On Schedulability Analysis of Non-Cyclic Generalized Multiframe Tasks

Noel TCHIDJO MOYO¹, Eric NICOLLET¹,
Frederic LAFAYE¹
¹THALES Communications S.A., Colombes, France
noel.tchidjomoyo@fr.thalesgroup.com
eric.nicollet@fr.thalesgroup.com
frederic.lafaye@fr.thalesgroup.com

Christophe MOY²
²SUPELEC/IETR
Campus de Rennes, France
christophe.moy@supelec.fr

Abstract—The generalized multiframe (GMF) task has been proposed to model a task whose execution times, deadlines and minimum separation times are changed according to a specified pattern. In this paper we relax the assumption of having a specified activation pattern, this yields to non-cyclic GMF task. In this context, current schedulability analysis techniques for GMF task sets under dynamic priority assignment cannot be used. This paper presents response time analysis of non-cyclic GMF tasks executing on a uniprocessor according to earliest deadline first (EDF) scheduling policy. Also, a density-based sufficient schedulability test for non-cyclic GMF task sets is given. Finally an efficient approach is presented, for exact feasibility determination using computer simulation.

Keywords-component: schedulability analysis, non-cyclic GMF task, EDF.

I. INTRODUCTION

Most of the studies on hard real-time scheduling algorithms have relaxed the assumption of the classical scheduling theory of Liu and Layland [1]. Mok and Chen were the first to introduce the multiframe concept [2] as a generalization of the task model of Liu and Layland. A multiframe task is a task whose execution time is periodically changed according to a specified pattern. For example a task with execution times of 5 ms and 8 ms is said to have two frames, and these two execution times change in each period following a specified pattern.

The GMF task model is an extension of the multiframe task model in the following two directions: (1) different frames may have different minimum separation time and (2) different frames may have different deadlines. The main difference between the GMF task model and the non-cyclic one comes from the fact that there is no activation pattern of task. In other words, in the non-cyclic GMF task model it is possible to activate any frame at the end of the minimum separation time of the previous frame. We use the term “non-cyclic” because the activation pattern, which creates the cycle, is not specified.

Figure 1 represents the two tasks models. In the top one a GMF task i ((1,4), (2,5,6), (3,6,7)) consists of three frames. The execution time of the second frame is 2, its relative deadline is 5, and its minimum separation time is 6. In other words when the second frame is requested, the following frame, which is the frame of execution time 3, is requested after 6 or more time units.

We choose to represent the activation pattern of a task, with a vector made of the successive frames execution times.

We assume that the GMF task is uniformly distributed over a reference interval, and not concentrated at specific points. Following this approach, the transformation of a non-cyclic GMF task to a real time transaction produces a transaction with different periods, so the period of the transaction changes from one phase to another and we do not know in which order since there is no pattern which repeats itself. Clearly the response time analysis proposed for transaction scheduled under dynamic priority assignment cannot be applied.

In order to take into account the relation of offset between tasks, Tindell [4] proposed the model of tasks with offsets (transactions). A response time analysis for transactions scheduled under EDF, has been proposed in [5] by Palencia and Harbour, followed by Pellizzoni and Lipari in [6]. Traore et al mentioned in [7] that the multiframe (MF) model was a particular case of tasks with offsets, so they assume that the offset analysis can be applied to the MF model. More recently Rahni shows in [8] how a GMF task can be transformed into a real time transaction.
An example of non-cyclic GMF task model often found in industrial radio modem applications is the multi-user implementation of a modulation scheme. The radio modem can serve multiple users/services through different frames size handling. Video frame usually takes much more processing time than the voice frame, and voice frame must have the shortest global latency in the system. That implies for tasks in the applications to have for each frame different execution times and deadlines. Clearly, we cannot presuppose an arrival pattern of frames since the system is driven by events coming from users (user A makes an audio call while user B watches a video broadcast).

A typical radio modem set, is represented in Figure 2. It is composed of three processors and we are focusing on the schedulability analysis in a processor of interest. The application contains a transmitting chain, a receiving chain and a chain to control the radio. Each chain consists in functions (A, B, C ...), which represent signal processing algorithms. The arrows between functions represent the dataflow. Typically, G is a demodulation function for which the execution time is proportional to frame size whereas I is a flexible decoding function for which different execution times are due to different parameters for audio and video frame. Without loss of generality, we choose to map each function to a task, which yields to non-cyclic GMF task model.

This paper provides the following contributions:

- Response time analysis and density-based sufficient schedulability test (which runs in polynomial time) for non-cyclic GMF tasks scheduled under EDF.
- An efficient approach, for exact feasibility determination using computer simulation.
The remainder of this paper is organized as follows. In the next section our system model is introduced. Section 3 shows the response time analysis and the density-based sufficient schedulability test for EDF scheduling. An efficient approach for exact feasibility determination is discussed in section 4. Conclusions are provided in section 5.

II. SYSTEM MODEL

The analysis on this paper considers a system that is composed of \( m \) independent non-cyclic GMF tasks with a non-stop runtime \( L \).

A non-cyclic GMF task \( \tau_i \) consisting of \( N \) frames is characterized by a sequence of 3-tuples \( ((C^0_i,D^0_i,P^0_i), \ldots, (C^{N-1}_i,D^{N-1}_i,P^{N-1}_i)) \) where \( C^j_i, D^j_i, \) and \( P^j_i \) represent the maximum execution time of the \( j \)-th frame of task \( \tau_i \), the relative deadline of the frame, and the minimum separation time between the frame and the following one, respectively. Each task makes an initial release at \( O_i \). The first frame is denoted as the 0-th frame, the second is denoted as the 1-st frame, and so on.

A GMF task \( \tau_i \) is said to satisfy the localized Monotonic Absolute Deadlines (l-MAD) property, when the absolute deadline of each frame is no later than that of the following frame, or when \( D^j_i \leq P^j_i + D^{(j+1)modN} \) holds for all \( j \) \( (0 \leq j \leq N-1) \)[3].

A GMF task is defined to satisfy the Frame separation (FS) property, when the absolute deadline of each frame is no later than the arrival time of the following frame, or when \( D^j_i \leq P^j_i \) holds for all \( j \) \( (0 \leq j \leq N-1) \)[9].

Obviously, a task set satisfying the FS property always satisfies the l-MAD property.

In this paper, we consider non-cyclic GMF tasks satisfying the FS property. All frames have the same priority, and we consider that frames are requested exactly at the minimum separation time. In the following section, we discuss the schedulability of non-cyclic GMF task set.

III. SUFFICIENT SCHEDULABILITY TEST

We consider for the purpose of determining sufficient feasibility test, a simultaneous initial release of all the tasks at time \( t \) \( (\forall i \in [0, m-1], O_i = t) \) and that their succeeding frames arrive as soon as possible \( (\forall i \in [0, m-1], \forall j \in [0, N-1] \ , \ D^j_i = P^j_i) \). This creates a maximum computation demand on the processor producing the hardest scenario for the tasks to meet their deadlines.

We first turn to the problem of finding response time of non-cyclic GMF tasks.

- Finding response time

For an activation of a non-cyclic GMF task \( \tau_i \) at time \( t \), due to the arrival of its \( j \)-th frame, the response time of the task is composed of the correspondent frame execution time, plus the interference that is suffered by \( \tau_i \) due to activations of other tasks of the system with arrival times and absolutes deadlines within \([t,t+D^j_i]\).

Let \( R_{t,t+D^j_i} \), the response time of \( \tau_i \), we have

\[
R_{t,t+D^j_i} = C^j_i + I_{t,t+D^j_i}
\]

where \( I_{t,t+D^j_i} \) represented the interference that is suffered by the task \( \tau_i \). We now present analysis to find \( I_{t,t+D^j_i} \).

To illustrate how we find a formulae for the interference, we consider the system example of TABLE I.

<table>
<thead>
<tr>
<th>task</th>
<th>( C^j_i )</th>
<th>( D^j_i = P^j_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \tau_1 )</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>17</td>
</tr>
<tr>
<td>( \tau_2 )</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>16</td>
</tr>
</tbody>
</table>

TABLE I. SYSTEM EXAMPLE 1

The system is composed of two tasks. Task \( \tau_1 \) has three frames while task \( \tau_2 \) has just one frame. We are interested in the schedulability of \( \tau_2 \) considering an initial simultaneous release with \( \tau_1 \). To calculate the interference due to the simultaneous activation of the task \( \tau_1 \), we first eliminate for the task \( \tau_1 \) its second frame (the frame with the deadline at time 17), this is because the arrival of this frame cannot delay the execution of \( \tau_2 \) since the deadline is greater than the one of \( \tau_2 \). Figure 3 represents as a tree, the different activations sequences of \( \tau_1 \) that can interfere with \( \tau_2 \) until the time 16. Each path of the tree represents a possible execution sequence, which can interfere, since arrival time and deadlines are within the time interval \([0;16]\).

For example path 4 corresponds to the sequence \([2;1;1]\). Let \( S_1 \) represents the set of different executions sequence of \( \tau_1 \) which can interfere with the execution of \( \tau_2 \), we have

\[
S_1 = \{[2;2;2],[2;2;1],[2;1;2],[2;1;1],[1;1;2],[1;1;1],[1;2;2],[1;2;1],[1;2;2],[1;2;1]\}
\]

where \([2;2;2] \) is for the path 1, \([2;2;1] \) for the path 2, \([2;1;2] \) for the path 3, and so on.
Formally, for a task \( \tau_u \), the set of its activation sequences which can interfere with the execution of another task \( \tau_i \) is given by:

\[
S_u = \left\{ C_{u,i}^{l_1}, C_{u,i}^{l_2}, ..., C_{u,i}^{l_{m_i}} \right\}_{l_1, l_2, ..., l_{m_i} \in [0, N-1]}
\]

such that

\[
t + \sum_{l_v} D_{u,v}^{l_v} \leq t + D_{i}^{l_i}
\]

We now propose a density-based test of non-cyclic GMF task sets.

**Theorem 3.1**

Given a task set composed of \( m \) non-cyclic GMF tasks dealing with \( N \) frames, if

\[
\sum_{i=0}^{m-1} \max_{0 \leq j \leq N-1} \left( \frac{C_{i,j}^{l_i}}{D_{i}^{l_i}} \right) \leq 1
\]

then the task set is schedulable under EDF on a uniprocessor.

**Proof:**

By considering a simultaneous initial release, the proof consists on verifying that, when the condition (2) is satisfied, all tasks meet their deadlines for their first activation. Indeed, the following activations, either at worse they will happen simultaneously and we will obtain the same situation as in the first activation or they will not happen simultaneously and therefore represent a less critical situation since the interference suffered by each task is not maximum.

For clarity purpose of the proof, let consider the \( x \)-th frame for each task \( i \), where \( \frac{C_{i,j}^{l_i}}{D_{i}^{l_i}} = \max_{0 \leq j \leq N-1} \left( \frac{C_{i,j}^{l_i}}{D_{i}^{l_i}} \right) \). The condition (2) becomes

\[
\sum_{i=0}^{m-1} \frac{C_{i,j}^{x_i}}{D_{i}^{x_i}} \leq 1
\]

Without loss of generality, consider that the first activation of a task \( \tau_j \), is initiated by its \( j \)-th frame, in order to the task \( \tau_j \), to meet its deadline, it is sufficient that the following condition is verified:

\[
C_{i,j}^{x_i} + \sum_{\tau_{\Delta} \in \Delta_{i}} \left( \sum_{l_v} D_{u,v}^{l_v} \leq D_{i}^{l_i} \right) \leq D_{i}^{l_i}
\]

We then show that equation (3) is verified, when equation (2) is verified.

We have

\[
\frac{C_{i,j}^{x_i}}{D_{i}^{x_i}} \leq \frac{C_{u,i}^{x_u}}{D_{u}^{x_u}}
\]

We have

\[
\frac{C_{u,i}^{x_u}}{D_{u}^{x_u}} \leq \frac{C_{u,i}^{x_u}}{D_{u}^{x_u}}
\]

We have

\[
\sum_{l_v} D_{u,v}^{l_v} \leq D_{i}^{l_i} \Rightarrow \sum_{l_v} D_{u,v}^{l_v} \leq D_{i}^{l_i}
\]

The interference suffered by a task \( \tau_j \) is given by

\[
I_{j+D_{i}^{l_i}} = \sum_{\tau_{\Delta} \in \Delta_{i}} \left( \sum_{l_v} D_{u,v}^{l_v} \leq D_{i}^{l_i} \right)
\]

The response time is then equal to

\[
R_{j+D_{i}^{l_i}} = C_{i,j}^{l_i} + \sum_{\tau_{\Delta} \in \Delta_{i}} \left( \sum_{l_v} D_{u,v}^{l_v} \leq D_{i}^{l_i} \right)
\]
by using this inequality we therefore have
\[ C_i^j + \sum_{\tau_i \in \mathcal{N}_i} \frac{\sum_{u} c_{iu}^j}{\sum_{u} d_{iu}^j} \leq C_i^j + \sum_{\tau_i \in \mathcal{N}_i} \left( \frac{\sum_{u} d_{iu}^j}{\sum_{u} d_{iu}^j} \right) \]
\[ C_i^j + \sum_{\tau_i \in \mathcal{N}_i} \frac{c_{iu}^j}{d_{iu}^j} \leq D_i^j \]
\[ D_i^j \leq D_i^j \]
\[ \sum_{\tau_i \in \mathcal{N}_i} \frac{c_{iu}^j}{d_{iu}^j} \leq 1 \] (by equation (2))

This method can be applied for each task \( \tau_i \).

This proves the theorem 3.1.

Response time analysis and density-based test presented above are both sufficient albeit not necessary, when tasks are permitted to have arbitrary initial release time, a common release time between all tasks in task set may not occur. Moreover we do not have necessarily \( D_i^j = P_i^j \) (succeeding frames arrive as soon as possible). Nevertheless, the interesting point of the density-based sufficient test is its very small runtime cost. Thus, in radio modem applications that can be dynamically reconfigured, this test can be easily done online, with respect to real-time constraints. The following section discusses about an efficient approach for exact analysis.

IV. EXACT SCHEDULING ANALYSIS OF NON-CYCLIC GMF TASKS

To illustrate the problem of exact analysis, another simple system example with 2 tasks (\( \tau_1 \) and \( \tau_2 \)) is represented in Table II. \( \tau_1 \) is a non-cyclic GMF task with 2 frames and an initial release at time 3. \( \tau_2 \) has just one frame with an initial release at time 0. The system has a non-stop-runtime L. The different execution possibilities of \( \tau_1 \) are represented in Figure 4.

<table>
<thead>
<tr>
<th>task</th>
<th>O_i</th>
<th>C_i^j</th>
<th>D_i^j</th>
<th>P_i^j</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \tau_1 )</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>6</td>
</tr>
<tr>
<td>( \tau_2 )</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>8</td>
</tr>
</tbody>
</table>

As shown in Figure 4, the possible execution behaviors of \( \tau_1 \) is easily represented with a tree. Each path of the tree represents a possible execution sequence. For example the path 5, has the sequence \([2; 1; 2; 2]\) at the time 25.

For feasibility determination by simulation, an EDF simulator can be implemented with a queue holding the task instances released and a clock representing the time progression. Whenever a task is released, it is added in the ready queue.

![Figure 4. Execution possibilities of \( \tau_1 \) in example 2](image)

Typically a 4-tuple \((X_i^q, Y_i^q, Z_i^q, \#i)\) is added on the queue \( q \) where \( X_i^q \) is the execution time of the task \( \tau_i \), \( Y_i^q \) his relative deadline, \( Z_i^q \) his next release time, and “\#i” is the task identifier. The queue is sorted according to increasing deadline. When the clock steps, execution time of the task at the first position in the queue is decreased. The relative deadline of each task in the queue is also decreased. Hence when the execution time of a task in the queue becomes greater than his relative deadline this means a deadline will be missed. When the computation time of a task instance
becomes 0, his parameters \((X_i^q)\) and \((Y_i^q)\) are not any more modified until the next release time is reached.

Let \(Pa_i^k\) the \(k_i\)-th path execution sequence of a task \(\tau_i\) during a given non-stop-runtime \(L\). The precise mathematical setting of feasibility determination is that we need to find all vectors \([Pa_0^k,Pa_1^k,...,Pa_{m-1}^k]\) whose can be run on the EDF simulator without encountered a “deadline missed” situation.

A. Naïve approach

In the naïve approach, we might first make a list of all possible vectors \([Pa_0^k,Pa_1^k,...,Pa_{m-1}^k]\), \(\forall k_0,k_1,...,k_{m-1}\). Then, from this list we check that each vector does not lead the EDF simulator to a “deadline missed” situation.

This approach requires considering all combinations of possible behaviors for all the tasks in the system and will quickly result in combinatorial explosion and rapidly prove computationally intractable. This problem is enormously further aggravated when the non-stop-runtime is long and there are many tasks. The following section presents an efficient approach, which reduces the number of required combinations.

B. Efficient approach

In our approach, we “grow” each vector from the left to the right, thus at each time, we simulate the execution behavior with a ready queue, and test to see if there is a queue produced by another vector which is identical. In this case, we immediately replace the two queues with one.

1) Definition

Let \(Q_1\) and \(Q_2\), two queues, which have constructed a partial schedule following EDF strategy at time \(t_1\). If for each task \(i\), we have \((X_i^1 = X_i^2 \text{ and } Y_i^1 = Y_i^2 \text{ and } Z_i^1 = Z_i^2)\) or \((X_i^1 = X_i^2 = 0 \text{ and } Z_i^1 = Z_i^2)\), then the two queues are identical (for scheduling point of view). In other words from \(t_1\) they will construct the same schedule.

Successively, while constructing the different executions behaviors of tasks, we can apply this technique to the queues, and replace identical queues by one, thereby saving the effort of constructing vectors and simulate identical schedules.

Thus, at the initial release of the first task, the number of queues corresponding to the possible execution behavior is created. When a new task is released, a routine combines the possible execution behavior of the new task with the previous released tasks. This duplicates the number of queues. At each time, each queue constructs a schedule as described above following the EDF strategy and identical queues are replaced by one.

In order to illustrate how our approach runs, we return to system example of TABLE II. As shown in Figure 5, at time 0 \(\tau_1\) is added in the first queue Q1. The queue simulates the EDF schedule. At the initial release of \(\tau_1\) (time 3), the 4-tuple \((2, 3, 9, \#1)\) is added to Q1. Q1 will therefore simulate the case where \(\tau_1\) (at time 3) receives his first frame. Simultaneously, a new queue Q2 is created for the second possible case for \(\tau_1\). It contains the previous 4-tuple of Q1 \((0, 2, 8, \#2)\), with the 4-tuple \((1, 4, 7, \#1)\). Q2 simulates the case where \(\tau_1\) at time 3 receives his second frame.
time = 0;
while ( time < NONSTOPRUNTIME )

If ( all tasks have not been released at least once )
for $T_i$ in remaining task to initiate their first release
  if ( $O_i$ == time )
    if ( the list is empty)
      for each frame of $T_i$
        q = create new queue;
        addtuple(q, $T_i$);
        addqueuetolist(list, q);
      endfor
    else
      adddifferentpossibilitieswithprevioustaskreleased(list, $T_i$);
    endif
  endif
endfor
endif

time++;
for each queue in list
  executeEDFstrategy(q);
  if ( deadlinemissed(q) ) return unschedulable;
  for each tuple in q
    if ( exist i: $Z_i^j$ = time ) adddifferentcombination(list, $T_i$ );
  endfor
  endfor
eliminateidenticalqueue(list);
endwhile
for each queue in list
  executeEDFstrategyforemainingworkload(q);
  if ( deadlinemissed(q) ) return unschedulable;
endfor
return schedulable;

Figure 6. Feasibility determination algorithm

This remaining workload must be less than the delay after the non-stop-runtime, fixed by the designer, before the system receives another burst of frames.

This algorithm has been implemented and tested on a computer. The efficiency of this approach can be measured according to the number of queues managed at each time, since each combination duplicates the number of queues according to the number of frames of tasks. Figure 7 shows a comparison between the number of queues managed in the new approach and the number of queues that would be managed in the naïve approach for task system of TABLE II.

As shown in Figure 7, for a non-stop-runtime of 300 time units the naïve approach would manage 149 queues while in the new approach we manage 4 queues. After 300 time units, the task set is correctly scheduled (with no deadline missed). The maximum remaining workload among the 4 queues is 1 time units of computation. So if a delay of 1 time units after the non-stop-runtime is acceptable by the designer, the system can be implemented.

Figure 7. Evolution of managed queues in example 1
This approach has been experimented with the system example of TABLE III. It is composed of three tasks. Each task has two frames.

As shown in Figure 8 the gain is also significant. After 300 time units, no deadline was missed. We obtain a maximum remaining workload of 13 time units of computation. While the naïve approach would deal with 203046 queues, the new approach deals with 32794 queues.

<table>
<thead>
<tr>
<th>Task</th>
<th>( O_i )</th>
<th>( C_i^j )</th>
<th>( D_i^j )</th>
<th>( P_i^j )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \tau_1 )</td>
<td>0</td>
<td>2</td>
<td>8</td>
<td>12</td>
</tr>
<tr>
<td>( \tau_2 )</td>
<td>3</td>
<td>4</td>
<td>11</td>
<td>16</td>
</tr>
<tr>
<td>( \tau_3 )</td>
<td>13</td>
<td>6</td>
<td>14</td>
<td>18</td>
</tr>
</tbody>
</table>

TABLE III. SYSTEM EXAMPLE

Figure 8 clearly shows how the new approach is more efficient (less exponential) than the naïve approach.

- **Discussion**

The non-cyclic GMF task model defined in this paper is less restrictive than the GMF task model proposed in [3]. Indeed, the cyclic GMF task model represents one execution sequence through the infinite possibility in the non-cyclic one. Clearly by considering the radio modem introductory example, previous modeling techniques [2] [3] [10] fail to model such a system. Using our model however, we can construct a model for the system.

V. CONCLUSION

This paper has relaxed the assumption of having a specified activation pattern for GMF tasks sets and has addressed response time analysis and a density-based sufficient schedulability test for EDF scheduling of non-cyclic GMF tasks. An efficient approach for exact analysis has been presented. It uses computer simulation for feasibility determination. In general, non-cyclic GMF tasks are frequently encountered in real-time signal processing systems because these applications are becoming more flexible and adaptive, hence workflow does not follow predetermined patterns. Whilst this paper presents a significant simplification of the exact analysis, we believe that further consideration in tasks parameters may yield more efficiency.

REFERENCES


