P-FRP Task Scheduling with Preemption Threshold

Jian (Denny) Lin
University of Houston - Clear Lake
Houston, Texas 77058, USA
LinJian@uhcl.edu

Albert M. K. Cheng*
University of Houston
Houston, Texas 77004, USA
cheng@cs.uh.edu

ABSTRACT

Abort-and-Restart model is used in Priority-based Functional Reactive Programming in which higher priority tasks can preempt lower priority tasks and the lower priority tasks are aborted and restarted after the higher priority tasks have finished execution. This paper discusses a potential of improving schedulability in P-FRP systems by using preemption threshold that it allows a task to only disable preemption of tasks up to a specified threshold priority. Also, a sufficient schedulability test condition is studied in this paper for P-FRP tasks using preemption threshold, which is a critical problem to be solved in order to explore the potential benefit.

CCS CONCEPTS

• **Software and its engineering → Real-time schedulability**;

KEYWORDS

functional reactive programming, real-time schuedling, fixed-Prioirty scheduling, preemption threshold

1 INTRODUCTION

Complexity of software is growing at an exponential rate and it adds a tremendous burden to software development in stages of analysis and design. Introduced in 1997, Functional Reactive Programming (FRP) [1] is a programming paradigm for reactive programming (asynchronous data flow programming) that aims to simplify cognitive overhead of the engineering of modern software. FRP combines the power of functional and reactive programming and it has been used to develop reactive systems such as robotics, gaming and animation applications. Recently, in order to support the development of real-time applications, Priority-based Functional Reactive Programming (P-FRP) [2] has been introduced. P-FRP contains properties inherited from FRP such as atomic execution, immutable data structures and stateless processing, as well as supports priority assignment for executing real-time tasks. In a P-FRP system, Abort-and-Restart (AR) model is used where higher priority tasks can preempt lower priority tasks, and the lower priority tasks are aborted and restarted after the higher priority tasks have completed execution. In the aspect of real-time scheduling, the AR model is significantly different from the classical preemptive model where preempted, lower priority tasks can continue to execute from where they are preempted.

It has shown that traditional real-time scheduling methods are not applicable to ensure the timely requirement of P-FRP tasks. In [3–5], it has been proved that both Rate-Monotonic (RM) and Deadline-Monotonic (DM) priority assignments are not optimal for general task-sets where RM assigns priorities based on tasks' arrival rate and DM assigns priorities based on tasks' relative deadline. For schedulability analysis, simulation-based methods are studied in [5–7] by using the Least Common Multiple (LCM) of tasks' periods to generate a schedule of the tasks' execution. Thus, the schedulability can be determined. A major problem of the simulation methods is that the LCM can be unacceptably large. In [8], Wong and Burns show that finding an exact (sufficient and necessary) schedulibity test is intractable , and thus a sufficient condition is developed in their work. Since preemptions add additional execution time to lower priority tasks and in some cases this additional cost is significant to the tasks to meet their deadline, alternative models are used to reduce the number of preemptions. Deferred Abort (DA) technique is used in [9] to combine preemptive and non-preemptive executions in order to improve schedulability. In the DA model, a task is divided into two regions: the first region is AR and the second region is non-preemptive and non-abortable. Once execution of the task enters into the second region, preemption is not allowed and thus unnecessary preemptions may be avoided. Zou et al. [10] propose a non-work-conserving model, Deferred Start (DS), for P-FRP systems to reduce the number of preemptions by postponing the start of a job.

In the context of fixed-priority scheduling, Preemption Threshold (PT) [11] has been discussed extensively to improve schedulability. PT allows a task to only disable preemption of tasks up to a specified threshold priority. Tasks having priorities higher than the threshold are still allowed to preempt. This scheduling model offers opportunities to improve schedulability of P-FRP tasks because preemptions can be disallowed in some cases to reduce the overhead incurred by the AR model. Thus, some tasks' response time may be improved. Applying PT for P-FRP tasks consists of two sub-problems. One is a schedulability test for a set of P-FRP tasks with their priorities and preemption thresholds assigned. Another is selections of those tasks' priorities and preemption thresholds to optimize the performance. In this short paper, we solve the first sub-problem that is a required solution to solve the problem completely.

The rest of the paper is organized as follows. In Section 2, we discuss the task model and show a motivative example for how PT can be used to improve schedulability of P-FRP tasks. Section 3 presents a sufficient schedulability condition for the problem. Future works are discussed in the last section.

2 TASK MODEL AND MOTIVATIVE EXAMPLE

In this section, we first define the task model used in this paper. Then, we demonstrate a motivating example to show that using PT may greatly reduce response time of a P-FRP task.

*Supported in part by the U.S. National Science Foundation under Awards No. 0720856 and No. 1219082.

Table 1: Notations and Definitions

Notation	Definition
τ_i	Task i.
P_i	The minimum time interval between any two consecutive arrivals or period for τ_i.
C_i	Worst-case execution time of τ_i.
C_j^i	New used execution time of τ_j when calculating τ_i's response time.
D_i	Relative deadline of τ_i. $D_i \leq P_i$
α_i	Priority used before τ_i's execution.
β_i	Priority used in τ_i's execution (PT).
R_i	Worst-case response time of τ_i.
B_i	Blocking time contributed to calculate R_i
hep_i	Higher or equal to the priority of τ_i.
hp_i	Higher than the priority of τ_i.
lp_i	Lower than the priority of τ_i.
PB_i	Preempted By τ_i (Tasks that τ_i can preempt).

Table 2: A Set of 3 P-FRP Tasks with Preemption Threshold

Task	P	C	D	Priority	PT
τ_1	70	20	70	1	1
τ_2	100	30	100	2	2
τ_3	200	30	180	3	2

2.1 Task Model

Tasks considered in this paper are assumed to be sporadic such that each task is an infinite sequence of instances where the task is executed. Each task has a maximum arrival rate, defined by a minimum time interval (period) between any two consecutive arrivals of the instances. A task needs to complete its Worst-Case Execution Time (WCET) by its deadline. We consider to use relative deadline which means an instance of a task needs to be completed within a time interval relative to the instance's arrival time, or otherwise a real-time violation occurs. Without loss of generality, we assume that the period, execution time and relative deadline are all integers, and the relative deadline is less than or equal to the period. Each task has a priority assigned used for competing for CPU before it executes, and another priority used for PT. We use an integer of 1 to denote the highest priority of a task and we assume that a larger integer indicates a lower priority. Preemptions are allowed but managed by the two priorities for each task and it does not allow self-preemption to happen. The AR model is used by the lower priority tasks after each time they are preempted. Notations and the detailed definitions used in this work are defined in Table 1.

2.2 A Motivative Example

We consider to use the following example to demonstrate the potential of using PT to improve schedulability. There is a task set of three tasks as shown in Table 2. In these tasks, neither τ_1 nor τ_2 has a raised priority during their execution while the priority of τ_3 in its execution is increased to 2. In other words, after τ_3 starts to execute, only τ_1 can preempt it and τ_2 cannot do the same. Figure 1 shows a schedule of these three tasks without PT where τ_3 is preempted three times and each time it is aborted and restarted. As a result, τ_3 misses its deadline at time instant 180. Figure 2 shows another schedule that τ_3 can only be preempted by τ_1. Due to the reduced number of preemptions on τ_3, τ_3 completes its execution much earlier than that in the previous schedule. It is worth to note that in this schedule τ_3 delays the execution of the second instance of τ_2 although the instance completes execution by its deadline too.

From the example, it can be seen that by raising a task's priority during execution, the task's response time may be significantly reduced by a reduction on preemptions. On the other side, when it disallows preemptions on the task up to some threshold, it may postpone some other tasks' execution that may impacts the tasks to complete by their deadlines. Therefore, a careful selection of tasks' PT may make a set of P-FRP tasks from unschedulable to schedulable in a schedulability test. In the next section, we study a schedulability problem of a set of P-FRP real-time tasks with PT.

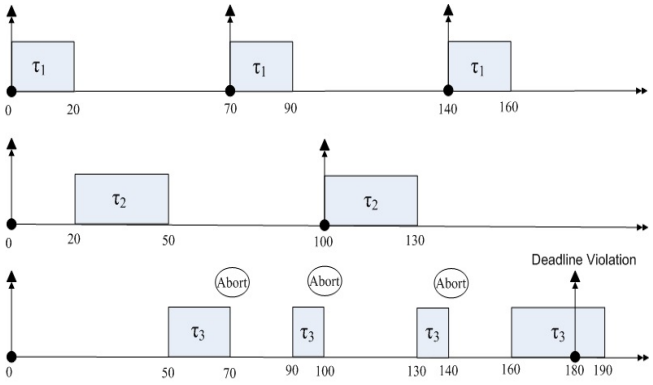

Figure 1: A Schedule without Preemption Threshold

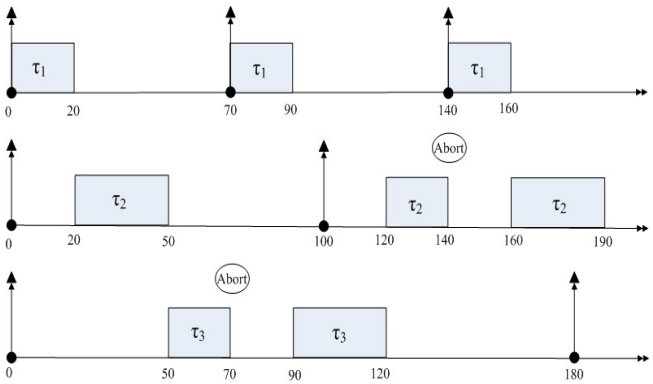

Figure 2: A Schedule with Preemption Threshold

Table 3: A Set of 4 P-FRP Tasks with New WCET C_j^4

Task	P_i	C_j	C_j^4	Priority
τ_1	15	2	7(2+5)	1
τ_2	25	3	8(3+5)	2
τ_3	45	4	9(4+5)	3
τ_4	100	5	5(5+0)	4

3 SCHEDULABILITY ANALYSIS WITH PREEMPTION THRESHOLD

Given a set of P-FRP tasks with PT, we perform the schedulability analysis by using the worst-case response time analysis of each task. If each task's worst-case response time is not larger than its relative deadline, all tasks are schedulable. The response time analysis used in this paper is an extension of the well-known time-demand analysis [12–15] in which the response time is calculated iteratively by adding new arrivals of tasks, starting from a critical instant. A critical instant is defined as a time instant when a task is ready together with all other tasks, the calculated response time of this task is the largest in all cases. In this section, we first introduce a method that is a sufficient condition for the schedulability problem of P-FRP tasks. Then, we extend this method with using PT.

3.1 A Sufficient Schedulability Test

The work in [8] has shown that finding the critical instant in an exact analysis is intractable. A difference between calculating a P-FRP task's response time and a general sporadic task's response time is the abort cost (the extra re-execution time of the task due to the AR model). This cost is the key to cause that all tasks starting at the same time is not a critical instant and the worst-case response time is not always associated with the first instance of a P-FRP task. To simplify the solution of the problem, an approach is proposed in [8] that a theoretically maximum abort cost is included in the execution time of all higher priority tasks of a task. Thus, the general iterative method using a critical instant to calculate a task's response time can be applied. Since the abort cost considered in that approach is only a theoretical upper-bound, the schedulability condition is only sufficient, not exact. We briefly introduce the approach as follows.

Let C_j^i be a new execution time of τ_j including abort cost when calculating the response time of τ_i. The following equation is used to calculate C_j^i.

$$C_j^i = C_j + \max_{\forall k \in hep_i \cap lp_j} C_k \tag{1}$$

Then, the worst-case response time of τ_i, R_i, can be calculated as:

$$R_i = C_i + \sum_{\forall j \in hp_i} \lceil \frac{R_i}{P_j} \rceil \times C_j^i \tag{2}$$

If R_i is not larger than D_i for every τ_i, the task set is schedulable. We show an example for how to use the approach. Table 3 shows the new execution times for all of the higher priority tasks than τ_4. These new WCETs are used to calculate R_4. In this example $R_4 > 100$ and hence τ_4 is not schedulable.

Table 4: A Set of 4 P-FRP Tasks with New WCET of $C_j^{PT_4}$

Task	P_i	C_j	$C_j^{PT_4}$	Priority	PT
T_1	15	2	7(2+5)	1	1
T_2	25	3	7(3+4)	2	2
T_3	45	4	4(4+0)	3	3
T_4	100	5	5(5+0)	4	2

We define another new execution time used in our later analysis. Let $C_j^{PT_i}$ be the new execution time of τ_j, when PT is used, to calculate τ_i's response time.

$$C_j^{PT_i} = C_j + \max_{\forall k \in hep_i \cap PB_j} C_k \tag{3}$$

CLAIM 1. *With using PT, replacing C_j^i in (2) with $C_j^{PT_i}$ is tighter for the new execution time of τ_j.*

Proof. The claim is true from an observation that between τ_j and τ_i, only if τ_j's priority is higher than τ_i's PT ($\tau_i \in PB_j$), τ_j can preempt τ_i and cause it to be aborted. Otherwise, τ_j does not abort τ_i. Thus, when calculating a new execution time of τ_j, if τ_j does not abort τ_i , it does not need to consider τ_i's abort cost. Since for some τ_js the equation (3) does not consider the WECT of some tasks that are considered by equation (1), $C_j^{PT_i} \leq C_j^i$. □

Table 4 shows the new computed $C_j^{PT_i}$ for the same task set in Table 3 where τ_4 has a PT of 2. Because both τ_2 and τ_3 cannot abort τ_4, τ_4's WECT is not used to calculate their new WCETs $C_j^{PT_i}$.

3.2 A Sufficient Test with Preemption Threshold

The previous motivative example shows that a higher priority task can be blocked by a lower priority task's PT for execution until the lower one completes its execution. When computing a task's response time, this blocking time has to be included in addition to the influence from the higher priority tasks. The following claim is useful to determine the length of the blocking time to a task.

CLAIM 2. *An instance of a higher priority task τ_i in its execution can be blocked by at most one instance of a lower priority task τ_j due to preemption threshold, and this lower priority instance can block the higher one only once.*

Proof. If τ_i is blocked by τ_j, $\beta_j >= \alpha_i$. It has been proved in [11] that a higher priority task τ_i can be blocked by at most one lower priority task τ_j due to preemption threshold. When τ_j is in its execution of blocking τ_i, it either continues to complete the execution or is preempted by another higher priority task. In the latter case, τ_j is aborted and its priority goes back to α_j. Since τ_j is a lower priority task to τ_i, $\alpha_j < \alpha_i$. τ_i will execute before τ_j after τ_j is aborted. □

The claim ensures that it does not need to consider the abort cost o a lower priority task when determining the blocking time due to this task's PT. Hence, the worst-case of the blocking time when calculating a task τ_i's response time is as follows.

$$B_i = \max_{\forall k \in lp_i \cap \notin PB_i} C_k - 1 \quad (4)$$

In the equation (4), the blocking time of τ_i is the maximum execution time minus 1 of the tasks that have a lower priority and τ_i cannot preempt when they are executing. After the blocking time is determined, we can extend the equation (2) to compute the worst-case response time of τ_i. The following equation includes the blocking time as defined in equation (4) and $C_j^{PT_i}$ as defined in (3) to calculate the response time of a task.

$$R_i = B_i + C_i + \sum_{\forall j \in hp_i} \lceil \frac{R_i}{P_j} \rceil \times C_j^{PT_i} \quad (5)$$

3.3 An Example for Calculating R_i with PT

We use the same task set in Table 4 for an example to show the uses of equation (3), (4) and (5) to calculate response times of a set of P-FRP tasks. The same priority assignment and same PT of τ_4 are used. Table 5 shows the calculated new WECTs of the tasks when computing the response times of τ_2, τ_3 and τ_4. We use these new calculated WCETs to obtain R_2, R_3 and R_4. The computation steps are shown as below. Please note that because τ_1 is the highest priority task and no task can preempt it or block it, $R_1 = 2$ and we skip its calculation below. Also, because both τ_2 and τ_3 can be blocked by τ_4, $B_2 = B_3 = 4$.

For τ_4:
$R_4 = 5 + \lceil \frac{5}{15} \rceil \times 7 + \lceil \frac{5}{25} \rceil \times 7 + \lceil \frac{5}{45} \rceil \times 4 = 23$
$R_4 = 5 + \lceil \frac{23}{15} \rceil \times 7 + \lceil \frac{23}{25} \rceil \times 7 + \lceil \frac{23}{45} \rceil \times 4 = 30$
$R_4 = 5 + \lceil \frac{30}{15} \rceil \times 7 + \lceil \frac{30}{25} \rceil \times 7 + \lceil \frac{30}{45} \rceil \times 4 = 37$
$R_4 = 5 + \lceil \frac{37}{15} \rceil \times 7 + \lceil \frac{37}{25} \rceil \times 7 + \lceil \frac{37}{45} \rceil \times 4 = 44$
$R_4 = 5 + \lceil \frac{44}{15} \rceil \times 7 + \lceil \frac{44}{25} \rceil \times 7 + \lceil \frac{44}{45} \rceil \times 4 = 44$

For τ_3:
$R_3 = 4 + 4 + \lceil \frac{8}{15} \rceil \times 6 + \lceil \frac{8}{25} \rceil \times 7 = 21$
$R_3 = 4 + 4 + \lceil \frac{21}{15} \rceil \times 6 + \lceil \frac{21}{25} \rceil \times 7 = 27$
$R_3 = 4 + 4 + \lceil \frac{27}{15} \rceil \times 6 + \lceil \frac{27}{25} \rceil \times 7 = 34$
$R_3 = 4 + 4 + \lceil \frac{34}{15} \rceil \times 6 + \lceil \frac{34}{25} \rceil \times 7 = 40$
$R_3 = 4 + 4 + \lceil \frac{40}{15} \rceil \times 6 + \lceil \frac{40}{25} \rceil \times 7 = 40$

For τ_2:
$R_2 = 4 + 3 + \lceil \frac{7}{15} \rceil \times 5 = 12$
$R_2 = 4 + 3 + \lceil \frac{12}{15} \rceil \times 5 = 12$

Table 6 shows a comparison of the response time of each task without and with using PT. It can be seen that after using PT, R_4 is improved greatly because of the reduction of preemptions. On the other side, R_2 and R_3 are increased slightly due to the blocking of the raised priority of R_4 during execution. If we assume $D_i = P_i$ for each τ_i, the task set is determined to be schedulable in the schedulability test.

4 DISCUSSION AND FUTURE WORKS

This work has demonstrated the opportunities of using PT to improve schedulability of P-FRP real-time tasks. In order to explore the opportunities, the schedulability test problem must be solved.

Table 5: Calculated $C_j^{PT_2}$, $C_j^{PT_3}$ and $C_j^{PT_4}$

Task	C_j	$C_j^{PT_4}$	$C_j^{PT_3}$	$C_j^{PT_2}$
τ_1	2	7(2+5)	6(2+4)	5(2+3)
τ_2	3	7(3+4)	7(3+4)	3(3+0)
τ_3	4	4(4+0)	4(4+0)	
τ_4	5	5(5+0)		

Table 6: Comparison of R_i without And with Using PT

Task	Without PT	with PT
R_1	2	2
R_2	8	12
R_3	23	40
R_4	> 100	44

The equation (5), extended from the work in [11], is a sufficient condition to the problem. The equation is used iteratively and conservatively to calculate the worst-case response time of each task. It assumes that when every task starts to run, it aborts a lower priority task with the longest execution time just before the task completes, creating the worst-case scenario. Currently, this is the only sufficient method to solve the schedulability problem. The exact method is still unknown except of the one using simulations for which in some cases it is not practical. It is easy to see that it is not very likely for the worst-case scenario to happen. Therefore, defining a tighter analysis is still a challenge.

As we mentioned earlier, using PT to improve schedulability of P-FRP tasks consists of two sub-problems. We solve the first one of the schedulability problem. The second one is how to select tasks to have PT and what thresholds of preemption are used for the selected tasks. From the example in Table 4, it can be seen that τ_4's PT can be set to 3, 2 and even 1. Similarly, it can also set a PT for τ_2 or τ_3. Each different setting can make a difference to the calculated response times of the tasks. This problem needs to be solved effectively and efficiently to optimize the overall response times. We will solve this problem in a long version of this paper.

REFERENCES

[1] C. Elliott and P. Hudak. Functional reactive animation. ACM the 2nd SIGPLAN international conference on Functional programming, 1997, pp. 263-273.
[2] R. Kaiabachev, W. Taha, and A. Zhu. E-FRP with priorities. ACM EMSOFT 2007, pp. 221-230.
[3] C. Belwal and A. M. K. Cheng. On priority assignment in P-FRP. IEEE RTAS 2010 WiP Session.
[4] C. Belwal and A. M. K. Cheng. Optimal priority assignments in P-FRP. TR-UH-CS-11-03, U. of Houston, 2011.
[5] Y. Jiang, A. M. K. Cheng, and X. L. Zou. Schedulability analysis for real-time P-FRP tasks under fixed priority scheduling. IEEE RTCSA, 2015, pp. 31-40.
[6] C. Belwal, A. M. K. Cheng, and B. Liu. Feasibility interval for the transactional event handlers of P-FRP. Journal of Computer and System Sciences, 2013, 79(5): 530-541.
[7] Y. Jiang, Q. Zhou, X. L. Zou, and A. M. K. Cheng. Minimal schedulability testing interval for real-time periodic tasks with arbitrary release offsets. IEEE ICESS 2014, pp. 611-614.
[8] H. C. Wong and A. Burns. Schedulability analysis for the abort-and-restart (AR) model. ACM 22nd International Conf. on Real-Time Networks and Systems (RTNS), 2014, pp. 119-127.

[9] H. C. Wong and A. Burns. Priority-based functional reactive programming (P-FRP) using deferred abort. IEEE RTCSA 2015, pp. 227-236.
[10] X. L. Zou, A. M. K. Cheng, and Y. Jiang. Deferred start: A non-work-conserving model for P-FRP fixed priority task scheduling. IEEE RTSS 2015 WiP Session.
[11] Y. Wang and M. Saksena. Scheduling Fixed-Priority Tasks with Preemption Threshold. IEEE RTCSA, pp. 328-335.
[12] M. Harbour, M. Klein, and J. Lehoczky. Fixed Priority Scheduling of Periodic Tasks with Varying Execution Priority. IEEE Real-Time Systems Symposium, 1991, pp. 116-128.
[13] M. Joseph and P. Pandya. Finding response times in a real-time system. Computer Journal, 29(5):390-395, 1986.
[14] J. Lehoczky. Fixed priority scheduling of periodic task sets with arbitrary deadlines. IEEE Real-Time Sysrems Symposium, 1990, pp. 201 -209.
[15] J. Lehoczky, L. Sha, and Y. Ding. The rate monotonic scheduling algorithm: Exact characterization and average case behavior. IEEE Real-Time Systems Symposium, 1989, pp. 166-171.