example

Table 8.1 represents a simple numeric example system consisting of two MF tasks. To analyze the schedulability of τ_2 we will consider the maximum approximation of τ_1. Table 8.2 represents the attributes of the merged system using maximum approximations for the MF tasks whose priorities are higher than τ_2(i.e. τ_1)).

The response time of τ_2 using maximum approach is found by applying Equation (2.9) on the attributes in Table 8.2 which leads the response time being $17 < D_2$. As τ_2's response time under maximum approach meets the deadline, τ_2 is schedulable.

task	C	$T = D$
τ_1	$(1,6,1,1,2)$	10
τ_2	$(1,2,5)$	20

Table 8.1: Original Example System

task	C	T
$\hat{\tau}_1$	(6)	10
τ_2	$(1,2,5)$	20

Table 8.2: Transformed System

The exact response time of τ_2 according to the exact analysis given in Chapter 5 is 12. So, although τ_2's response according to the maximum approach is safe and easy to apply, it evaluates a very pessimistic response time. Pessimism of the maximum approach comes from the fact that the execution times of the maximum approximation could be hugely deviated from the real execution times of the original MF tasks. For example, the execution times of $\hat{\tau}_1$ in Table 8.2 has the deviations $(5,0,5,5,4)$ from each execution time of the original MF task τ_1. So the amount of interference that $\hat{\tau}_1$ generates when $\hat{\tau}_1$ provides four interference would be 24 while in reality the amount of interference that τ_1 generates for four interference is only 10 in the worst case, so there is a deviation of 14 from the real amount of interference. To reduce the deviation of the approximation from the real values of the execution times we introduce another schedulability test called the *Reordering approach*. The advantage of the maximum approach is however its ease of application.

8.2 Re-ordering Approach

Another way of solving the intractability problem of analysing response times of MF tasks is to safely transform the non-AM multiframe tasks into AM multiframe tasks that generate the same or greater amount of interference within lower priority tasks. One way of performing this transformation is to transform the MF task τ_j into its *re-ordering approximation* $\tilde{\tau}_j$ with a deadline and a period identical to τ_j's while its execution time sequence is a descended sequence of the execution times of τ_j; so the reordering approximation satisfies the AM restriction and therefore it has only one critical frame. For example, the execution time sequence of the re-ordering approximation of τ_j whose execution times are $(1,6,1,1,2)$ is $(6,2,1,1,1)$.

In the reordering approach, we transform all multiframe tasks in the system to their relative re-ordering approximations and then check the schedulability of the transformed system using the response time formula of the AM multiframe tasks (i.e. Equation (3.2)). To be more accurate, checking the schedulability of a MF task relies on testing the schedulability of its peak frame assuming the reordering approximations for all higher priority MF tasks. The test shows that having a schedulable transformed system means that the original system is schedulable.

As mentioned earlier, the schedulability test using reordering approach must be definitely safe to be considered, the following theorem proves the safety of the reordering approach.

Theorem 13 *Given a system S with N MF tasks, $S = \{\tau_j;\ j = 1..N\}$. Each multiframe task τ_j has n_j execution times. A lower priority multiframe task τ_i is definitely schedulable if it is schedulable assuming the re-ordering approximations for all multiframe tasks whose priorities are higher than τ_i's.*

Proof

For any arbitrary order of an execution time sequence of a multiframe task τ_j; the descending order of that sequence provides, for any number of invocations of τ_j, the maximum amount of interference on lower priority tasks. So, for any number of invocations of τ_j, the peak frame in the re-ordering approximation that is relative to τ_j generates amount of interference greater than or equal to the amount that the original τ_j generates. Therefore, the response time of a lower priority task τ_i under reordering approach is always greater than or equals to the exact worst case response time of τ_i due to the bigger amount of interference the reordering approximations of higher priority tasks provide. As a result, schedulability of τ_i using re-ordering approach means that its response time meets its deadline, therefore its exact response time is within its deadline and hence, τ_i is schedulable. □

The following example illustrates the procedure of analysing the response time of MF tasks using re-ordering approach.

Example

Table 8.3 represents the re-ordering approximation of the MF task τ_1 that is given in Table 8.1. To analyze the schedulability of τ_2, we will consider this approximation of the only MF task whose priority is higher than τ_2, then apply Equation (3.2) to the attributes in Table 8.3. So, the response time of τ_2 according to the re-ordering

task	C	T
$\tilde{\tau_1}$	$(6,2,1,1,1)$	10
τ_2	$(1,2,5)$	20

Table 8.3: Transformed System Using Re-ordering Approach

approach is 13 which is much closer to the exact response time of τ_2 than when using the maximum approach as explained in the previous section.

However, although re-ordering approach evaluates better response than the maximum approach, there are some situations in which the re-ordering approach evaluates a pessimistic response of the original MF task. For example, the execution time sequence of the reordering approximation, that is relative to the multiframe task whose execution times are $(1,10,1,1,1,8,4,1)$, is $(10,8,4,1,1,1,1,1)$. So, the amount of interference that the reordering approximation provides for two invocations is 18 while in reality the maximum amount of interference the original relative multiframe task provides for just two invocations is just 12. To think positively towards optimising the approach so it gives response time value closer to the exact one, we introduce another schedulability test called the *Complementary approach*.

8.3 Complementary Approach

The complementary approach is another way of solving the intractability problem of response time analysis of non-AM multiframe tasks by transforming the tasks into AM multiframe tasks. In this approach, we apply Mok and Chen [57]'s way of modelling a MF task to what we call the *complementary approximation*. All attributes of the complementary approximation are identical to the original MF task apart from the execution time values where they are derived from the original execution times as the

subtraction between each two consecutive maximums of interference that the original multiframe task provides. Symbolically, given a multiframe task τ_j having n_j execution times (i.e. $C_j^l; l = 0..n_j - 1$); its *complementary approximation* is the multiframe task $\overline{\tau_j}$ whose execution times $\overline{C_j^k}; k = 0..n_j - 1$ are derived from the execution times of τ_j according to the Formula (8.1).

$$\overline{C_j^k} = \max_{l=0..n_j-1} \{\xi_j^l(k+1)\} - \max_{l=0..n_j-1} \{\xi_j^l(k)\};\ \textit{where};\ k = 0..n_j - 1 \qquad (8.1)$$

The following example illustrates Formula (8.1), assume a multiframe task τ_j with the execution times $(1,10,1,1,1,3,3,1)$, τ_j could provide the sequence of maximum amounts of interference, regarding to the number of its invocations, as following $(10,11,12,15,18,19,20,21)$. So, the execution times of the complementary approximation, $\overline{C_j}$, is found by subtracting each two consecutive values in the former sequence assuming that $\max_{l=0..n_j-1} \{\xi_j^l(k)\} = 0$ when $k = 0$, and therefore $\overline{C_j} = (10,1,1,3,3,1,1,1)$. Note that $\overline{C_j}$ has only one critical frame that is the first frame whose execution time is 10 while the original multiframe task has three critical frames which are the one that is at position 1 where the execution time is 10, the one that is at position 5 where the execution time is 3 and the one that is at position 6 where the execution time is again 3. To explain more, the complementary approximation satisfies the AM restriction [57] so that is why it has only one critical frame.

The main idea of the complementary approach for testing the schedulability of a multiframe task τ_i is to check the schedulability of its peak frame assuming the complementary approximations for all higher priority multiframe tasks. So, if τ_i is schedulable under complementary approach then τ_i is definitely schedulable; while unschedulability of τ_i under complementary approach does not mean that τ_i is not schedulable. However, to make certain that this approach is applicable to the scheduling tests so what we can argue it is safe, we have to prove the safety of this test. Although [57] proved the safety of the transformation to the complementary approximations, the following theorem proves the safety of the complementary approach within the response time scheduling context.

Theorem 14 *Given a system S with N multiframe tasks,* $S = \{\tau_j;\ j = 1..N\}$, *each*

multiframe task τ_j has n_j execution times. A lower priority multiframe task τ_i is definitely schedulable if it is schedulable under the complementary approach.

Proof

To start with, we investigate the amount of interference that the complementary approximation $\overline{\tau_j}$ generates, within the execution of the lower priority tasks, for f number of its invocations, then we find out what this amount is equivalent to. The execution times of $\overline{\tau_j}$ are given by Equation (8.1), so the amount of interference $\overline{\tau_j}$ generates is given by the following function:

$$\sum_{k=0}^{f-1}(\max_{l=0..n_j-1}\{\xi_j^l(k+1)\}-\max_{l=0..n_j-1}\{\xi_j^l(k)\})$$

which is equal to

$\max_l\{\xi_j^l(1)\}$
$+\ \max_l\{\xi_j^l(2)\}\ -\ \max_l\{\xi_j^l(1)\}$
$+\ \max_l\{\xi_j^l(3)\}\ -\ \max_l\{\xi_j^l(2)\}$
$+\ .$
$+\ .$
$+\ \max_l\{\xi_j^l(f)\}\ -\ \max_l\{\xi_j^l(f-1)\}$
$$= \max_l\{\xi_j^l(f)\}$$

which is identical to the maximum amount of interference that τ_j generates for the same number of invocations f; which is given by the following cumulative function:

$$\max_{l=0..n_j-1}\{\xi_j^l(f)\}.$$

So, $\forall f = 1..n_j-1$, the maximum amount of interference that $\overline{\tau_j}$ generates is always equal to the maximum amount of interference that τ_j generates within the lower priority multiframe tasks. Therefore, considering $\overline{\tau_j}$ for all MF tasks whose priorities are higher than τ_j doesn't affect the schedulability of τ_j since the amount of interference from the higher priority tasks within τ_j is the same in both cases.□

Example

The following example explains the procedure of analysing the schedulability of a MF

task using complementary approach. To analyze the schedulability of τ_2 in the system in Table 8.4, we will consider the complementary approximation of τ_1, then apply Equation (3.2) to the attributes in Table 8.5. So, the response time of τ_2 assuming

task	C	$T = D$
τ_1	$(1,10,1,1,1,8,4,1)$	15
τ_2	$(1,2,6)$	20

Table 8.4: Example System

task	C	$T = D$
τ_1	$(10,2,1,1,10,1,1,1)$	15
τ_2	$(1,2,6)$	20

Table 8.5: Transformed System Using Complementary Approach

the complementary approximation of τ_1 is $18 \leq D_2$. As the response time meets τ_2's deadline τ_2 is definitely schedulable. Note that the exact response time of τ_2 is 17; which is less than estimated by the complementary approach.

As a matter of fact, the complementary approach is an equivalent approach to the one that was presented by Baruah et.al [13] in 1999. The difference between the two approaches is the way that each of them is presented.

8.4 Max Accumulations Approach

The previous three approaches (i.e. Maximum, Reordering, and Complementary approaches) were based on solving the intractability problem of response time scheduling of non-AM multiframe tasks by using transformation ways of converting the non-AM multiframe tasks into AM tasks. However, as we are only considering sufficient scheduling tests, here we consider an alternative means of constructing sufficient scheduling test.

This section introduces a straightforward approach that does not analyze any response times and does not need any transformation. The main idea of the presented approach is to pre-calculate the worst case expected interference within the deadline of the analysed multiframe task and then add this interference to its maximum execution time. If the calculated amount is less than or equal to the deadline then the

analysed task is schedulable. We call this way of testing the schedulability the *Max Accumulations Approach*.

Max accumulations approach is a simple way of testing the schedulability of MF tasks using off line calculations of the expected amount of interference within the deadline of the analysed task. We assume two aspects of this approach, the first aspect is the synchronous release of the analysed task and higher priority MF tasks. The second aspect is that, for the schedulable MF task, all MF tasks whose priorities are higher than the analysed MF task that are released within the deadline of the analysed task have finished their execution, within this deadline, with the maximum amount of interference they can provide.

To explain the procedure of the approach, we give the analysed MF task a virtual busy period; which is the execution time of its peak frame plus all interference from higher priority MF tasks within its deadline. So, the schedulability test is the following: τ_i is schedulable if its virtual busy period that is calculated by Equation (8.2) is less than or equal to its deadline.

$$C_i^{m_i} + \sum_{j=1}^{i-1} \max_{l=0,..n_j-1} \{\xi_j^l(\lceil \frac{D_i}{T_j} \rceil)\}. \tag{8.2}$$

Similar to the previous three approaches, the scheduling approach has to be safe to be accepted. The following theorem proves the safety of the max accumulations approach.

Theorem 15 *If a MF task is schedulable using max accumulations approach, it is definitely schedulable.*

Proof

Trivial, as the interference from higher priority MF tasks using max accumulations approach is greater than or equal to the exact interference from higher priority MF tasks. So, the virtual busy period of the analysed task is greater than or equal to its exact response time. Therefore, if the virtual busy period of the analysed task is less than or equal to its deadline, its exact response time is less than or equal to its deadline.□

For more clarifications, Algorithm 5 presents the pseudocode of calculating the virtual busy period that is given by Equation (8.2). *Max_Cum* in this algorithm is a non-square matrix and has N raws and maximum number of columns equals n; where $n = \max_{j=1,\ ..,\ N}\{n_j\}$. The value $Max\_Cum(j,k)$ represents the maximum cumulative function of the MF task τ_j for k number of its invocations. In other words,

$$Max\_Cum(j,k) = \max_{l=0,..n_j-1}\{\xi_j^l(k)\}. \tag{8.3}$$

The benefit of *Max_Cum* is to determine the term $\max_{l=0,..n_j-1}\{\xi_j^l(\lceil\frac{D_i}{T_j}\rceil)\}$ in Equation (8.2). However, Algorithm 5 is followed by a numeric example to illustrate the procedure of the max accumulations approach.

Algorithm 5 Finding Virtual Busy Period

Inputs: N: Number of Tasks, Task_Level, Execution_Times sequences .
Outputs: V_Busy_Period: Estimated amount of execution within D_i.

Max_Cum(j,k) $\Leftarrow$ matrix of $\max_{l=0,..n_j-1}\{\xi_j^l(k)\}$; $j = 1,\ ..,\ N$ and $k = 1,\ ..,\ n_j$
$V\_Busy\_Period \Leftarrow 0$
for j = 1 to Task_Level **do**
  $V\_Busy\_Period \Leftarrow V\_Busy\_Period + Max\_Cum(j, \lceil\frac{D_i}{T_j}\rceil)$
end for

Example

Assume the system that was previously given by Table 8.4 in the previous section. To test the schedulability of τ_2 we first find *Max_Cum* that is given by Algorithm 5; for $j = 1,2$, $k = 1,..n_j$. *Max_Cum* is found by applying Equation (8.3), so we get:

$$Max\_Cum = \begin{pmatrix} 10 & 12 & 13 & 14 & 24 & 25 & 26 & 27 \\ 6 & 8 & 9 & & & & & \end{pmatrix}.$$

Therfore, the virtual busy period of τ_2 (i.e. V_Busy_Period in Algorithm 5) is found

by applying Equation (8.2). In other words,

$$\begin{aligned} V\_Busy\_Period(2) &= C_2^2 + Max\_Cum(1, \lceil \frac{D_2}{T_1} \rceil) \\ &= 6 + Max\_Cum(1, \lceil \frac{20}{15} \rceil) \\ &= 6 + 12 \\ &= 18. \end{aligned}$$

As $V\_Busy\_Period(2) = 18 \leq D_2$ we say that τ_2 is schedulable.

8.5 Coverage of the Sufficient Approaches

Up to this point, we have covered four sufficient scheduling approaches but what we do not know about is the schedulability coverage of each of them. In other words, what is the order of the approaches in which if a task is schedulable using one approach it is definitely schedulable using followed approaches. In this section, we discuss the coverage order of the approaches depending on the amount of interference, from higher priority tasks, each approach estimates as the difference between the approaches is the estimation of this interference. As the first three approaches (i.e. the maximum, the reordering and the complementary approaches) use the same manner of using response time analysis but different ways of transforming MF tasks into AM multiframe tasks, we discuss the coverage order of these approaches and leave the coverage order of the max accumulations approach to be determined by the experiments.

As a matter of fact, the estimated interference from higher priority MF tasks under maximum approach is greater than or equal to the estimated interference from higher priority MF tasks under any of the other approaches. This is because in the maximum approach, the execution times of the higher priority MF tasks are estimated by their maximum execution times. So, if a task is schedulable under the maximum approach, it is definitely schedulable under any of the reordering or complementary approaches. In this sense, we say that the schedulability of a MF task using maximum approach is sufficient to determine the schedulability of this MF task using any of the other two approaches. Therefore, the maximum approach covers the other two approaches.

To determine the second approach after the maximum approach according to the coverage criterion, we compare the estimated interference from higher priority tasks under each of the reordering and complementary approaches. We already know that the descending order of a sequence of integers always provides equal or greater summation of any consecutive numbers than any other order of the original sequence. On the other hand, the reordering approximation transform the execution time sequence of the original MF task into a descent sequence. So, the cumulative functions of the reordering approximation $\tilde{\xi_j^l}(k)$ are greater than or equal to the cumulative functions of the original MF task $\xi_j^l(k)$. In other words,

$$\tilde{\xi_j^l}(k) \geq \xi_j^l(k);\ \forall l = 0,1,..,n_j-1,\ \forall k = 1,2,...$$

So,

$$\tilde{\xi_j^l}(k) \geq \max_{l=0,1,..,n_j-1} \xi_j^l(k);\ \forall k = 1,2,... \tag{8.4}$$

The right side of Equation (8.4) represents the estimation amount of interference from the complementary approximation $\overline{\tau_j}$ for k number of its invocations. So, the amount of interference from higher priority MF tasks under reordering approach is greater than or equal to the amount of interference from higher priority MF tasks under complementary approach. Therefore, if a task is schedulable using reordering approach, it is definitely schedulable under the complementary one. In this sense we say that the reordering approach covers the complementary approach.

As a result of the previous discussion, the coverage order of the approaches starts with the maximum approach followed by the reordering approach followed by the complementary approach. In addition, the ease of applying the tests goes in the same direction where the easiest is the maximum then the reordering then the complementary. One aim of the experiments is to determine if it is worthwhile to apply the more complicated tests.

8.6 Comparison Between Sufficient Scheduling Approaches

In this section, we compare by evaluation the previous mentioned approaches to consider the trade off between the ease of use against accuracy. In the comparison, we look at the scheduling performance each approach provides. Evaluations are done by generating random real-time systems.

The comparison is done, in summary, in two ways; one for small systems where the exact test is possible and another for large systems where the exact test is not possible. The system is considered as large (or big) when the experiments took more than one day to process the exact analysis of the system, using the departmental PC. This is because the exact analysis of a MF task must maximise its busy periods over all combinations of the critical frames of all higher priority MF tasks. So, the function of the worst case response time of a MF task is a polynomial function so the exact response time analysis is NP-hard. Hence, the analysis is intractable.

In the first set of experiments we find the percentage of the schedulable systems, for each approach, out of the exactly schedulable systems. The second way is based on finding the number of schedulable systems under each approach out of 10000 randomly generated systems. The following sections show the scope and algorithm of the experiments (i.e. choosing parameters and how each of the experiments is run) whilst the last section presents the results of the experiments.

8.6.1 Experimental Setup

Experiments in this chapter require the generation of real-time systems to check their schedulability under each approach and then compare them. The generation of a real-time system, in its turn, means the generation of the size of the system as well as the generation of the multiframe tasks that form the system. From the system size point of view, the exact experiments (i.e. experiments that take the exact test into account) are done for systems with 5 and 10 multiframe tasks because once the size of the system becomes greater than 10 multiframe tasks, the time of running the experiments

becomes too long. On the other hand, the non-exact experiments are done for systems with $5, 10, 20, 40, 80, 100$ multiframe tasks.

From the multiframe task's generation point of view, we require generating four parameters for each multiframe task, τ_i, (i.e. n_i, T_i, D_i, C_i; which are respectively: number of frames, Period, Deadline, and the execution time sequence). These four parameters of the MF task are generated similarly to what is done in Chapter 3 and 5 as the following. The number of frames of the multiframe task is assumed as fixed for all multiframe tasks in the system and it is chosen, for each experiment, as a prime number in the range $[3, 29]$. Choosing prime numbers for the number of frames is to follow similar scenario to what was introduced before in Chapters 3 and 5 so all parts of the thesis can be coherent and therefore, the results of the chapters can be compared to each other. Second and third parameters that are the period and deadline of the multiframe task are assumed to be identical to each other and are randomly generated in the range of $[1, 2500]$ using uniform distribution. Once the deadlines are assigned to each task, the priorities of the tasks are also assigned according to DM assignment; where the lower deadline the task has, the higher priority it is assigned [51].

The sequence of the execution times, which is the fourth parameter, is generated similarly to Chapter 3. Algorithm 2 illustrates the procedure of this generation; which uses UUnifast algorithm [20] that is illustrated by Algorithm 1. Further details can be found in Chapter 3.

8.6.2 Scope of Running the Experiments

We run the experiment 10000 times for each chosen parameters on five steps as following. Firstly, we generate the multiframe tasks by generating the parameters of the experiment (i.e. number of frames, periods, deadlines, and execution time sequence) as previously explained. Secondly, from the execution times, we find the critical frames of the generated multiframe tasks. Thirdly, we calculate the exact worst case response time of each task taking into account all critical frames of the higher priority multiframe tasks and then check if it is within its deadline. In other words, we check the schedulability of the system by checking the schedulability of all multiframe tasks

in this system. Fourthly, for the same parameters of the system we find the relative approximations of each approach and check the schedulability of the approximations. Lastly, for each approach and for small systems, we find the percentage of the schedulable systems out of the ones that are exactly schedulable; whilst for big systems we find the number of schedulable systems out of the 10000 generated systems.

For the small systems where the exact schedulability test is possible, we investigate all values of the utilisations within $(0.2, 0.3, 0.4, 0.5, 0.6)$ but we did not investigate the values that are less than 0.2 or greater than 0.6. That is because, for the most assumed number of frames, the number of the exact schedulable systems is very high and close to 100% when the overall utilisation of the system is below 0.2 as well as the number of the exact schedulable systems becomes very low and close to zero when the overall utilisation of the system goes beyond 0.6.

For the big systems where the exact schedulability test is not possible, we investigate the values of the utilisations that is in $[0.3, 0.5] = (0.3, 0.4, 0.5)$. That is because the range of U within $[0.3, 0.5]$ represents a converted range for most behaviours of the number of exactly schedulable systems; where the number of schedulable systems decreases within this range from around 100% to around only 10% (see Chapter 3 Figure 3.2) which gives importance to investigate the percentage of the improvement each approach gives.

8.6.3 Results of the Experiments

This section discusses the results of the experiments in two groups: the first group that is presented by Figures 8.1 - 8.6 considers the systems with 5 or 10 MF tasks where the systems are small enough to exactly test their schedulability. The second group that is presented by Figures 8.7 - 8.15 does not take the exact analysis into account as the systems are too big to exactly test their schedulability.

Figures 8.1 - 8.6 present the percentage of the number of schedulable systems for each of the four approaches (i.e. maximum, reordering, complementary, max accumulations approaches) out of the systems that are exactly schedulable. For more clarification for the results, those figures include the exact line which is the one hundred

percent of the exact schedulable systems. Results show that for small systems the closest approach to the exact one is the complementary one with different scheduling performance for all chosen parameters of the experiments; while the worst approach is always the maximum one, even when the overall utilisation of the system is very low 0.2 where the results of the approaches are so close to each other as in the last graph in Figure 8.1; where the systems have 10 MF tasks. Another example in Figure 8.1 is the first graph where it shows that, for systems with 5 MF tasks and number of frames is less than 19, more than 95% of the exactly schedulable systems are also schedulable by the four approaches; whilst this percentage decreases to about 91% using the maximum approaches when the number of frames increases to 23 frames. So, the complementary approach gives between 5% and 9% better performance than the maximum approach when the number of frames is between 19 and 23.

In addition, Figures 8.2, 8.3 and 8.4 show that when the systems have only 5 MF tasks the performance of the complementary approach becomes even better than the other approaches. For example, Figure 8.2 shows that when the overall utilisation of the system is 0.3 the complementary results is very close to the exact results. Also, this figure shows that when the overall utilisation of the system is 0.3 the complementary approach gives more than 95% schedulable systems for all chosen parameters; where its performance reduces from about 100% to 95% when the number of frames increases from 7 to 23; while at the same time the performance of the max accumulations approach, reordering approach, maximum approach reduce from 98%, 98%, and 83% respectively to about 82%, 58%, and 18% respectively when the number of frames increases from 7 to 23. So, the complementary approach gives ranges of $[2\%, 13\%]$, $[2\%, 37\%]$, and $[17\%, 77\%]$ better performance than the maximum accumulations approach, reordering approach and maximum approach when the number of frames increases from 7 to 23. Similarly, Figure 8.3 shows that when the overall utilisation of the system is 0.4 the complementary approach gives ranges of $[8\%, 19\%]$, $[5\%, 61\%]$, and $[45\%, 91\%]$ better performance than the maximum accumulations approach, reordering approach and maximum approach respectively when the number of frames increases from 5 to 23.

Using same argument, Figure 8.4 shows that when the overall utilisation of the system is 0.5 the complementary approach gives ranges of $[10\%, 22\%]$, $[1\%, 61\%]$, and

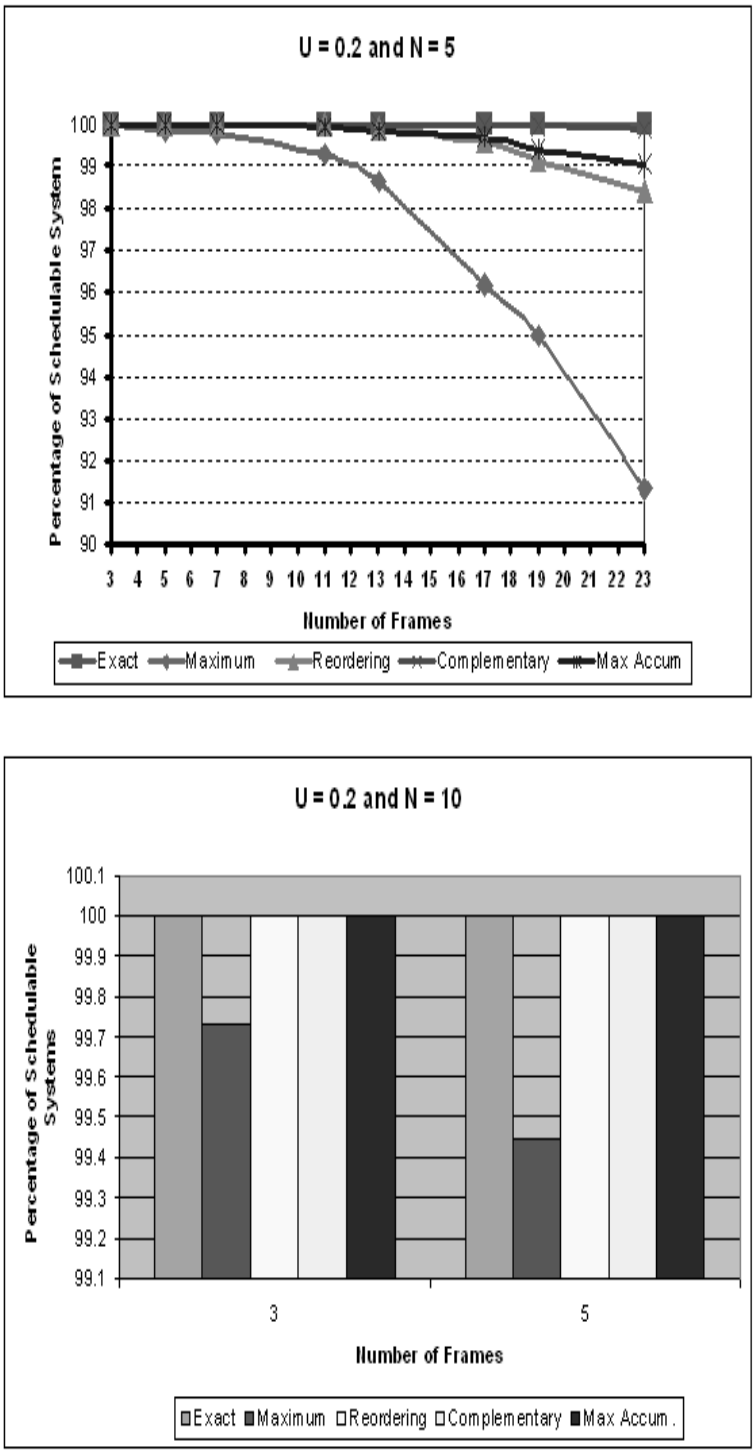


Figure 8.1: Percentage of Schedulable Systems $U = 0.2$ and $N = 5$ *and* 10

$[49\%, 90\%]$ better performance than the maximum accumulations approach, reordering approach and maximum approach when the number of frames increases from 3 to 19.

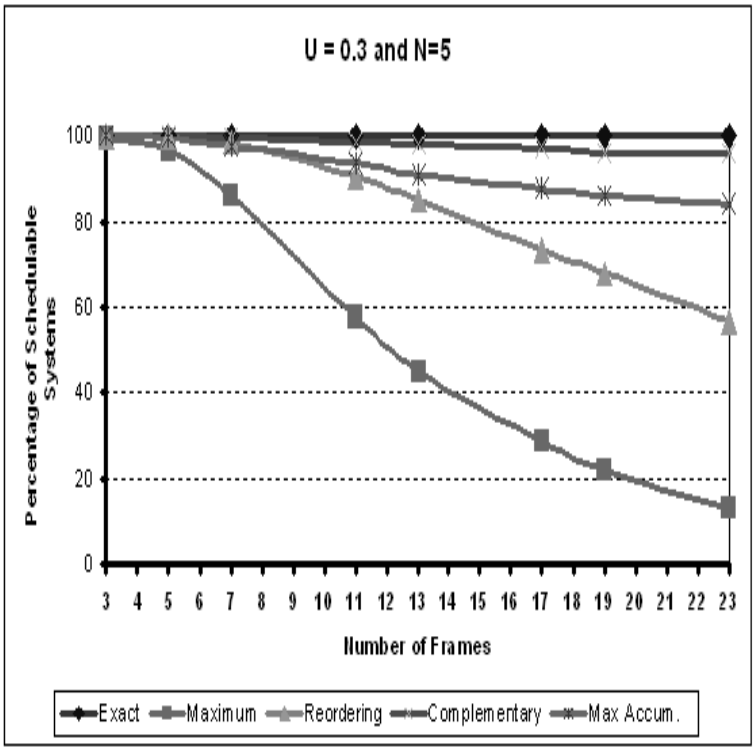


Figure 8.2: Percentage of Schedulable Systems $U = 0.3$ and $N = 5$

For more investigation, Figures 8.5 and 8.6 present the scheduling performance for systems with 10 MF tasks and from different overall utilisations point of view. The first graph in Figure 8.5 shows that for the systems with 10 MF tasks and 0.3 overall utilisation, the performance of all four approaches is very close to the performance of the exact analysis for MF tasks with 3 or 5 frames. However, Figures 8.5 and 8.6 show that when the overall utilisation of the system increases from 0.4 to 0.6 the complementary approach and reordering approach give similar performance when the number of frames is 3 whilst the complementary approach gives ranges of $[0\%, 21\%]$ and $[61\%, 97\%]$ better than the max accumulations and maximum approaches respectively. However, the performance of the reordering approach decreases to 60% when the number of frames becomes 5; whilst the complementary approach gives ranges of about $[0\%, 30\%]$, $[0\%, 29\%]$, and $[61\%, 89\%]$ better performance than the maximum accumulations approach, reordering approach and maximum approach when the overall utilisation increases from 0.4 to 0.6 and the number of frames is 5.

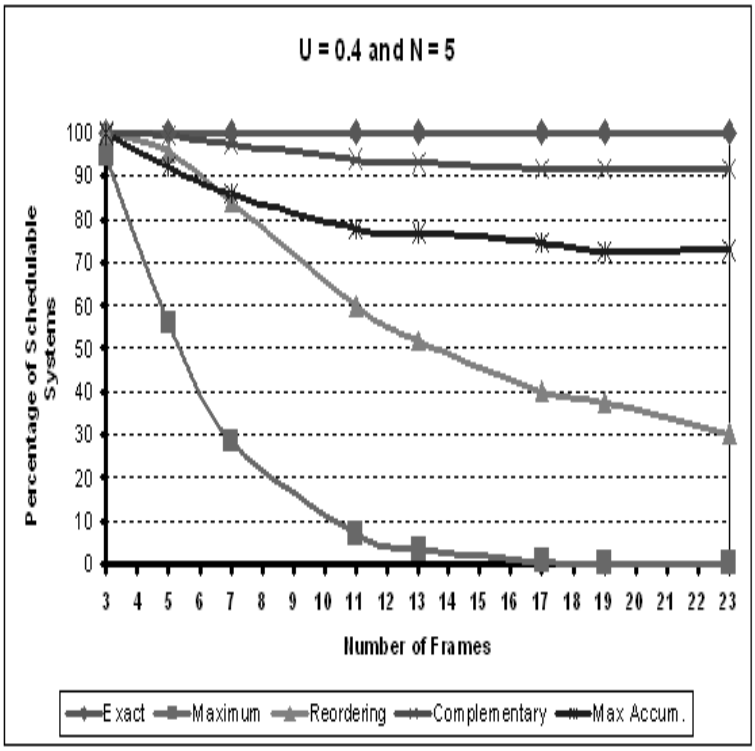


Figure 8.3: Percentage of Schedulable Systems When $U = 0.4$ and $N = 5$

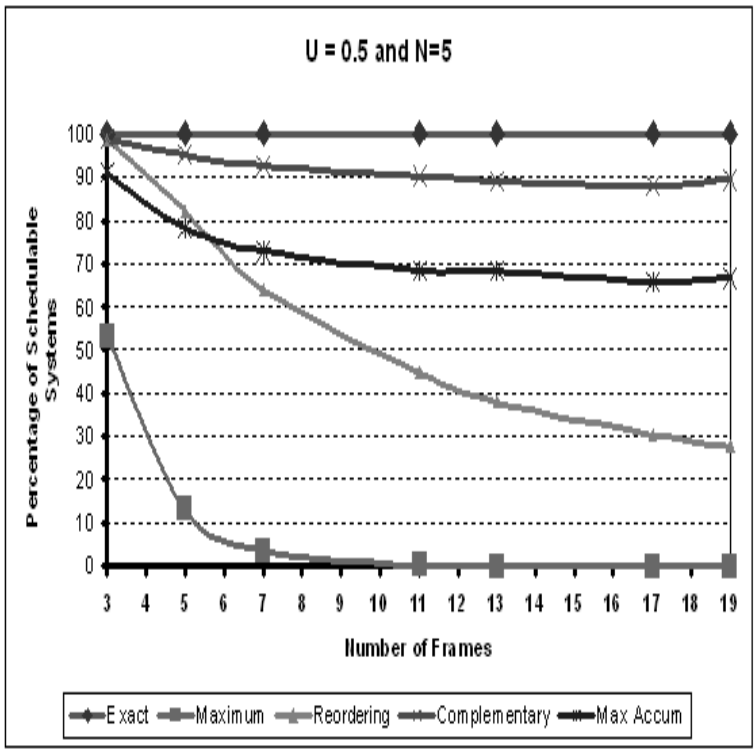


Figure 8.4: Percentage of Schedulable Systems When $U = 0.5$ and $N = 5$

On the other hand, for more coverage of the performance of the approaches, Figures 8.7 - 8.15 present the schedulability performance for big systems where the exact schedulability analysis is intractable. The schedulability performance in these figures is represented by the number of schedulable systems out of the 10000 randomly generated systems. All results show that the best approach for the big systems is the complementary approach while the worst one is the maximum one. For example, Figure 8.7 shows that for systems with an overall utilisation of 0.3, 10 MF tasks and number of frames is 29, there are 6400 schedulable systems out of the 10000 generated systems using complementary approach while there are 5500 schedulable systems out of the 10000 generated systems using max accumulations approach. So the complementary approach gives 9% (i.e. $\frac{6400-5500}{10000}\%$) better performance than the reordering approach for the mentioned parameters. Using similar argument Figure 8.7 also shows that the complementary approach provides ranges of $[0\%, 9\%]$, $[0\%, 42\%]$, and $[0\%, 64\%]$ better performance than the max accumulations, reordering and max-

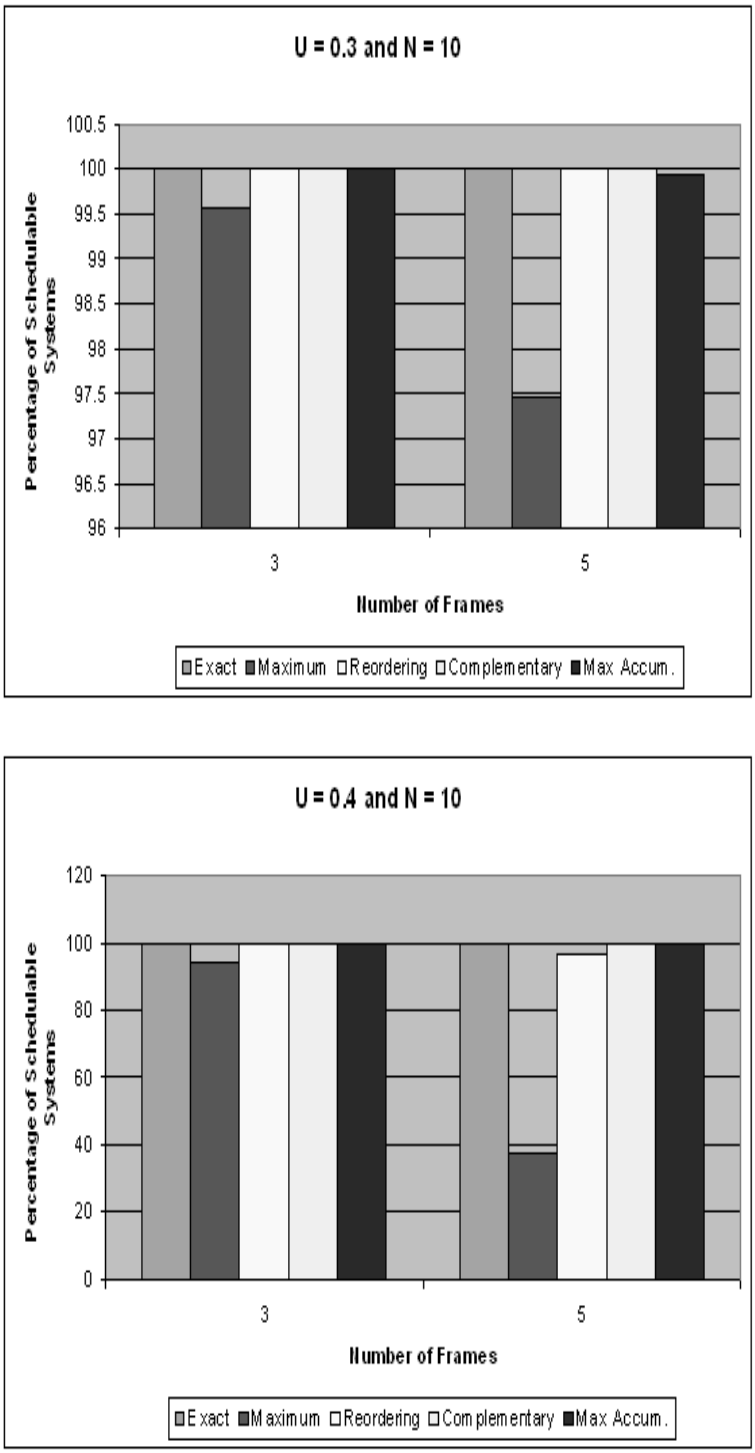


Figure 8.5: Percentage of Schedulable Systems When $N = 10$ and $U = 0.3$ *and* 0.4

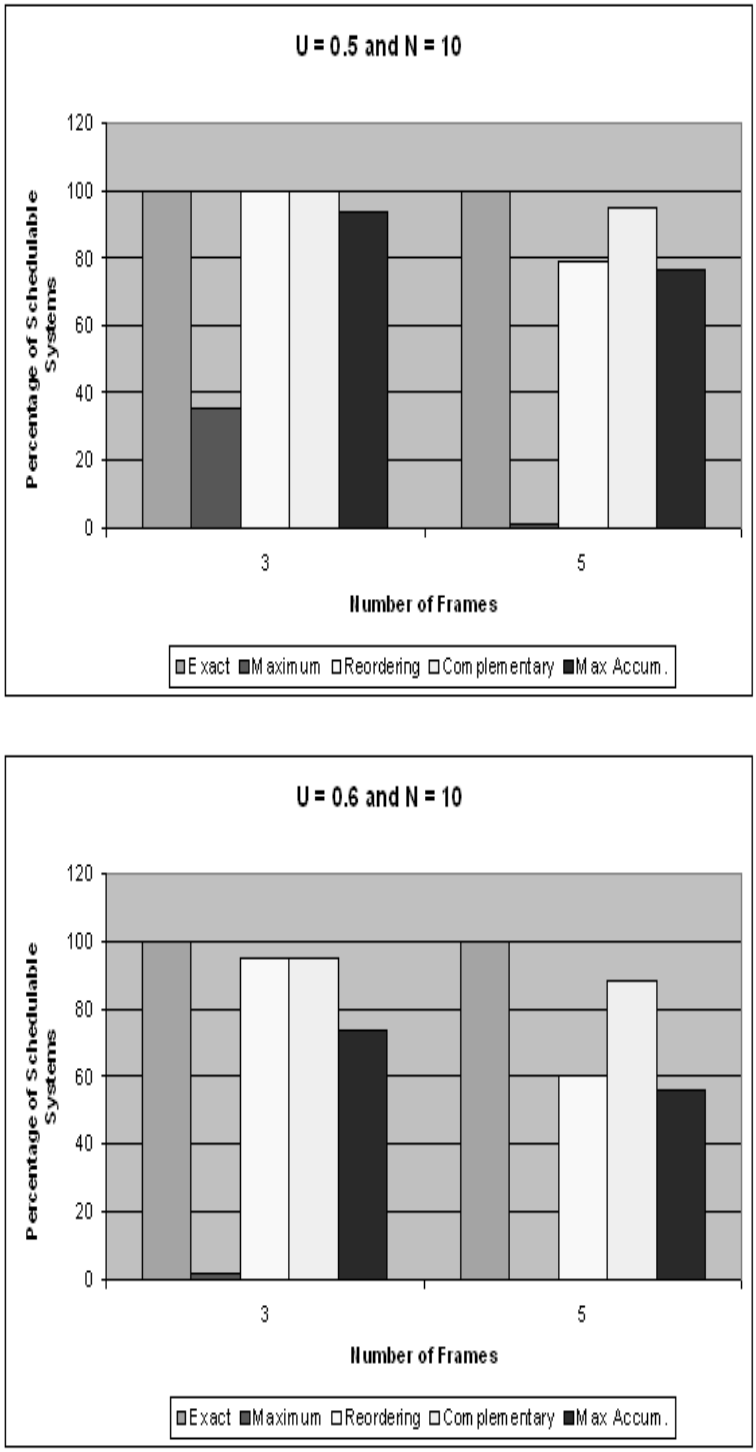


Figure 8.6: Percentage of Schedulable Systems When $N = 10$ and $U = 0.5$ *and* 0.6

imum approaches respectively when the overall utilisation is 0.3 and the number of frames increases from 3 to 29.

Similarly, Figure 8.8 shows that the complementary approach provides ranges of around $[0\%, 10\%]$, $[0\%, 26\%]$, and $[10\%, 45\%]$ better performance than the max accumulations, reordering and maximum approaches when the overall utilisation is 0.4 and the number of frames increases from 3 to 13. Moreover, Figure 8.8 shows that the complementary approach provides about 75% better performance than the maximum approach when the number of frames is 7.

However, Figure 8.9 shows that when the overall utilisation is 0.5 the performance of the complementary approach becomes lower with increasing the number of frames, although it provides better performance than the other approaches. For example, the number of schedulable systems under complementary approach decreases from about 1200 out of the 10000 generated systems to 0 when the number of frames increases from 11 to 19.

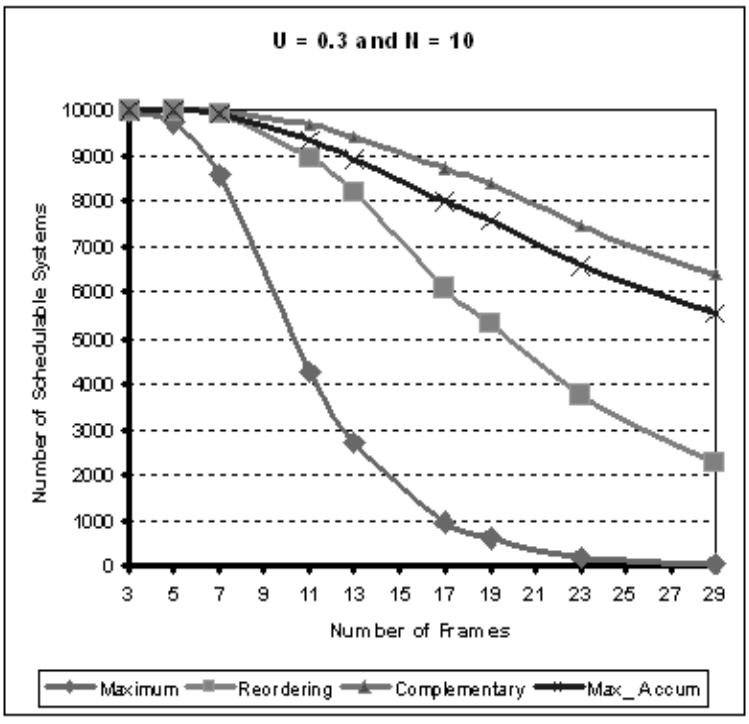


Figure 8.7: Number of Schedulable Systems When $N = 10$ and $U = 0.3$

In the following, we discuss the schedulability performance of the approaches when the number of MF tasks increases from 20 to 100 for each overall utilisation 0.3, 0.4, and 0.5. Figures 8.10 and 8.11 show that for both complementary approach and max accumulations approach, the greater number of MF tasks the system has the better performance the approach provides whilst the other way round with both maximum and reordering approaches. For example, for number of frames equals 29, the number of schedulable systems using complementary approach increases from 7000[3] to 8800[4] when number of MF tasks increases from 20 to 100. Also, the second graph in Figure 8.11 shows that the complementary approach gives ranges of $[0\%, 8\%]$, $[0\%, 70\%]$ and $[0\%, 78\%]$ better performance than the max accumulations approach, reordering approach and maximum approach respectively when the overall utilisation is 0.3, number of tasks is 40 and number of frames increases from 3 to 29. Moreover, Figure 8.11 shows that the maximum accumulations approach becomes very close to the comple-

[3]Found from the first graph in Figure 8.10.
[4]Found from the second graph in Figure 8.11.

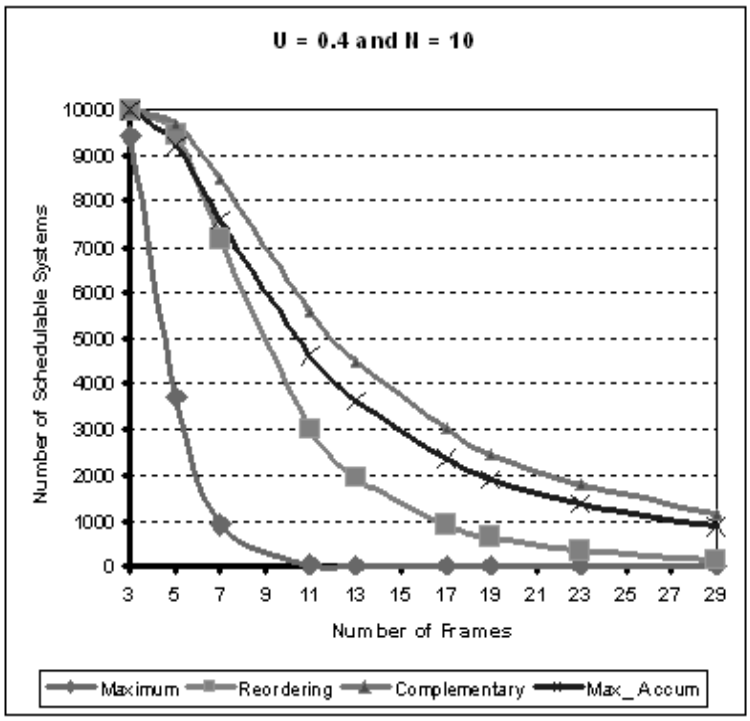


Figure 8.8: Number of Schedulable Systems When $N = 10$ and $U = 0.4$

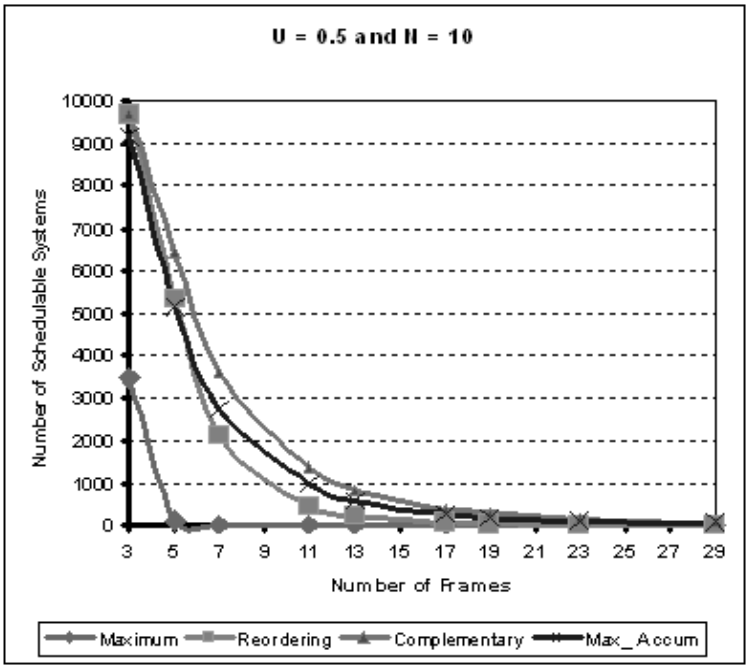


Figure 8.9: Number of Schedulable Systems When $N = 10$ and $U = 0.5$

mentary one when the number of tasks is 80 or 100 and their performance could be identical for number of frames less than 19.

From another point of view, Figures 8.12 and 8.13 show that when the overall utilisation of the system is 0.4 and number of frames are less than or equals to 13 the performance of the complementary and max accumulations approaches becomes better with increasing the number of MF tasks in the system from 20 to 100. For example, the number of schedulable systems under complementary and max accumulations approaches increase from 5900 and 5000 respectively when the number of frames is 11 and number of tasks is 20 in Figures 8.12 to 7000 and 6400 for the same number of frames and number of tasks is 100 in Figures 8.13. However, the performance of these two approaches reduces with increasing the number of frames beyond 13 and increasing the number of tasks from 20 to 100. For example, the number of schedulable systems under complementary and max accumulations approaches decrease from about 2100 and 1800 respectively when the number of frames is 19 and number of tasks is 20 in Figures 8.12 to 900 and 800 for the same number of frames and number

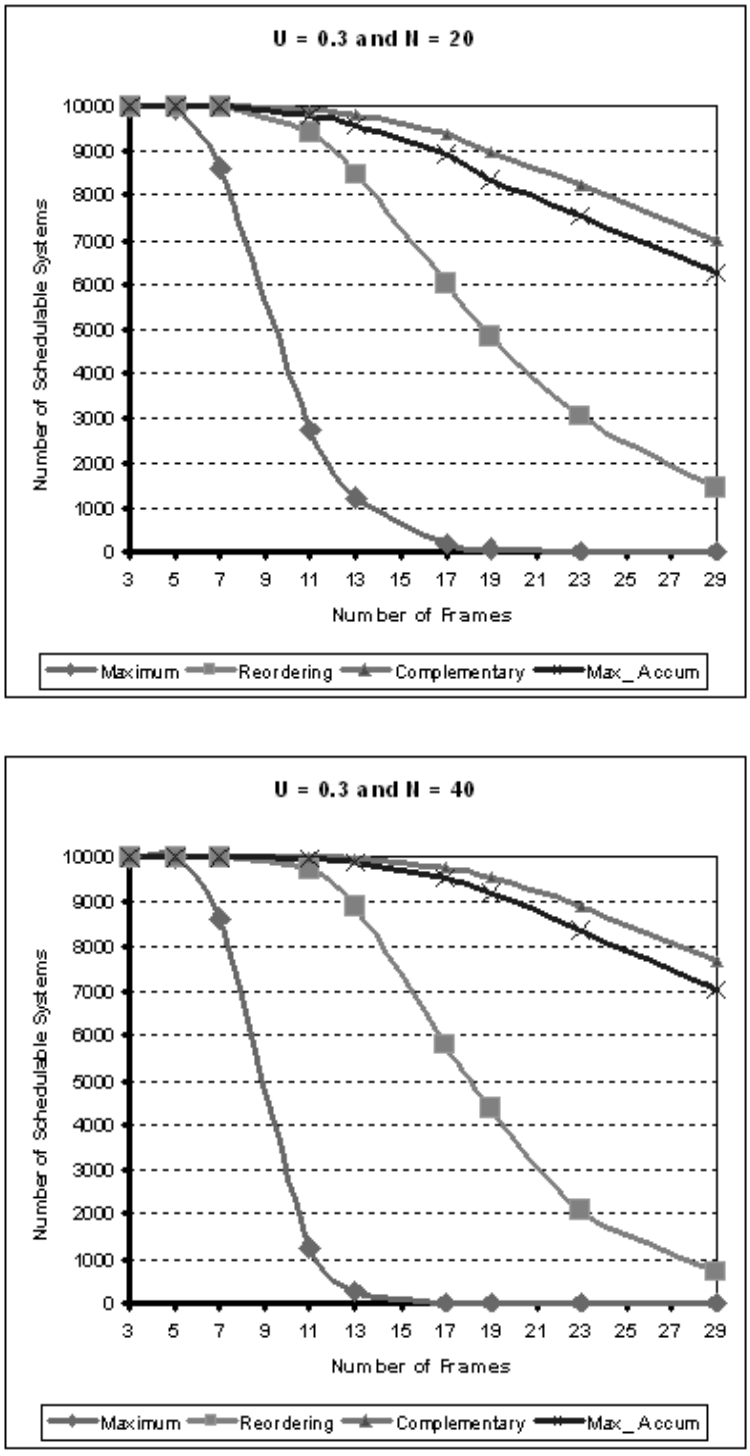


Figure 8.10: Number of Schedulable Systems When $N = 20$ *and* 40 and $U = 0.3$

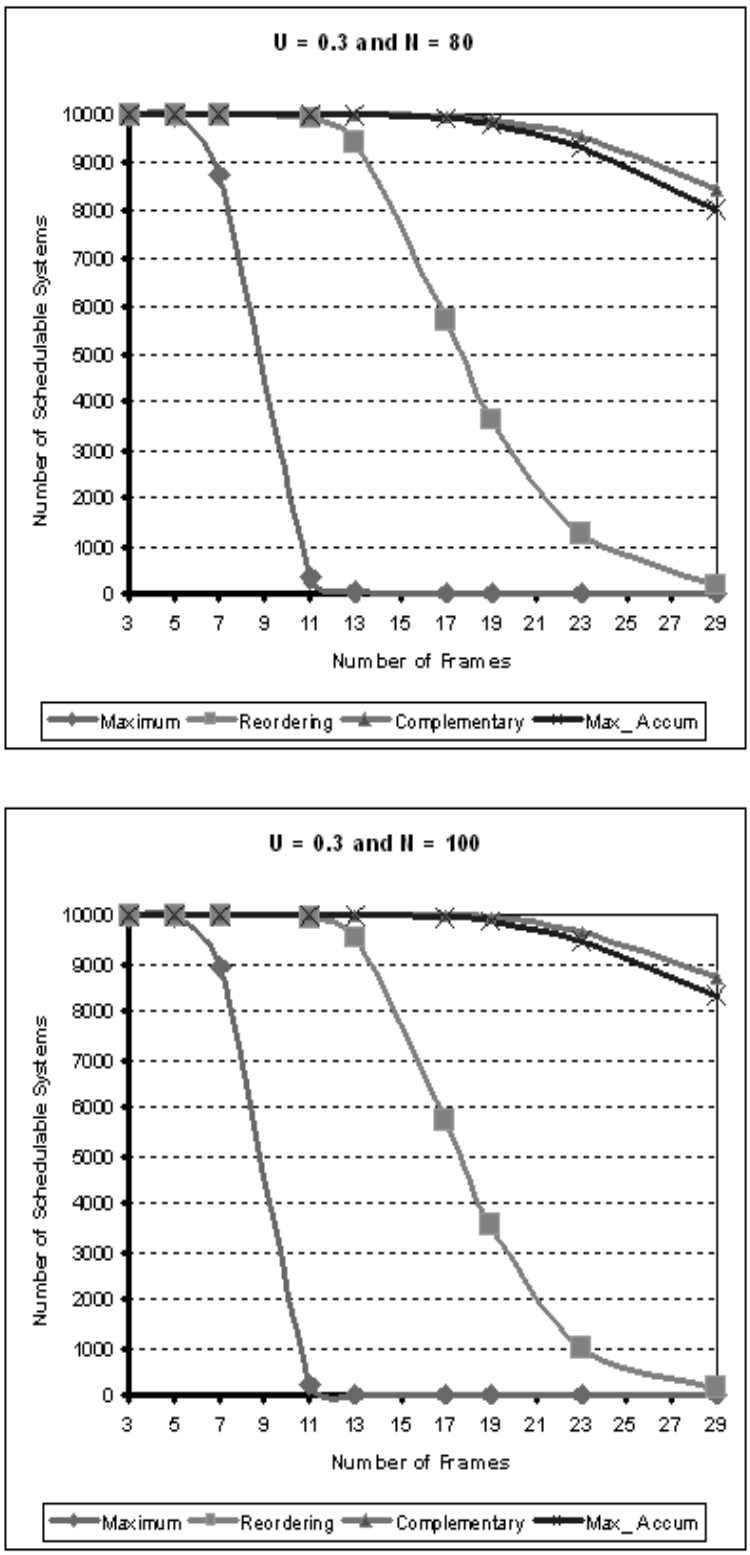


Figure 8.11: Number of Schedulable Systems When $N = 80$ *and* 100 and $U = 0.3$

of tasks is 100 in Figures 8.13.

In addition, Figures 8.12 shows that, when the overall utilisation is 0.4 and number of MF tasks is 40, there is a sharp decrease in the performance of the reordering approach when the number of frames increases from 5 to 11 where the number of schedulable systems decreases from 10000 to about 1500. At the same time the number of schedulable systems using complementary approach decreases from 10000 to about 6200. So the complementary approach provides 47% (i.e. $\frac{6200-1500}{10000}$%) better performance than the reordering approach for the mentioned parameters. Using similar argument, Figures 8.13 shows that the complementary approach gives ranges of $[0\%, 6\%]$, $[0\%, 46\%]$ and $[0\%, 100\%]$ better performance than the max accumulations approach, reordering approach and maximum approach respectively when the overall utilisation is 0.4, number of tasks is 100 and number of frames increases in the ranges $[3, 13]$, $[3, 13]$, and $[3, 5]$ respectively.

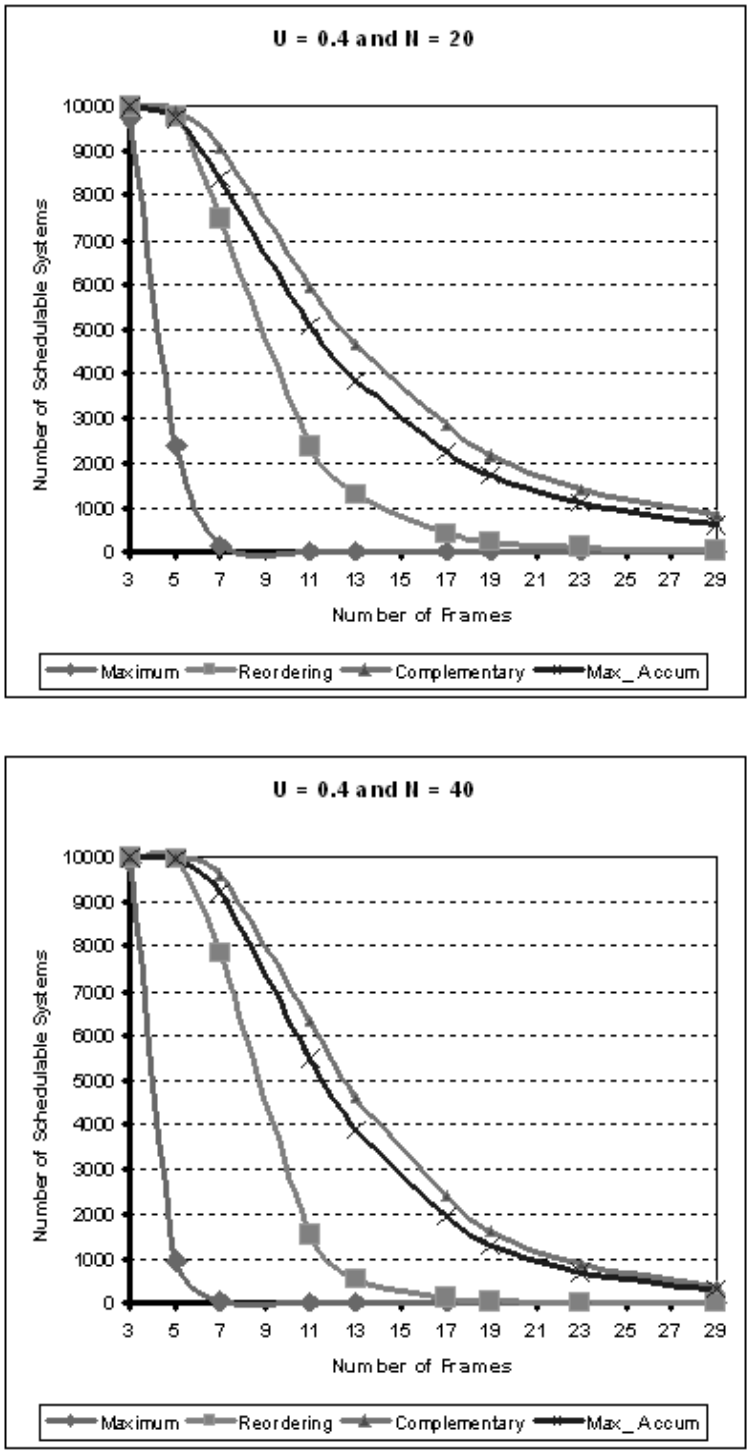


Figure 8.12: Number of Schedulable Systems When $N = 20$ *and* 40 and $U = 0.4$

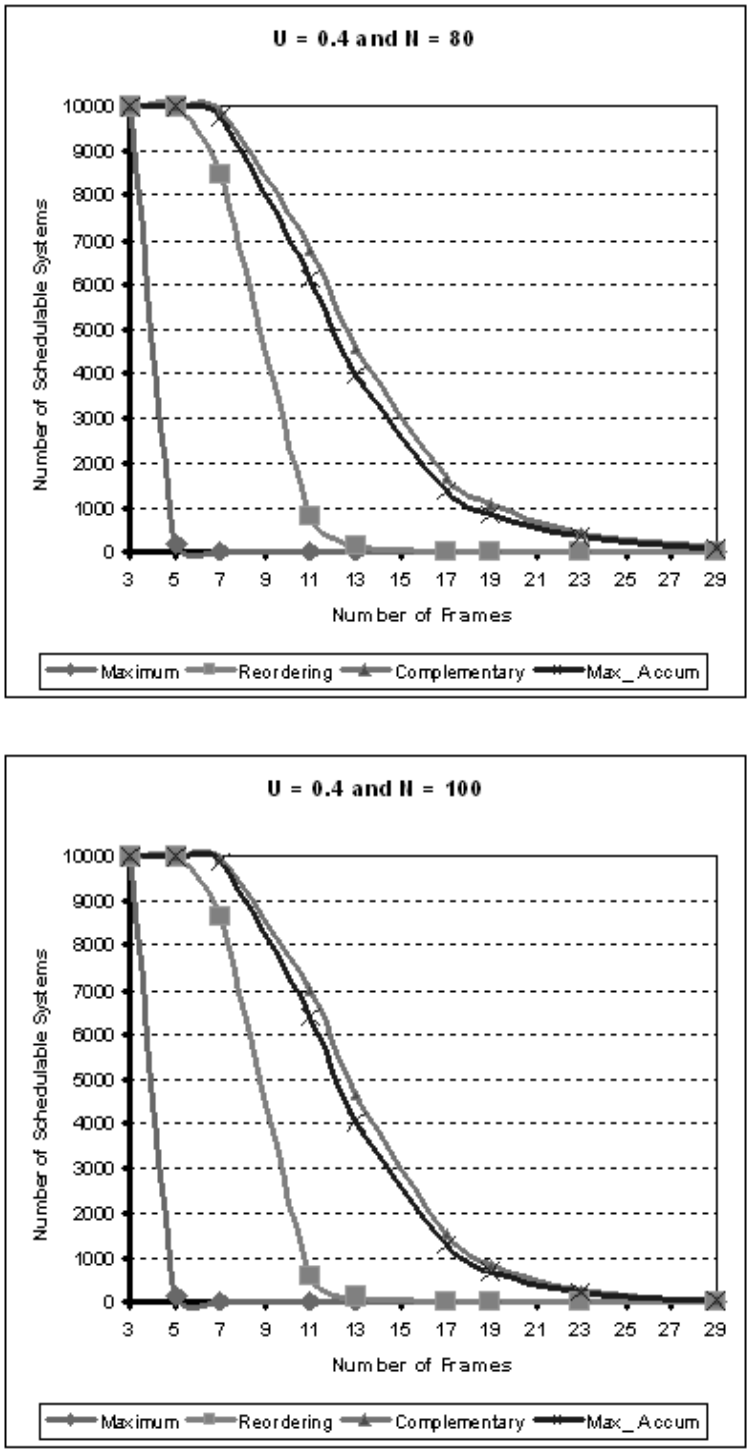


Figure 8.13: Number of Schedulable Systems When $N = 80$ *and* 100 and $U = 0.4$

For more investigation about the schedulability performance of the approaches, Figures 8.14 and 8.15 present the schedulability performance of the four approaches when the overall utilisation of the system is 0.5 and number of tasks is $20, 40, 80,$ *and* 100. Figures 8.14 and 8.15 show that although the performance of the complementary and max accumulations approaches becomes low when increasing the overall utilisation and size of the system, their performance is still better than the reordering and maximum approaches. Moreover, Figure 8.15 shows that each of the complementary, max accumulations, reordering approaches provides about 100% better performance than the maximum approach when the number of tasks is greater than 80 and number of frames is only 3.

8.7 Summary and Recommendations

This chapter has investigated by evaluation four sufficient scheduling approaches (i.e. Maximum, reordering, complementary, and max accumulations approaches) for relatively small and big systems. The main idea of the first three approaches was to safely transform the non-AM multiframe tasks to AM multiframe tasks and then apply the tractable response time analysis on the transformed system. On the other hand, the idea of the fourth approach (i.e. max accumulations approach) was to pre-calculate the maximum execution of both the analysed task and higher priority MF tasks; then check if this maximum execution can be achieved within the deadline of the analysed MF task to consider it as a schedulable MF task.

Results show that for all chosen parameters the best scheduling performance is generated by the complementary approach for both big and small systems so we classify the complementary approach as the best approach. This is because of two issues, the first issue is that the complementary approach provides the closest results to the exact one when the systems are small enough to exactly test their schedulability where the maximum differentiation between the exact performance and complementary performance was only about 9% showed by Figures 8.3, 8.4 and 8.6. The second issue, is that all results show that the complementary approach always provides better results than the other three approaches for both small and big systems.

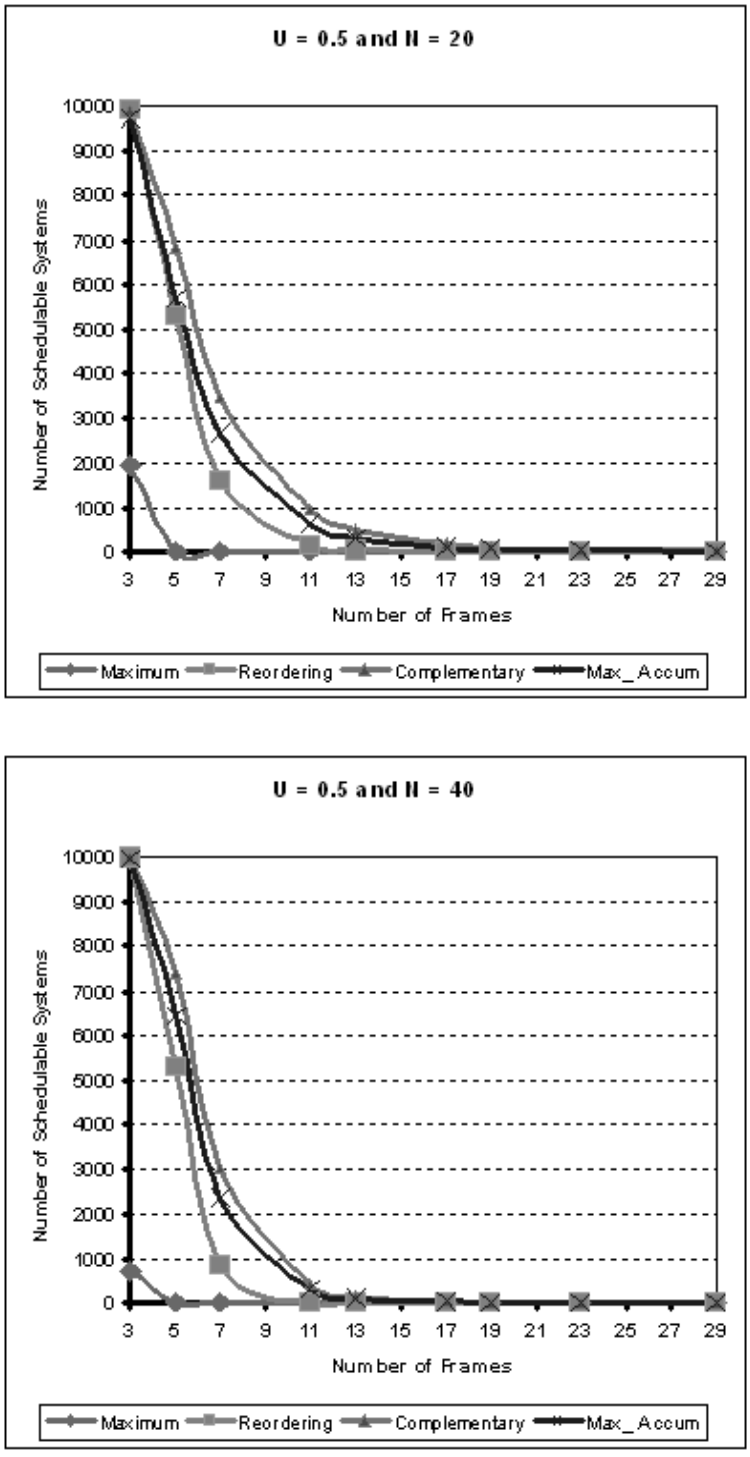


Figure 8.14: Number of Schedulable Systems When $N = 20$ *and* 40 and $U = 0.5$

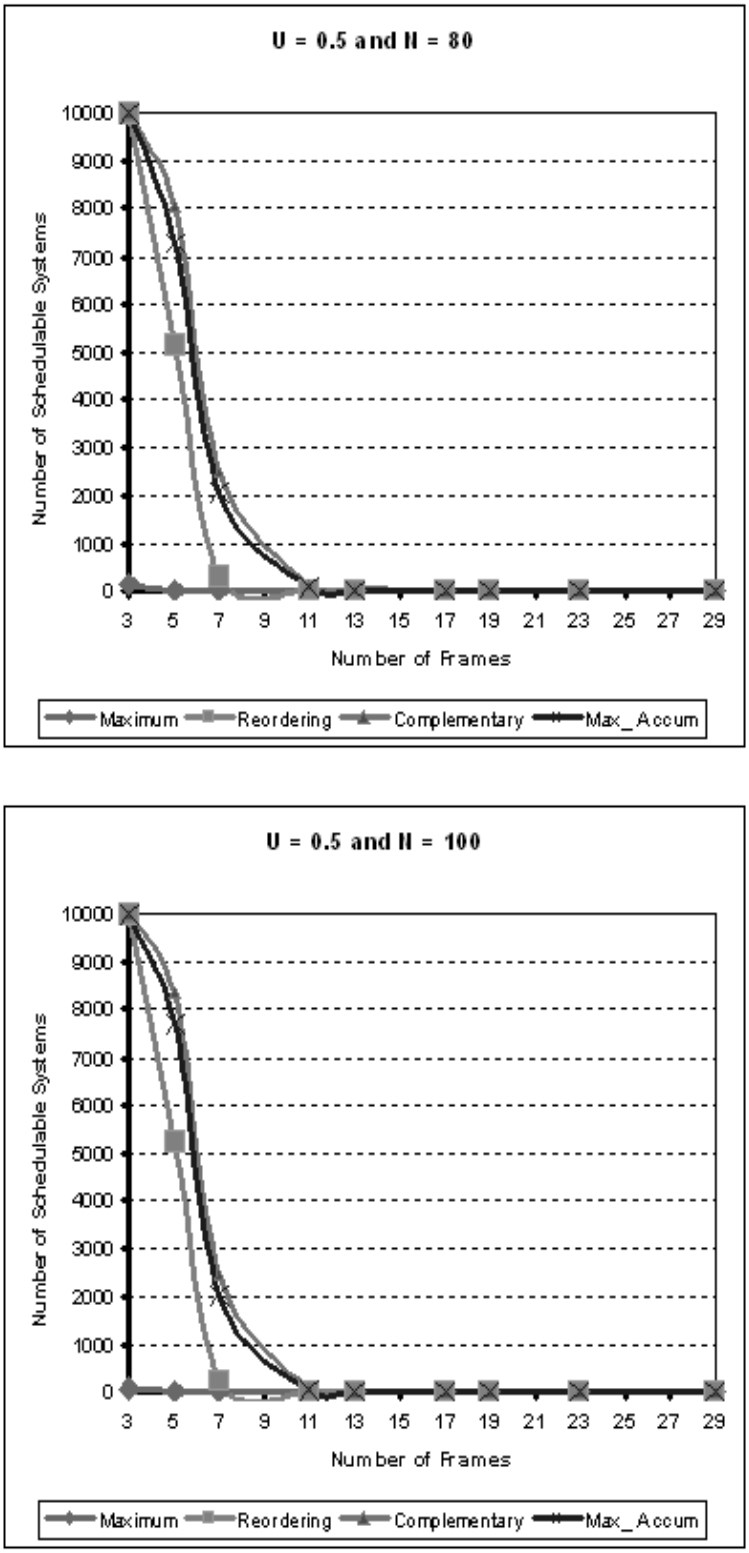


Figure 8.15: Number of Schedulable Systems When $N = 80$ *and* 100 and $U = 0.5$

Actually, we are interested in the best approach for big systems because when the system is small we can exactly analyze it without using any approximation approach. However, for big systems, the performance of the max accumulations approach becomes very close to the complementary approach where results show that in the worst case, the complementary approach provides a maximum better performance of only 8%. So, from the coverage point of view, we say that the max accumulations approach covers the complementary one; where if a task is schedulable using max accumulations approach, it is definitely schedulable using complementary one. On the other hand, the max accumulations approach is the easiest approach to perform where no need to do any recurrence calculations but we do need for the complementary approach. So, there is a trade off between the ease of use of the approaches and the accuracy of the results they provide, although all of them are safe. Therefore, to arrange the schedulability performance of the approaches, we identify the maximum accumulations approaches the second approach after the complementary one with a maximum differentiation rate between their performance of only 9%.

9 Evaluation, Conclusions and Future Work

This chapter summarises the contributions of this thesis and presents an overview of some ideas for future work. The main idea of the thesis is to exactly analyze the schedulability of hard real time systems with MF tasks. This has been done by analysing the response time of each MF task in the system. However, for big systems with general MF tasks; the response time analysis is not tractable so some sufficient approaches are introduced to test the schedulability of the systems that can not be exactly analysed.

9.1 Contributions of the Thesis

An exact response time scheduling test is an exact scheduling test in terms of being sufficient and necessary for hard real-time systems. This thesis has shown the flexibility of the response time analysis by analysing systems with MF tasks.

This thesis started to prove its claims when Chapter 3 presented a formula that calculates the worst case response time of basic MF tasks whose execution times are accumulatively monotonic (i.e. AM). This formula allows MF tasks to share resources, so it allows MF tasks to suffer blocking. To show the performance of the response time analysis, this chapter compares schedulability of the presented response time analysis with the most improved utilisation based scheduling test (i.e. Lu's test [55]). All results show a clear improvement in the schedulability performance using response time analysis rather than using Lu's test where the performance of response time test can be 100% better than Lu's test.

Chapter 4 has proved the flexibility of the response time analysis of MF tasks by extending the basic response time analysis, that is presented in Chapter 3, to two models more general than the basic model. In the first model, MF tasks are allowed to be subjected to release jitter. In the second model, the MF tasks are allowed to have arbitrary deadlines that could be greater than their relative periods, so the worst case response time analysis must include some amount of execution from the analysed task itself. Moreover, Chapter 4 has joined these two models and analysed the worst case response time of AM multiframe tasks that are subjected to both release jitter and arbitrary deadlines.

To further generalise the response time analysis, Chapter 5 has relaxed the AM restriction and analysed the worst case response time of non-AM multiframe tasks. Analysis in this chapter has used a new concept called a critical frame; where the analysis has considered only the synchronous releases of the critical frames of MF tasks whose priorities are higher than the analysed task. This chapter has shown that, in the worst case, a MF task with n frames could have $n-1$ critical frames. However, evaluation has shown that the number of critical frames per MF task is likely to be significantly less than $n-1$ and usually less than 65% of the original number of frames.

Chapter 6 has extended the response time of non-AM multiframe tasks to two models. The first model is when the MF tasks are subjected to release jitter and the second model is when the MF tasks have arbitrary deadlines. In addition, a combined analysis of the release jitter and arbitrary deadlines has been presented in this chapter.

For further proof of the flexibility of the response time analysis of MF tasks, Chapter 7 has presented an analysis of the worst case response time of MF tasks whose frames can have different deadlines; which has been called the frame specific deadlines scenario. A new concept called covering frame has been introduced in this chapter to optimise the number of frames that have to be analysed when the deadlines of the MF task are less than the relative period. However, general response time analysis has also been presented in this chapter when the deadlines of the MF task becomes greater than the relative period. As deadline monotonic priority assignment is not optimal anymore within the frame specific deadline scenario, another priority assignment for this model has been introduced in this chapter.

As the response time analysis is computationally intractable for relatively big systems, Chapter 8 has introduced four sufficient and computationally tractable approaches that can test the schedulability of big systems with non-AM multiframe tasks. These approaches are called maximum, reordering, complementary and max accumulations approaches. As the response time is tractable for AM multiframe tasks, three of these approaches have been based on transforming the non-AM multiframe tasks to AM tasks. Whilst the fourth one has been based on pre-calculations of the maximum invocations of higher priority MF tasks within the deadline of the analysed task. In this chapter the safety of these four approaches has been proved, coverage of the approaches has been explained, and a comparison between the approaches by evaluations has been presented. Results have shown the performance of the complementary approach comparing to each of the other approaches. Results have shown that although the best approach is the complementary one, its schedulability performance is very close to the performance of the max accumulations approach when the system is relatively big. This latter test is the easiest one to perform.

9.2 Future Work

Although this thesis has addressed some problems, there are some issues, related to what has been done in this thesis, are still open to be solved. The following is an overview of these open issues arranged according to the order of the chapters.

1. The analysis in this thesis considers that priorities of the MF tasks are assigned before performinging the analysis. However, a non covered issue in this thesis is to find an optimal priority assignment for the MF model and whether DM is optimal for this model.
2. The analysis in Chapters 3, 4, 5 and 6 can be improved to include variable blocking times instead of considering it as a single value.
3. In Chapter 5, the policy of identifying the critical frames could generate critical frames that can be safely discard from the response time analysis. Actually, this policy can be optimised to generate an optimal number of critical frames; where in reality there are frames in the generated critical frames who are dominated by

more than one other critical frames. This will involve using the critical frames that have been already generated in Chapter 5.

4. Solving exact response time equations, that is presented in Chapters 5, 6 and 7, requires a huge number of iterations to get either a stable solution or to identify the unschedulability of the MF task. To speed up the solutions of these exact response time equations we could start the solutions by calculating the minimum interference from higher priority MF tasks plus the maximum execution time of the analysed MF task instead of only the maximum execution time of the analysed task. This could be done by incorporating the work in [27] to serve as the system model in this thesis.

5. Moving on to the frame specific deadlines scenario, that is presented in Chapter 7, a number of issues arose within this chapter. The first issue is an improvement of the identification of the covering frames in the case of having arbitrary deadlines (i.e. Section 7.2) could be achieved using the cumulative function that is given by Definition 1. The second issue is to find a way that can avoid overlaps for solving $r_{i,\tilde{v},f}(C_i^q)$. In this issue we can consider two points: one is the minimum interference from higher priority MF tasks within the maximum execution of the analysed MF task and another is to make use of $r_{i,\tilde{v},f}(C_i^q)$ for a specific value of $f = f_1$ as a starting point to solve Equation (7.9) in Theorem 11. The third issue is an open problem that is still in progress; this problem is summarised by the following question: Is there a method that can optimise Corollary 5 so that there is no need to check all values of f? The fourth issue is to improve the analysis to include blocking time and release jitter.

9.3 Concluding Remarks

In the introduction, we claimed that the
"The schedulability of real-time systems with multiframe tasks can be exactly analysed using formulated response time analysis that is extensible to a wide variety of situations. Where response time analysis is intractable, appropriate non-optimal heuristics exist and allow all systems to be analysed."

This claim has been supported when three issues are considered. The first issue is the presentation of exact worst case response time formula for AM multiframe tasks and non-AM multiframe tasks, then extending these formulas to the situation where MF tasks suffer from blocking, release jitter and arbitrary deadlines. The second issue is the presentation of exact worst case response time formula for MF tasks whose deadlines are different from one frame to another. The third issue is the presentation of four sufficient and tractable scheduling approaches that can be applied to large systems.

List of References

[1] Available On line `http://www.vector.com/vi_oscan_en.html`, accessed in 07/01/2009.

[2] Available On line `http://shark.sssup.it/distrib/slides/2005_first_shark_workshop/handouts/FIRST_michael_handouts.pdf`, accessed in 07/01/2009.

[3] Available On line `http://www.lynuxworks.com`, accessed in 07/01/2009.

[4] Available On line `http://www.mathreference.com/lan-cx-np,vcp.html`, accessed in 07/01/2009.

[5] N. C. Audsley. Optimal priority assignment and feasibility of static priority tasks with arbitrary start time. Technical report, University of York, 1991.

[6] N. C. Audsley. *Flexible Scheduling of Hard Real-Time Systems*. PhD thesis, University of York, 1993.

[7] N. C. Audsley. On priority assignment in fixed priority scheduling. *Information Processing Letters*, (79):39–44, 2001.

[8] N. C. Audsley and A. Burns. On fixed priority scheduling, offsets and co-prime task periods. *Information Processing Letters*, pages 65–69, 1998.

[9] N. C. Audsley, A. Burns, M. Richardson, K. W. Tindell, and A. J. Wellings. Applying New Scheduling Theory to Static Priority Preemptive Scheduling. *Software Engineering Journal*, 8(5):284–292, September 1993.

[10] T. P. Baker. Stack-based scheduling of realtime processes. *Real Time Systems Journal*, 3(1):67–99, March 1991.

[11] N. Bala. Real time operating systems (rtos) modelling. Technical report, University of California, June 2004.

[12] S. K. Baruah, D. Chen, and A. Mok. Generalized multiframae tasks. *The International Journal Of Time Critical Computing Systems*, 17:5–22, 1999.

[13] S. K. Baruah, D. Chen, and A. Mok. Static-priority scheduling of multiframe tasks. In *proceedings* 11^{th} *Euromicro Conference on Real-Time Systems*, pages 38–45, June 1999.

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

[15] S. Baskiyar and N. Meghanathan. A survey of contemporary real time operating systems. *Informatica*, 29:233–240, June 2005.

[16] I. Bate. *Scheduling and Timing Analysis for Safety Critical Real-Time Systems*. PhD thesis, University of York, 1999.

[17] I. Bate and A. Burns. An integrated approach to scheduling, in safety-critical embedded control systems. *Real Time Systems*, pages 6–37, 2003.

[18] G. Bernat and A. Burns. New results on fixed priority aperiodic servers. In *20th IEEE Real Time Systems Symposium*, pages 68–78, 1-3 December 1999.

[19] E. Bini and S. K. Baruah. Efficient computation of response time bounds under fixed-priority scheduling. In *Real-Time and Network Systems (RTNS07)*, pages 95–104, March 2007.

[20] E. Bini and G. C. Buttazzo. Biasing effects in schedulability measures. In *16th Euromicro Conference on Real-Time Systems (ECRTS'04)*, pages 196–203, 2004.

[21] R. J. Bril, L. Steffens, and W. F. J. Verhaegh. Best-case response times of real-time tasks. In *Philips Workshop on Scheduling and Resource Management(SCHARM)*, pages 19–27, June 2001.

[22] R. J. Bril, W. F. J. Verhaegh, and E-J. D. Pol. Initial values for on-line response

time calculations. In *15th Euromicro Conference on Real-Time Systems*, pages 13–22, July 2003.

[23] I. Broster and A. Burns. An analysable bus guardian for event-triggered communication. In *24th IEEE International Real Time Systems Symposium*, pages 410–420, 3-5 December 2003.

[24] A. Burns, K. W. Tindell, and A. J. Wellings. Effictive analysis for engineering real-time fixed priority schedulers. *IEEE Transactions On Software Engineering*, pages 475–480, 1995.

[25] A. Burns and A. J. Wellings. *Real Time Systems and Programming Languages*. Addison-Wesley, England, 2001.

[26] A. Burns, A. J. Wellings, C.H. Forsyth, and C.M. Bailey. A performance analysis of a hard real-time system. *Control Engineering Practice*, 3(4):447–464, 1995.

[27] R. I. Davis and A. Burns. Response time upper bounds for fixed priority real-time systems. In *Real time Systems Symposium*, pages 407–418, December 2008.

[28] R.I. Davis, A. Zabos, and A. Burns. Efficient exact schedulability tests for fixed priority real-time systems. *IEEE Transactions on Computers*, 57(9):1261–1276, September 2008.

[29] F. Eisenbrand and Th. Rothvob. Static priority real-time scheduling response time computation is np-hard. In *Real time Systems Symposium*, pages 397–406, December 2008.

[30] N. Fisher and S. Baruah. A fully polynomial-time approximation scheme for feasibility analysis in static-priority systems with arbitrary relative deadlines. In *EuroMicro Conference on Real-Time Systems*, pages 117–126, 2005.

[31] N. Fisher and S. Baruah. A fully polynomial-time approximation scheme for feasibility analysis in static-priority systems with bounded relative deadlines. In *13th International Conference on Real-Time Systems*, pages 233–249, 2005.

[32] J. Goossens. *Scheduling of Hard Real Time Periodic Systems with Various Kinds of Deadline and Offset Constraints*. PhD thesis, University Liber de Bruxelles, Belgium, 1999.

[33] J. Goossens. Scheduling of offset free systems. *Real Time Systems*, vol.24:239–258, 2003.

[34] J. Goossens and R. Devillers. The non-optimality of the monotonic priority assignments for hard real-time offset free systems. *Real Time Systems*, vol.13:107–126, 1997.

[35] R. Grehan, R. Moote, and I. Cyliax. *Real-Time Programming, A Guide To 32-Bit Embedded Development*. Addison Wesley Longman, Inc, United States of America, 1998.

[36] J. C. P. Gutiérrez, J. J. G. Gracía, and M. G. Harbour. Best-case analysis for improving the worst-case schedulability test for distributed hard real-time systems. In *10th Euromicro Workshop on Real Time Systems*, pages 35–44, 17-19 June 1998.

[37] C. J. Han. A better polynomial-time schedulability test for real-time multiframe tasks. In *proceedings of 19th IEEE Real-Time Systems Symposium*, pages 104–113, December 1998.

[38] H. Hansson and M. Sjodin. Improved response-time analysis calculations. In *19 th IEEE Real-Time Systems Symposium*, pages 399–408, December 1998.

[39] M. Joseph. *Real Time Systems Specification, Verification and Analysis*. Prentice Hall Inc., 1st edition, 1996.

[40] M. Joseph and P. Pandya. Finding Response Times in a Real-Time system. *The Computer Journal*, 29(5):390–395, October 1986.

[41] D. Katcher, H. Arakawa, and J. Strosnider. Engineering and analysis of fixed priority schedulers. *IEEE Tran. Software Engineering*, 19:920–34, 1993.

[42] T. Kim, J. Lee, H. Shin, and N. Chang. Best-case response time analysis for improved schedulability analysis of distributed real-time tasks. In *ICDCS Workshop on Distributed Real-Time Systems*, pages B14–B20, 2000.

[43] D. E. Knuth. *The Art of Computer Programming*, volume 2:Seminumerical Algorithms. Addison-Wesley, 2ed edition, 1981.

[44] T.W. Kuo, L.P. Chang, Y.H. Liu, and K.J. Lin. Efficient on-line schedulability

tests for real-time systems. *IEEE Trans. on Software Engineering*, 29(8), 2003.

[45] J. Labetoulle. *Computer Architecture and Networks*. North-Holland, Amsterdam, 1974.

[46] J. J. Labrosse. *MicroC/OS-II The Real Time Kernel*. Miller Freeman, Inc, United States of America, 1999.

[47] B. W. Lampson and D. D.Redell. Experiences with processes and monitors in Mesa. *Communication ACM*, 23(2):105–117, Feb 1980.

[48] J. lehoczky, L. Sha, and Y. Ding. The Rate Monotonic Schduling Algorithm:Exact Characterization and Average Case Behaviour. In *IEEE Real Time Systems Symposium*, pages 166–171, 5-7 December 1989.

[49] J. P. Lehoczky. Fixed Priority Scheduling of Periodic Task Set with Arbitrary Deadlines. In *IEEE Real Time Systems Symposium*, number 11, pages 201–209, December 1990.

[50] J. Y. T. Leung and M.L. Merrill. A note on preemptive scheduling of periodic real time tasks. *Information Processing Letters*, 11(3), November 1980.

[51] J. Y. T. Leung and J. Whitehead. On the Complexity of Fixed Priority Scheduling of Periodic, Real Time Tasks. *Performance Evaluation*, 2(4):237–250, December 1982.

[52] C. L. Liu and J. W. Layland. Scheduling Algorithm for Multiprogrammimg in a Hard Real-Time Environment. *Journal of the Association for Computing Machinery*, 20(1):46–61, January 1973.

[53] J. W. S. Liu. *Real Time Systems*. Prentice Hall, United States of America, 2000.

[54] C. D. Locke. Software architecture for hard real-time applications: Cyclic executives vs fixed priority executives. *Real Time Systems*, 4(1):37–53, March 1992.

[55] W. C. Lu, K. J. Lin, H. W. Wei, and W. K. Shih. New schedulability conditions for real-time multiframe tasks. In *19th Euromicro Conference on Real Time Systems, (ECRTS07)*, Pisa, Italy, July 4-6 2007.

[56] A. K. Mok and D. Chen. A multiframe model for real time tasks. In *proceedings*

of IEEE International Real Time System Symposium, pages 22–29, December 1996.

[57] A. K. Mok and D. Chen. A multiframe model for real-time tasks. *IEEE Trans. on Software Engineering*, 23(10):635–645, Oct 1997.

[58] K. Paul and JR. Harter. Response Times in Level-Structured Systems. *ACM Transactions on Computer Systems*, 5(3), 1987.

[59] J. L. Peterson and A. Silberschatz. *Operating System Concepts*. Addison-Wesley Publishing Company, United States of america, 1983.

[60] M. Pilling, A. Burns, and K.Raymond. Formal specification and proofs of inheritance protocols for real-time scheduling. *Software Engineering Journal*, 5(5):263–279, 1990.

[61] R. Rajkumar. Real time synchronisation protocols for shared memory multuprocessors. In *10th IEEE Int.conf. on Distributed Computing Systems*, 28 May-1June 1990.

[62] P. Richard and J. Goossens. Approximate feasibility analysis and response-time bounds of static-priority tasks with release jitters. In *Real-Time and Network Systems (RTNS07)*, pages 105–112, March 2007.

[63] D. L. Ripps. *An Implementation Guide To Real-Time Programming*. Prentice Hall, Inc, United States of America, 1990.

[64] M. A. Rivas and M. G. Harbour. Marte os: An ada kernel for real-time embedded applications. In *International Conference on Reliable Software Technologies, Ada-Europe*, May 2001.

[65] O. Serlin. Scheduling of time critical processes. In *AFIPS Spring Computing Conference*, 1972.

[66] L. Sha, R. Rajkumar, and J. P. Lehoczky. Priority Inheritance Protocol: An Approach to Real Time Synchronization. *IEEE Transactions on Computers*, 39(9):1175–1185, Sept 1990.

[67] A. Silberschatz, P. Galvin, and G.Gagne. *Applied Operating System Concepts*. John Wiley & Sons, Inc, New York, United States of America, 2000.

[68] A. Silberschatz, P. B. Galvin, and G. Gagne. *Operating System Concepts*. John Wiley & Sons,Inc, United States of America, 2002.

[69] H. Takada and K. Sakamura. Schedulability of generalized multiframe task sets under static priority assignment. In *Real Time Computing Systems and Applications*, pages 80–86, 1997.

[70] K. W. Tindell. *Fixed Priority Scheduling Of Hard Real-Time Systems*. PhD thesis, University of York, 1993.

[71] K. W. Tindell, A. Burns, and A. J. Wellings. An extendible approach for analyzing fixed priority hard real-time tasks. *Real-Time Systems*, 6:133–151, 1994.

[72] K. W. Tindell and J. Clark. Holistic schedulability analysis for distributed hard real-time systems. *Microprocessing and Microprogramming*, 40:117–134, 1994.

[73] K. Traore, E. Grolleau, A. Rahni, and M. Richard. Response-time analysis of tasks with offsets. In *11th IEEE International Conference on Emerging Technologies and Factory Automation, (ETFA 2006)*, Prague, Czech Republic, September 2006.

[74] C. Y. Yang, J. J. Chen, and T. W. Kuo. Efficient on-line scheduling for energy minimization of multiframe real-time tasks on a dynamic voltage scaling processor. In *IEEE Real Time Systems Symposium*, number 29, pages 17–20, December 2008. Work In Progress.

[75] A. Zuhily. Exact response time analysis for multiframe tasks. Technical Report YCS 410, University ofYork, 2007.

[76] A. Zuhily and A. Burns. Optimal (D-J)-monotonic priority assignment. *Information Processing Letters*, 103(6):247–250, April 2007.

[77] A. Zuhily and A. Burns. Exact response time scheduling analysis of accumulatively monotonic multiframe real time tasks. In *5th International Colloquium on Theoretical Aspects of Computing (ICTAC)*, pages 410–424, 2008.

[78] A. Zuhily and A. Burns. Exact scheduling analysis of accumulatively monotonic multiframe tasks subjected to release jitter and arbitrary deadlines. In *13th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 600–608, 2008.

[79] A. Zuhily and A. Burns. Exact scheduling analysis of non-accummulatively monotonic multiframe tasks. In *16th International Conference on Real-Time and Network Systems (RTNS)*, pages 67–76, 2008.

Appendix

Theorem 16 *The priority assignment scheme $(D-J)-$monotonic is an optimal priority scheme in the sense that if any task set, Q, is schedulable by priority scheme, W, it is also schedulable by $(D-J)-$monotonic priority ordering.*

Proof

To prove the optimality of $(D-J)-$*monotonic* priority assignment, the priorities of Q (as assigned by W) will be transformed until the ordering is $(D-J)-$*monotonic* while preserving schedulability. Let τ_i and τ_j be two tasks with successive priorities in Q such that under W: $P_i > P_j$ and $D_i - J_i > D_j - J_j$. If it is not possible to find tasks τ_i and τ_j with this property then the tasks are already in $(D-J)$ order. Define scheme W' to be identical to W except that tasks τ_i and τ_j are swapped. The schedulability of all tasks whose priorities are higher than τ_i or whose priorities are lower than τ_j are not affected by swapping the two tasks τ_i and τ_j. Moreover, the schedulability of task τ_j will also not be affected by the swap since it will have higher priority than before and therefore it will suffer less interference. It remains to prove that task τ_i is still schedulable under W'.

Let R_j^W be the response time of task τ_j under scheme W, and $R_j^{W'}$ be the response time of task τ_j under scheme W'. It can be seen that $R_j^W \leq D_j - J_j$ because τ_j is schedulable and in the worst case may not be released until time t = J_j. In addition, it is given that $D_j - J_j < D_i - J_i \leq D_i \leq T_i$. Therefore, task τ_i only interferes once during the execution of τ_j (under W). So, the worst case response time of task τ_j can be split, under scheme W into

$$R_j^W = C_j + C_i + \sum_{k \in S} \lceil \frac{R_j^W + J_k}{T_k} \rceil C_k, \tag{9.1}$$

where S is the set of tasks whose priorities are higher than τ_i under W (which is equal to the set of higher priority than τ_j under W'). Equation (9.1) can be rewritten as

$$R_j^W - C_j = C_i + \sum_{k \in S} \lceil \frac{R_j^W + J_k}{T_k} \rceil C_k. \tag{9.2}$$

The response time equation of the task τ_i under scheme W' is given by

$$R_i^{W'} = C_i + \sum_{k \in hp(i)} \lceil \frac{R_i^{W'} + J_k}{T_k} \rceil C_k.$$

Hence,

$$R_i^{W'} = C_i + \lceil \frac{R_i^{W'} + J_j}{T_j} \rceil C_j + \sum_{k \in S} \lceil \frac{R_i^{W'} + J_k}{T_k} \rceil C_k. \tag{9.3}$$

Assuming Lemma 1 (given below), R_j^W is a solution of this equation for $R_i^{W'}$; which means that $R_i^{W'} \leq R_j^W$.

On the other hand, we have that $R_j^W \leq D_j - J_j$ as well as $D_j - J_j < D_i - J_i$. Therefore,

$$R_i^{W'} < (D_i - J_i)$$

which implies that task τ_i is schedulable after swapping tasks τ_i and τ_j.

Now, priority scheme W' can be transformed to W'' by choosing two more tasks that are in the wrong order for $(D-J)-monotonic$, and swapping them. Each such swap preserves schedulability. Eventually, there will be no more tasks to swap and the priority ordering will be exactly $(D-J)-monotonic$. Hence, $(D-J)-monotonic$ is optimal. □

Lemma 7 *The response time of the task τ_j under scheme W, R_j^W, is a solution of Equation (9.3).*

Proof

The idea of the proof is to substitute R_j^W for $R_i^{W'}$ into the right side of Equation (9.3) then we get its left side. Therefore, we say that R_j^W satisfies Equation (9.3) :
$C_i + \lceil \frac{R_j^W + J_j}{T_j} \rceil C_j + \sum_{k \in S} \lceil \frac{R_j^W + J_k}{T_k} \rceil C_k$

$= R_j^W - C_j + \lceil \frac{R_j^W + J_j}{T_j} \rceil C_j$
$= R_j^W - C_j + C_j$
$= R_j^W$

This is because of Equation (9.2) and because task τ_j is schedulable under scheme W, so $R_j^W < D_j - J_j$; which means $R_j^W + J_j < D_j < T_j$. So, $\lceil \frac{R_j^W + J_j}{T_j} \rceil = 1$. $\square$