Time Petri Nets for Schedulability Analysis of the Transactional Event Handlers of P-FRP

Chaitanya Belwal  
Department of Computer Science  
University of Houston, TX, USA  
cbelwal@cs.uh.edu

Albert M.K. Cheng  
Department of Computer Science  
University of Houston, TX, USA  
cheng@cs.uh.edu

Yuanfeng Wen  
Department of Computer Science  
University of Houston, TX, USA  
wyf@cs.uh.edu

ABSTRACT

Priority-based FRP (P-FRP) is a functional programming formalism for reactive systems that guarantees real-time response. Preempted tasks in P-FRP are aborted and have to restart when no higher priority tasks are present in the execution queue. The abort and eventual restart makes the response time of a lower priority task completely dependent on the execution pattern of higher priority tasks. Exact schedulability analysis methods of P-FRP that have been presented so far require the evaluation of all release scenarios of higher priority tasks. Unfortunately, the number of such scenarios scales exponentially with the size of the task set, making exact schedulability analysis of this execution model a computationally expensive proposition. The formal method of Time Petri Net (TPN) has previously been used for schedulability analysis of preemptive and non-preemptive models. TPNs for P-FRP or other transaction like execution models have not been developed yet. In this paper, we develop TPN models for the transactional execution model of P-FRP and show that TPNs offer an efficient alternative for schedulability analysis of this model. We have implemented our TPNs in the model checker ROMEO and have validated the correctness of our models through experimental task sets.

Categories and Subject Descriptors
D.3.2 [Programming Languages]: Language Classifications – Applicative (functional) languages

General Terms

Algorithms, Languages

Keywords
FRP, Schedulability analysis, Formal methods, Time Petri Net

1. INTRODUCTION

Functional Reactive Programming (FRP) [25] is a declarative programming language for modeling and implementing reactive systems. FRP elegantly captures continuous and discrete aspects of a hybrid system using the notions of behavior and event, respectively. Because this language is developed as an embedded language in Haskell [11], it benefits from the wealth of abstractions provided in this language. However, Haskell provides no real-time guarantees and therefore, neither does FRP.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

RACS’12, October 23-26, 2012, San Antonio, TX, USA. 
Copyright 2012 ACM 978-1-4503-1492-3/12/10...$10.00.

To address this limitation, resource-bounded variants of FRP were studied ([23],[24],[25]). Recently, it was shown that a variant called priority-based FRP (P-FRP) [13], combines both the semantic properties for FRP, guarantees resource boundedness, and supports assigning different priorities to different events. In P-FRP, higher priority events can preempt lower-priority ones. However to maintain guarantees of type safety and state-less execution, the functional programming paradigm requires the execution of a function to be atomic in nature. To comply with this requirement as well as allow preemption of lower priority events, P-FRP implements a transactional model of execution. Using only a copy of the state during event execution and atomically committing these changes at the end of the event handler (or task), P-FRP ensures that handling an event is an “all or nothing” proposition. This preserves the easily understandable semantics of the FRP and provides a programming model where response times to different events can be tweaked by the programmer, without ever affecting the semantic soundness of the program.

Prior work in P-FRP has dealt with computing approximate ([13], [18]) response time bounds as well as methods to determine response time under a given release scenario ([2], [3]). Multi-processor scheduling of P-FRP has been studied in [19]. For exact schedulability analysis, computing the Worst-Case Response Time (WCRT) of a task is essential. In the preemptive model of execution, a critical instance of release leading to the WCRT exists [14]. However as shown in [2], no such critical instant exists in P-FRP and the WCRT can be different for each unique task set. The only known method of computing the exact WCRT for P-FRP requires evaluation of all release scenarios within a lower and upper bound of release offsets of higher priority P-FRP tasks. Unfortunately, this technique scales exponentially with the size of the task set and is impractical even for small sized task sets.

In a seminal paper, Merlin and Farber [15] introduced Time Petri Net (TPN) as a formal method for verification of timed systems. TPN was shown to be more expressive than Timed Petri Net (TdPN) which was developed by Ramachandani [17]. Following Merlin and Farber’s paper, several researchers have developed techniques which apply TPN for schedulability analysis of real-time systems, notable among these being the works of [21], [10] and [26]. These available methods only deal with different forms of preemptive and non-preemptive execution and so far the application of TPN to a transactional execution model like P-FRP has not been studied.

The use of TPN for schedulability analysis in an execution model like P-FRP’s, provides more immediate benefits than for the standard preemptive or non-preemptive models. This is because while several polynomial time exact schedulability analysis techniques of the preemptive execution model are available, no polynomial time methods exist for such analysis in
P-FRP. Using expressive formalism like TPNs to model the execution of P-FRP and analyzing the schedulability of the model allows us to leverage highly efficient state space exploration techniques available in software tools. Hence, TPN offers an efficient alternative for schedulability analysis in P-FRP.

In this paper, we present Time Petri Net models that can be used for schedulability analysis in P-FRP. These models have been analyzed in a publicly available TPN tool called ROMEO [9]. The results from our TPN model have been validated against simulations using experimental task sets.

2. EXECUTION MODEL AND NOTATIONS

In this section, we introduce the notations used in the rest of the paper. In addition, we review the P-FRP execution model and assumptions made in this study.

2.1 Notations

The notations and formal definitions used in the paper are as follows:

- Let task set \( \Gamma_n = \{ \tau_1, \tau_2, \ldots, \tau_n \} \) be a set of \( n \) sporadic tasks. \( \Gamma_n \) is also called a \( n \)-task set.
- If \( i > j \) then \( \tau_i \) has a higher priority than \( \tau_j \), a convention used in both Linux and Windows operating systems.
- \( \tau_i \) represents the \( k^{th} \) job of \( \tau_i \).
- \( T_j \) is the minimum time separation between two successive jobs of \( \tau_j \).
- \( C_j \) is the worst-case execution time (WCET) for \( \tau_j \).
- \( D_j \) is the relative deadline of \( \tau_j \). In this paper, we assume \( D_j = T_j \).
- Interference on \( \tau_i \) is the action where the execution of \( \tau_i \) is interrupted by the release of a higher priority task.

2.2 Execution Model and Assumptions

In this study, all tasks are assumed to execute in a uniprocessor system with no precedence constraints. Events which trigger the tasks are sporadic and are converted to a periodic model by assuming minimum time interval between any two events of the same type. When a job of a higher priority task \( \tau_i \) is released, it can immediately preempt an executing lower priority task and changes made by the lower priority task are rolled back. The lower priority task will be restarted when the higher priority task has completed execution. Due to P-FRP’s transactional nature of execution, all tasks are assumed to run without concurrency constraints. In the algorithms to derive the actual response time of a task \( \tau_j \), we have considered the release offset of \( \tau_j \) to be 0.

When a task is released, it enters an execution queue which is arranged by priority order such that all arriving higher priority tasks are moved to the head of the queue. The length of the queue is bounded, and no two instances of the same task can be present in the queue at the same time. This requires a task to complete execution before the release of its next job. To maintain this requirement we assume a hard real-time system with task deadline equal to the time period between jobs. Hence,

\[ \forall \tau \in \Gamma_n, D_\tau = T_\tau \]

A task set is schedulable in some time interval only if no task in the set has a deadline miss. While TPNs are capable of building dense time models, in this work we have only considered modeling in discrete time.

3. TIME PETRI NETS

Merlin and Farber [15] presented Time Petri Net as an extension of classical Petri nets where the firing time of transitions are bounded in time. For details on classical Petri Net models readers can refer to [16]. The formal definition of a Time Petri Net is given as follows:

A Time Petri Net is a tuple \( (P, T, B, F, M_0, SI) \) where:
- \( P = \{ p_1, p_2, \ldots, p_p \} \) is a finite non-empty set of places.
- \( T = \{ t_1, t_2, \ldots, t_q \} \) is a finite nonempty set of transitions.
- \( B : P \times T \rightarrow N \) is the backward incidence function, where \( N \) is the set of non-negative integers.
- \( F : T \times P \rightarrow N \) is the forward incidence function.
- \( M_0 \) is the initial marking.
- \( P, T, B, F \) and \( M_0 \) together define a Petri net.
- \( SI \) is a mapping called static interval, \( \forall \tau \in T, \text{SI}(\tau) = (\text{SEFT}(\tau), \text{SLFT}(\tau)) \), where \( \text{SEFT}(\tau) \) is the static earliest firing time and \( \text{SLFT}(\tau) \) the static latest firing time.

The state of a TPN is a pair where \( S = (M, I) \):
- \( M \) is a marking.
- \( I \) is a firing interval set which is a vector of possible firing times. The number of entries in this vector is given by the number of the transitions enabled by marking \( M \).

3.1 TPN modeling using ROMEO

Developed by IRCCyN in France, ROMEO [9] is a tool that allows for the description and evaluation of Time Petri Net models. Using a Graphical User Interface (GUI), places and transitions can be drawn and connected by directional arcs. The static earliest and latest firing times are added to the transitions using an user entry form. Markings are represented by placing tokens in places. In ROMEO, each place has a unique numerical index associated with it.

Once a TPN model is built, a separate simulation program is activated to simulate and analyze the model. ROMEO allows for interactive simulation where the user fires a ready transition through a mouse click. The tokens are dynamically moved to the next place in the forward incidence function after firing the transition. Reachability of states is analyzed using Timed
Computational Tree Logic (TCTL) formatted queries which allow for checking the marking of a state in a specified time bound. Reachability analysis in TPN is done by converting a model to its state space graph. We now demonstrate a simple TPN schedulability model for a P-FRP task set containing 2 tasks.

## 4. TPN For 2-Task Sets

In this section, we present a TPN model for P-FRP 2-task sets. Important concepts dealing with our TPN design can be explained with this simple case, and the model is expanded to include multiple tasks in subsequent sections. To design our TPN, we use the following task set as a case study:

<table>
<thead>
<tr>
<th>Task</th>
<th>( C_i )</th>
<th>( T_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \tau_1 )</td>
<td>4</td>
<td>12</td>
</tr>
<tr>
<td>( \tau_2 )</td>
<td>3</td>
<td>10</td>
</tr>
</tbody>
</table>

The TPN derived for this task set can be used for the schedulability analysis of any 2-task set by changing certain parameters defined in the TPN.

As discussed previously, there is no known single instance of release of higher priority tasks that can lead to the WCRT of a lower priority P-FRP task. Determination of the WCRT requires analysis of all release orderings of higher priority task for their response times. Such a scenario can be modeled quite well in TPN analysis of all release orderings of higher priority task for their parameters defined in the TPN.

### 4.1 Task Release TPN

An important consideration in schedulability analysis of real-time systems, is the release of multiple jobs of higher priority tasks which can interfere with the execution of lower priority tasks. In TPN, we model the periodic release by introducing tokens in initial places. Each token represents a single job of the task. First the maximum number of job releases of a higher priority task \( \tau \), is computed using the following formula:

\[
\text{Number of jobs of } \tau = \left\lceil \frac{D_j}{T_j} \right\rceil \text{ where } \tau_j \text{ is the task whose response time has to be determined.}
\]

This is the maximum number of tokens that need to be placed since if \( \tau_j \) does not complete execution by time \( D_j \), it is deemed unschedulable. Fig. 1(a) shows the TPN design for the release of jobs of \( \tau_2 \) of our sample 2-task set. The deadline of \( \tau_1 \) is 12 hence, the maximum number of jobs of \( \tau_2 \) that can be released within this time is \( \frac{12}{10} \approx 2 \), including the 1st job.

The release of the 1st job of a task is considered separately since it can be released at any time in the interval \([0, D_j]\). Subsequent task releases have to follow the periodic order. 2 tokens are placed for the 1st job while \( \left\lceil \frac{D_j}{T_j} \right\rceil - 1 \) tokens are placed for periodic releases. In fig. 1(a), 2 tokens are in the place ‘First_Release’ while 1 token is in the place ‘Periodic_Release_Start’. Transition ‘T1’ can non-deterministically fire anytime in the time interval \([0, 12]\). These semantics is represent by setting \( SEFT(T1)=0 \) and \( SLFT(T1)=12 \). Since the weight of the arc from place ‘First_Release’ is 2, after ‘T1’ is fired, 1 token is moved to place ‘In_Queue’ while 1 token moves to the place ‘Temp’ and waits for ‘T2’ to fire. A token present in the place ‘In_Queue’ denotes a job of \( \tau_2 \) is present in the execution queue. The token in ‘In_Queue’ is used by the scheduler TPN defined in subsequent sections.

As \( SEFT(T2)=0 \) and \( SLFT(T2)=10 \), ‘T2’ is immediately fired once a token is available in ‘Temp’. After ‘T2’ is fired, all tokens from ‘Periodic_Release_Start’ are moved to ‘Periodic_Release_Temp’. The weight of the arcs connected these two places has a weight equal to the number of tokens in ‘Periodic_Release_Start’. The values \( SEFT(T3)=10 \) and \( SLFT(T3)=10 \) allow ‘T3’ to be fired after 10 time units which is the arrival time period between jobs of \( \tau_2 \). The weight of the arcs connected from ‘T3’ is 1 hence after time intervals of 10, a single token will be moved from ‘Periodic_Release_Temp’ to ‘Periodic_Release’. Once in ‘Periodic_Release’ the token reaches ‘In_Queue’ without any time delay since \( SEFT(T4)=0 \) and \( SLFT(T4)=0 \). This way after the release of the 1st job of \( \tau_2 \) in the interval \([0, D_j]\), rest of the jobs are released at periodic time intervals of \( T_2 \).

The markings in the TPN after firing ‘T1’, releases a token in the execution queue is seen in fig 1(b). The token in ‘Periodic_Release_Temp’ will reach ‘Periodic_Release’ when ‘T3’ is fired after 10 time units.
4.2 Scheduler TPN

In this section, we derive the scheduler TPN for a 2-task set which utilizes the task release TPN introduced earlier. Before that we introduce the important definition of abort costs.

Definition. In the P-FRP execution model, preempted tasks are aborted. The amount of time spent in aborted execution of a task \( \tau \) is called the abort cost. When \( \tau \) is preempted immediately after starting, a minimum abort cost of 1 time unit in added to its response time. If \( \tau \) is preempted just before it is about to complete execution, maximum abort costs of \( C_j - 1 \) time units are added to its response time.

Fig. 2 shows the TPN scheduler model of our 2-task P-FRP set. The task release TPN for \( \tau_2 \) has been included and when a token is available in the place ‘Tau_2_In_Queue’ it implies that a job of this higher priority task is available for execution. The execution of \( \tau_1 \) which is assumed to be released at time 0, is represented by places ‘P1’, ‘P3’ and ‘Tau_1_Complete’. A single token to denote the execution state of \( \tau_1 \) is placed in ‘P1’ and if the token reaches ‘Tau_1_Complete’ it implies that \( \tau_1 \) has completed execution. The maximum abort cost that can be induced on \( \tau_1 \) is \( (4 - 1) = 3 \), hence \( \tau_1 \) can only be preempted if it has executed for any time in the interval \( (0,3] \). This semantics is reflected by setting \( SEFT(T1) = 3 \) and \( SLFT(T1) = 3 \) denoting that \( \tau_1 \) has not been preempted. After ‘T1’ fires, \( \tau_1 \) enters place ‘P3’ and waits for ‘T4’ to fire. \( SEFT(T4) = 1 \) and \( SLFT(T4) = 1 \) which allows ‘T4’ to fire after exactly 1 time unit allowing the marking to reach the place ‘Tau_1_Complete’.

While \( \tau_1 \) is executing (tokens in place ‘P1’), it can be preempted by arrival of jobs of \( \tau_2 \). The preempt of \( \tau_1 \) is denoted by the transition ‘T2’ which can immediately fire \( (SEFT(T2)=SLFT(T2) = 0) \) once tokens are available in ‘P1’ and ‘Tau_2_In_Queue’. After ‘T2’ is fired 2 tokens reach ‘P4’ and wait for ‘T5’ to fire. \( SEFT(T5) \) and \( SLFT(T5) \) values are set to the execution time of \( \tau_2 \). Note that no other transitions are possible from ‘P4’ since \( \tau_2 \) cannot be preempted by any other task. After ‘T5’ is fired, a token is returned to ‘P1’ and another token reaches ‘Tau_2_Complete’. The number of tokens in ‘Tau_2_Complete’ denotes the number of jobs of \( \tau_2 \) that have completed execution. Once a token reaches ‘P1’ it waits for ‘T1’ to fire denoting that \( \tau_1 \) has restarted execution. If another token reaches the place ‘Tau_2_In_Queue’ before ‘T1’ is fired, the similar process described before is repeated.

After \( \tau_1 \) has completed, jobs of \( \tau_2 \) can directly execute through transition ‘T12’, which only requires a single token to be present in ‘Tau_2_In_Queue’. After ‘T12’ is fired, a token reaches ‘P12’ where after 3 time units it reaches ‘Tau_2_Complete’. This path is not of particular interest since once the token in ‘P1’ reaches ‘Tau_1_Complete’, it implies that \( \tau_1 \) has completed execution. However for completeness of our TPN, it is required to model the case when no job of \( \tau_1 \) is ready for execution.

In our TPN model, it is clear that if jobs of \( \tau_2 \) are released at a rate which does not allow transition ‘T1’ to fire, \( \tau_1 \) will never be able to complete execution. We now show how the TPN for 2-task sets can be used to determine schedulability.

4.3 Schedulability Analysis

To determine the schedulability of a 2-task set we have to determine if lower priority task \( \tau_1 \) can complete execution before its deadline. In our TPN model, this problem is translated to the place ‘Tau_1_Complete’ getting marked with a token within bounded time. Or, we have to check if any release scenario of higher priority task \( \tau_2 \) exists in which the token present in ‘P1’ does not reach ‘Tau_1_Complete’ after the deadline of \( \tau_1 \) has passed.

ROMEO allows such queries on a TPN model using the syntax of Time Computational Tree Logic (TCTL) which is derived from the original syntax of Computational Tree Logic (CTL). Readers can refer to [7] for a syntactical reference on CTL. The TPN model defined by the user is converted to a state space graph. Every possible state that can be reached using the markings and time values defined in the TPN is contained in the graph. Efficient state space search techniques like DBM are used to find specific states defined in the TCTL query. Details on implementation of TCTL in ROMEO are given in [6]. The ROMEO TCTL query on the TPN model of fig. 2 for determining schedulability of \( \tau_1 \) in our sample task set is:

\[
EF[13,13](M(22) = 0).
\]

The query returns a Boolean true or false depending whether it is satisfied or not. The query specifies that starting at time 13 and ending at time 13, is it possible for the place in index number 22 (index for place ‘Tau_1_Complete’) not to be marked with a single token? If true, then there exists a scenario in which the token at ‘P1’ is unable to reach ‘Tau_1_Complete’ within the given time bound. Since, the deadline of \( \tau_1 \) is 12, if ‘Tau_1_Complete’ is unmarked after this time it implies that there exists a worst-case release scenario of \( \tau_2 \) in which \( \tau_1 \) misses its deadline and the task set is unschedulable. Hence, for \( \tau_1 \) to be schedulable in all release scenarios the query \( EF[13,13](M(22) = 0) \) should return a Boolean value of false.

5. TPN FOR 3-TASK SETS

We extend the 2-task TPN presented earlier, for schedulability analysis of a 3-task set. As before, we use a sample task set to illustrate the model of the TPN. The task set considered for the 3-task set is the following:

<table>
<thead>
<tr>
<th>Task</th>
<th>( C_i )</th>
<th>( T_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \tau_1 )</td>
<td>4</td>
<td>36</td>
</tr>
<tr>
<td>( \tau_2 )</td>
<td>2</td>
<td>15</td>
</tr>
<tr>
<td>( \tau_3 )</td>
<td>3</td>
<td>10</td>
</tr>
</tbody>
</table>

The TPN model for this task set is given in fig. 3. Places ‘Tau_3_First_Release’, ‘P18’, ‘P5’, ‘P20’, ‘Tau_3_Periodic_Release’, ‘Tau_3_In_Queue’ deal with the first and periodic release of jobs of \( \tau_3 \). \( SEFT \) and \( SLFT \) values of transitions ‘T18’ and ‘T17’ are assigned relevant values. Places ‘Tau_2_First_Release’, ‘P14’, ‘P15’, ‘P16’, ‘Tau_2_Periodic_Release’, ‘Tau_2_In_Queue’ deal with the first and periodic release of jobs of \( \tau_2 \) and \( SEFT \), \( SLFT \) values of transitions ‘T15’ and ‘T14’ are assigned values accordingly. Places ‘P1’ and ‘P3’ deal with the execution of \( \tau_1 \). As in 2-task sets, a single token is placed in ‘P1’ and when the token reaches ‘Tau_1_Complete’ it implies that \( \tau_1 \) has completed execution.

\( T_i = 36 \), hence the number of jobs of \( \tau_2 \) that can be released till the deadline of \( \tau_1 \) are, \( \left\lfloor \frac{36}{15} \right\rfloor = 3 \). Therefore, 2 tokens are placed in ‘P15’ which is part of the release TPN of \( \tau_2 \). Similarly, \( \left\lfloor \frac{36}{10} \right\rfloor = 3 \) tokens are places in ‘P5’ for jobs of \( \tau_3 \). Arcs from transitions ‘T16’ and ‘T9’ are assigned weights accordingly.
Once a job of $τ_2$ is released, a token is placed in 'Tau_2_In_Queue' and the token moves through places 'P9', 'P10' and finally, 'Tau_2_Complete'. 'P9' is reached after firing 'T7' which requires a token to be present in 'P1' and 'Task_2_In_Queue'. This way $τ_1$ is not allowed to execute once $τ_2$ commences execution. After 'T3' fires, $τ_2$ has completed execution and one token is returned to 'P1' while the other reaches 'Tau_3_Complete'.

An important consideration is that while $τ_2$ can preempt $τ_1$, it can itself be preempted by $τ_1$. This semantics is reflected by firing transition 'T12' once tokens are available in 'P1'. To fire, 'T12' also requires a token to be present in 'Tau_3_In_Queue'. Note that once 'T12' is fired, the token which was originally in 'P1' is now in 'P12'. After 'T13' is fired, $τ_2$ completes execution and a token is returned to 'Tau_2_In_Queue' while the remaining tokens reach 'P1' and 'Tau_3_Complete'. SEFT(T13) = SLFT(T13) = 3, to denote the execution of $τ_2$. Like for $τ_1$, the execution of $τ_2$ is broken into two parts. SEFT(T8) and SLFT(T8) values are set to the maximum abort costs that can be induced on $τ_2$. Once $τ_2$ executes for its maximum abort cost, 'T8' is fired allowing a token to reach 'Tau_2_Complete' once 'T3' fires after a single time unit.

If $τ_1$ is released and no job of $τ_2$ is executing or ready to execute, then $τ_2$ preempts the execution of $τ_1$. This semantics is reflected in transitions 'T2' and 'T5'. 'T5' denotes the execution of $τ_1$ therefore, SEFT(T5) = SLFT(T5) = 3. After 'T5' fires, a single token is returned to 'P1' while the other reaches 'Tau_3_Complete'.

### Schedulability Analysis

As for 2-task sets, schedulability analysis requires determining if $τ_1$ can complete execution before its deadline in all release scenarios. In ROMEO, this can be accomplished by the following TCTL query:

\[ EF[37,37](M(22)=0) \]

The query specifies that at time 37 it is possible for place at index 22 (i.e., 'Tau_1_Complete') to have no tokens. If at time 37, 'Tau_1_Complete' has no token it implies that a release scenario of higher priority tasks exists in which $τ_1$ misses its deadline. Hence for $τ_1$ to be schedulable, the query $EF[37,37](M(22)=0)$ should be a Boolean value of false.

In certain task sets it may be possible that even though $τ_1$ completes execution before its deadline, some jobs of $τ_2$ have a deadline miss. To ascertain the schedulability of the overall task set it is important to check for this condition as well. TCTL queries are powerful enough to allow us to check for these conditions in the same TPN. There are two ways in which this can be done. First, we can use the same query format as for $τ_1$. Note that the non-deterministic release of the 1st job of $τ_3$ in the interval [0,36) will also determine the WCRT of $τ_2$. Hence, we have to check if it is possible for the 1st job of $τ_2$ to not complete before its deadline. This is achieved by the following TCTL query:

\[ EF[16,16](M(6)=0) \]

The place at index 6 is 'Tau_2_Complete'. For $τ_2$ to be schedulable, this query should return a Boolean false. Another query format that can determine the schedulability of $τ_2$ is described below.

In the period [0, 16), the maximum number of tokens that can be present in the place 'Tau_2_Complete' is 1 to account for a single job of $τ_2$ that is released in the interval [0, 16). If this job of $τ_2$ is schedulable under all release scenarios of $τ_2$ in the time period [0, 16), only one job of $τ_2$ will be executing. Hence the sum of the number of tokens in 'P16' and 'Tau_2_Complete', will not be less than (3 – 1) = 2. This check can be accomplished by the following TCTL query:

\[ EF[0,16](M(16)+M(6)<2) \]

This query specifies to look for a state between times 0 and 16, where the sum of tokens in state 'P16' (index 16) and 'Tau_2_Complete' (index 6) is less than 2. If such a state exists it means that in the time interval [0, 16) there exists a scenario when more than one job of $τ_2$ is executing or in the execution queue resulting a deadline miss. Hence, for $τ_2$ to be schedulable this query should also return a Boolean false.
We did an experimental comparison between simulations and our TPN models to ascertain the schedulability of P-FRP task sets. Experimental results validated the accuracy of the TPN models and highlighted the short time it takes to conduct the analysis using TPN relative to simulations. Details on the experimental evaluation are available in the expanded edition of this paper [5]. Guidelines to scale the TPN models to any general n-task set are also available in [5].

6. RELATED WORK

Related work includes schedulability techniques developed for P-FRP as well as using Time Petri Nets for schedulability analysis in other execution models. Apart from a minimal discussion on response time given by Kaibachev et al [13], an algorithm to compute approximate values of WCRT is given by Ras and Cheng [18]. Though the average approximation factor of the WCRT derived by this algorithm is unknown, for several task sets it fails to give a result. This algorithm has also been used for response time analysis of P-FRP in multi-processor systems by Ras and Cheng [19]. Important contributions on schedulability analysis of P-FRP have also been made by Belwal and Cheng [2], [3].

One of the first papers that studies actual TPN models for schedulability analysis in real-time systems was presented by Tsai et al [21]. In this paper, methods to transform a schedulability model to Time Petri Nets are presented and the authors prove that schedulability analysis is reduced to a state reachability problem. Xu et al [26] do schedulability analysis using TPN by separating timing and behavioral properties. They present TPN models which allow for easy schedulability analysis. The TPN models presented in [21] and [26] are theoretical in nature since they have not been implemented in a software tool, and thus not been validated. Furfaro and Negro [8] also have present TPN Models for schedulability analysis. In [8], the TPN models have been translated to a Timed Automata model and validated in the modeling tool UPPAAL [22]. A formal mechanism to convert Time Petri Nets to corresponding Timed Automata models has been given by Gu and Shin [10]. However, all these prior works on Time Petri Nets deal only with the preemptive or non-preemptive models of execution and cannot be directly applied for schedulability analysis in P-FRP.

7. CONCLUSIONS

P-FRP is a new functional programming formalism for implementing real-time embedded systems. Due to the state-less execution of functional programs, preempted tasks in P-FRP have to be aborted. The abort of preempted tasks lead to an execution semantics which is different from preemptive or non-preemptive execution and makes the schedulability analysis of this model complicated. We have presented TPN models for the schedulability analysis of P-FRP and shown that TPN offers an efficient alternative to existing polynomial time methods.

In future work, we will be modifying our TPN models to do schedulability analysis lock-free execution [1] and transactional memory [12], which use similar transactional execution semantics.

8. ACKNOWLEDGMENTS

This work is supported in part by the National Science Foundation under Award Nos. 0720856 and 1219082.

9. REFERENCES