Introduction

In real-time systems, many scheduling algorithms are used to schedule tasks so that they will all fit and reach their deadlines in time. Reaching temporal correctness is a goal of real-time systems, as this leads to predicable behavior [4]. As tasks need more functionality at higher levels of assurance [8], more solutions are needed with varying levels of reliability. Over many years, algorithms were developed to design tasks if tasks can be feasibly put together in a time range. Tasks are made of many jobs and each job has a period which they are released. Jobs with an equal deadline and release time period are known as implicit deadline tasks. Implicit deadlines are very important with a popular algorithm: EDF.

Explanation of EDF

EDF (Earliest Deadline First) is one of the simplest algorithms to test because of its conditions. In EDF, all tasks have a static priority, determined by their deadline. A shorter deadline means the task should have a higher priority in execution. By theorem 4-1 [3. Liu and Layland, Horn], EDF algorithm can produce an optimal feasible schedule. EDF has a sufficient and necessary condition; all tasks can be scheduled on a uniprocessor if and only if its total utilization is at most 1. This means, that under certain conditions, EDF is the best scheduling algorithm and will always work if it means those conditions.

Utilization of an implicit deadline task can be calculated by execution time (C) divided by period (T). If the total utilization of all tasks <= 1, the task set is schedulable by EDF. If the task set is not schedulable by EDF, it cannot be scheduled at all due to EDF being optimal for all preemptive, dynamic-priority tasks. An example of this can be shown with 3 tasks: (3, 1), (5, 2), and (8, 2).

When calculating the utilizations, we get: 1/3 (0.33), 2/5 (0.4), and 2/4 (0.25) for tasks 1, 2, and 3, respectively. Summing all the utilizations gives us 0.98 < 1. This schedule should be feasible by EDF, and we can create the schedule to show it.

All jobs of the tasks meet their deadline, however if we change the execution time of task 3 to 3 this set is no longer schedulable. Total utilization becomes 1.11 > 1, so EDF cannot give us a feasible task set and this set cannot be scheduled preemptively.

Mixed Criticality

In embedded systems, there are many functions that are implemented in a platform. However, not all the functionalities that are implemented into a shared platform are important (or critical) to the overall system [1]. Systems have many implementations in them, some are required like landing systems or calculating altitude on a plane. However, some functional implementations like entertainment and comfort are not critical to the system functioning correctly, they are just extras for the user. This mixed of safety-critical and non-safety critical systems are known as mixed criticality (MC) systems.

In MC systems, high criticality and low criticality systems are integrated together, due to the increasing trend towards computerized control of functionalities. [2015 paper intro] While only a small fraction of the total system is high criticality, they still need to be met to prevent total disaster.

MC systems also have different notation that normal systems. Jobs J, have 4 parameters {a, d, L, C}, where a is the absolute release time and d is the absolute deadline.

Criticality is maintained by the certification authorities (CA). As CAs are conversative, they require safety-critical functionalities to be shown at a very high level of assurance; and the rest of the non-safety-critical functions are validated by the system designer/integrator. [1] This gives us the smallest criticality level: 2. One level of criticality is needing certification and the other not needing it. There can be many different criticality levels for a system. For example, RTCA published DO-178B, a software development process for commercial aircrafts that has 5 different criticality levels.

<table>
<thead>
<tr>
<th>Level</th>
<th>Failure Condition</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Catastrophic</td>
<td>Failure may cause a crash</td>
</tr>
<tr>
<td>B</td>
<td>Minor</td>
<td>Failure has a large negative impact on safety or performance, or reduces the ability of the crew to operate the plane due to physical distress or a higher workload, or causes serious or fatal injuries among the passengers</td>
</tr>
<tr>
<td>C</td>
<td>Major</td>
<td>Failure is significant, but has a lower impact than a Catastrophic failure (for example, leads to passenger discomfort rather than injuries)</td>
</tr>
<tr>
<td>D</td>
<td>Minor</td>
<td>Failure is noticeable, but has a lower impact than a Major failure (for example, causing passenger inconvenience or a routine flight plan change)</td>
</tr>
<tr>
<td>E</td>
<td>None</td>
<td>Failure has no impact on safety, aircraft operation, or crew workload</td>
</tr>
</tbody>
</table>

At a higher level, you need more rigorous validation requirements to ensure you have the right level and the whole system does not fail. The rigorous validation can cause problems, as a lot of resources can go into validation causing worse-case execution time. [1] When high criticality tasks execute for longer than they are supposed to, the
extra time is taken from the lower-criticality tasks. MC systems need to obtain temporal correctness despite the high-criticality tasks and allow lower-criticality tasks to perform as well.

**What is this paper about?**

This survey topic is about EDF-VD in mixed criticality systems. We know that EDF is optimal for a set of jobs with arbitrary release times and deadlines when utilization \( \leq 1 \), but what are the conditions for EDF with mixed criticality systems? Is EDF optimal for mixed criticality systems and what effect do they have on EDF being schedulable. This paper will try to answer those questions.

**What is EDF-VD?**

Normal EDF was already analyzed, but EDF-VD is Earliest Deadline First with virtual deadlines. EDF-VD was proposed by Baruah et. Al. to scheduled MC systems. The purpose of EDF-VD is to be able to finish low-criticality tasks early so extra time can be spent for high-criticality tasks. Virtual deadlines help finish low-criticality tasks earlier by giving a shorter deadline. EDF will follow these new deadlines and can give extra slack at a high-criticality switch. Virtual deadlines evaluated at runtime and assigned to jobs under EDF scheduling [2]. A schedulability test is ran with these deadlines to see if EDF-VD is schedulable with the deadlines.

**Virtual Deadlines**

In [2], a preemptive processor schedules an EDF task system’s virtual deadline as such with 2 steps:

1) Find an \( x \) for all low-criticality tasks are scheduled
2) If all high-criticality tasks are less than 1 then the \( x \) is correct otherwise calculate another \( x \)

\( x \) is a parameter to find the modified period of the high-criticality task. The \( x \) should of course be the minimum possible, to prevent a long deadline.

Once the test deems the set schedule, a modified period \( \hat{T} \) is computed for each Hi-criticality task. [5]

Normal EDF with a single criticality level is not the same as EDF with multiple levels. In [2 14:7], EDF is optimal for any set that has a utilization \( \leq 1 \). This is sufficient and necessary. However, this is not true if are multiple criticality levels. EDF is no longer optimal and maybe not create a feasible schedule, even with total utilization \( \leq 1 \). In [2] there is an example to disprove this with 2 tasks.

<table>
<thead>
<tr>
<th>( \tau_1 )</th>
<th>( \chi_i )</th>
<th>( c_i(1) )</th>
<th>( c_i(2) )</th>
<th>( p_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \tau_1 )</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>( \tau_2 )</td>
<td>2</td>
<td>1</td>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

In this example [6] total utilization on level 1 is \( (2/4) + (1/6) = (2/3) \) and on level 2 it is \( (5/6) \). Both utilizations are \( \leq 1 \) but there can still be a deadline miss. If a task \( \tau_1 \) holds the process for too long, when the system switches to high criticality mode, \( \tau_2 \) may not have enough time to execute itself and miss the deadline.

When trying to schedule MC tasks using EDF algorithms is to deal with the overrun of high-criticality tasks when the system switches from low-criticality mode to high-criticality mode. An example of this switch is given in the paper in Liu’s 2016 paper [1]:

![Illustrative example](image)

**Table I: Illustrative example**

<table>
<thead>
<tr>
<th>Task</th>
<th>( L )</th>
<th>( G_{LO} )</th>
<th>( G_{HI} )</th>
<th>( T_r )</th>
<th>( D_t )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \tau_1 )</td>
<td>LO</td>
<td>4</td>
<td>6</td>
<td>10</td>
<td>7</td>
</tr>
<tr>
<td>( \tau_2 )</td>
<td>HI</td>
<td>4</td>
<td>8</td>
<td>10</td>
<td>7</td>
</tr>
</tbody>
</table>

![Figure 1: Scheduling of Example 1](image)

Task 1 originally need 4 execution, but once the switch to high-criticality happens, changes to its execution for high criticality. Task 1 would originally need to be executed for 2-time units, but it already did 1 in low-criticality mode. Task 2 has already finished by the time the switch happens, so it can remain idle.

When you artificially restrict high-criticality deadlines in low-criticality mode so that a system can have more time for high-criticality mode, you make virtual deadlines. A virtual deadline is never larger than the deadline, \( d_i \). In Figure 1, we can see the virtual deadline is at time 7, denoted by \( D_i \). While in low-criticality mode, \( \tau_2 \)’s deadline is 7. However, when the system switches to high-criticality mode, \( \tau_2 \) no longer has its virtual deadline, it has now been switched to its implicit deadline. Also, because this is in EDF, \( \tau_1 \) can now preempt \( \tau_2 \) due to \( \tau_2 \)’s deadline change. \( \tau_1 \) only has a budget of 2 in high-criticality mode, so it executes once and suspends, leaving the rest of the execution to \( \tau_2 \).

**EDF vs EDF-VD**

EDF-VD may seem like EDF with restricted deadlines, but it is more than that. There are many
things that go on the systems that cannot be shown on a scheduling diagram. EDF on a uniprocessor only
needed to manage the scheduling, changing, and
execution of the tasks. The deadlines needed to be
compared for the current time and executed
accordingly. However, EDF-VD needs to not only
spend time calculating the virtual deadlines (such as
the minimal X), but also perform the switch for
criticality modes.

The switch from low to high or high to low
modes takes some overhead time. This is a run-time
event in [1], and after the switch all the calculations
for deadlines must be redone in the opposite mode.
One of the tasks with scheduling EDF-VD is to be
able to have the scheduling done in a way that the
time reserved outweighs the cost from all the
switching and calculations EDF-VD does. [2]

**Criticality modes and schedulability**

For EDF-VD, there are 2 criticality modes:
low and high. When check for EDF-VD schedulability,
we should first check if low-criticality has valid
schedulability.

Mixed Criticality also has a different
notation for utilization with different modes.
Utilization is a ratio of Worse Case Execution Time to
a period, rather than execution time to deadline.
Utilization for MC systems has 2 scripts: the
subscript is the sum of utilizations under that level,
and the superscript is the limit of execution. For
example, \( U_{\text{LO}}^{\text{HI}} \), is the sum of utilization under high-
criticality and LO is the assumption each task
executions with a max limit of the low-critical worse-
case. [2] However, \( U_{\text{LO}}^{\text{LO}} \) does not exist, because there
is no execution for high-criticality tasks specified for
low-criticality tasks. [7]

Any tasks that break this rule of execution
(executing longer than its limit) must signal its
completion.

**Low-criticality scheduling**

Low criticality is very important for not only
itself but also for high-criticality. In our case of 2
criticality levels, the system will always start off in
low-criticality.[8] Therefore, analysis of low criticality
will eventually affect high-criticality.

**Theorem 1** [1] shows the condition for
scheduling low-criticality tasks. There is a modifier x,
used to modify the relative deadline of tasks. This
modifier can be used to only compare only the low-
criticality utilization or include the high-criticality
too; by being 1 or 0. This is the only theorem used to
prove schedulability for low-criticality.

\[
1 \geq U_{\text{LO}}^{\text{LO}} + \frac{U_{\text{HI}}^{\text{LO}}}{x}
\]

It should be no surprise that the sufficiency
condition for EDF-VD, is nearly identical to
uniprocessor EDF, however this is no longer
sufficient and necessary.

**High-criticality scheduling**

For high-criticality mode, MC systems
discard all low-criticality jobs running, but with a
sorter execution time (degraded quality). In [1], Liu
derives the sufficiency test for high-criticality mode.
For high-criticality mode, 1 theorem is no longer
acceptable like low-criticality’s schedulability test.
This is due to the many different factors a high-
criticality schedule can have. The main case shown in
[1] is the release of a job in the high-criticality
interval. If a job is released in the interval, it may
need to execute some of its high-criticality then
suspend. The example shown in figure 1’s schedule
has no task released during high criticality.

Also, the job does not have to start as soon
as the switch happens, it can be carried in due to its
deadline. While EDF-VD is preemptable, the task
may still have higher priority when it is brought into
high-criticality.

Assumptions and propositions are made to
assure all the cases are covered. Also, as stated
before, you cannot look at high-criticality without
checking for the schedulability for low-criticality,
therefore high-criticality requires not only it is
theorem (theorem 2 in [1]) but also theorem 1 for
low-criticality. If both conditions are met, EDF-VD is
schedulable under mixed criticality systems.

One of the assumptions is minimality of the
task set. Minimality is the minimal number of jobs
that can be released by a task set that results in a
deadline miss. Minimality is important because it
allows EDF-VD to schedule those jobs with a
deadline and helps to contradict any incorrect
proofs. [1]

However, unlike normal EDF and low-
criticality EDF-VD, this condition no longer compares
the utilization 1. Instead, the 2 theorems are
combined to give the equation for high-criticality
EDF-VD scheduling.

\[
\frac{U_{\text{LO}}^{\text{LO}}}{1 - U_{\text{LO}}^{\text{LO}}} \leq 1 - (U_{\text{HI}}^{\text{HI}} + U_{\text{HI}}^{\text{LO}})
\]

This is Lui’s equation for scheduling under EDF-VD.
[1]

**Conclusion**
In conclusion, EDF-VD is most like EDF. It uses the same kind of analysis, the deadline that is the earliest, to schedule tasks. However, when applying multiply criticality levels, EDF changes in its schedulability. EDF-VD still schedules the same way, but it is no longer optimal for uniprocessors with MC systems. When the system is in low-criticality mode, we can see that in theorem 1, the utilization only needs to be \( \leq 1 \), for the most part.

However, the instant the system switches into high-criticality mode, EDF starts to have trouble. By the proof in theorem 2 [1]: EDF-VD can still schedule the task set, but it takes more calculations. The virtual deadlines use to allow fewer resources to be spent on high-criticality tasks are tested and placed to allow for all the other low-criticality tasks. Schedulability requires many more calculations for high-criticality. Many assumptions and propositions are needed to prove that EDF-VD is schedulable for high-criticality mode. Low-criticality only required itself, but high-criticality requires itself and low-criticality as the modes start from low-criticality. High-criticality mode also factors in the switch time, this can make a big difference for tasks that are not only in the interval from instance it switched but also when it switches back (if it does). The virtual deadlines that were created are removed for high-criticality mode.

While EDF-VD is no longer optimal or necessary, it can still be used to schedule MC systems.

References