Real-Time Scheduling of Mixed-Criticality Systems: What are the “X” Factors?

Risat Pathan
Chalmers University of Technology, Sweden

Workshop on Mixed Criticality Systems, Rome, Italy
December 2, 2014
Mixed-Criticality (MC) Systems

• Tasks have different criticalities

• Criticality specifies “importance” of a task
  ➢ Higher criticality => higher importance
  ➢ Correctly executing higher criticality tasks is more important

• What is correct execution?
  ➢ The functional output is right, i.e., 1+2=3
  ➢ The functional output is timely, i.e., deadline is met
The “X” Factors

• Correct execution of high-critical tasks can be threatened at runtime by events ("X" factors)

• Examples of “X” factors for MC systems:
  ➢ WCET overrun (Steve Vestal, RTSS 2007)
    ✓ Difficulty in estimating WCET
  
  ➢ Occurrences of errors and the need for recovery
    ✓ Hardware problems, environmental effects, software bugs
This research

• Scheduling constrained-deadline sporadic tasks with three constraints:
  ➢ Meeting *hard deadlines*
  ➢ Error recovery using *time-redundant execution* (called, backups)
  ➢ Respecting criticality to facilitate *certification*

• An instance of a task is called a *job* that must
  ➢ generate the correct output, and
  ➢ meet its deadline
Error Model

• Task (Job) Errors
  – wrong path, wrong output, etc.
  – Why do we have errors?
    • Errors are caused by faults
      – Hardware Transient Faults
        » temporary malfunctioning of the computing unit
        » happen for a short time and then disappear (not permanent)
    – Software Bugs
      » Bugs may remain undetected after testing
Error Recovery

• Hardware Transient Faults
  ➢ By re-execution
  
  ![Bubble Sort](image1)

  ![Bubble Sort](image2)

  Primary and backup have same WCET

• Software Bugs
  ➢ Re-execution may not be effective (permanent error)
  ➢ By executing a diverse implementation of the same task

  ![Bubble Sort](image3)

  ![Bubble Sort](image4)

  ![Heap Sort](image5)

  Primary and backup may have different WCET
Error Recovery

Time-redundant Backups:
re-execution, or
diverse-implementation execution

Each task has one **primary** and several **backups**
- Backups are executed until it is detected to be non-erroneous

Tolerating **multiple** errors are considered
- the **same** job may be erroneous multiple times
- **different** jobs of the same or different tasks may be erroneous
How errors are detected?

• Based on existing error detection mechanisms
  – Examples
    ▪ HW based: watchdog processor, illegal opcode detection, etc.
    ▪ SW based: assertions, duplication and comparison

• Undetected errors have to be tolerated using space redundancy (Not covered in this work)
Certification and Mixed-Criticality
Certification of MC Systems

• Certification is about assurance
  ✓ higher criticality=>higher assurance in meeting deadlines

• Different WCETs of the same task [Vestal, RTSS07]
  ✓ Higher assurance => larger WCET
  ✓ C^{LO} and C^{HI} where C^{LO} ≤ C^{HI}

• Different numbers of errors in each interval ≤ D_{max}
  ✓ Higher assurance => higher number of error recovery
  ✓ f and F where f ≤ F
Different numbers of errors in each interval $\leq D_{\text{max}}$

- $f$ and $F$ where $f \leq F$

What does it mean by particular number errors in each interval $\leq D_{\text{max}}$?

$f=1 \quad F=3 \quad D_{\text{max}}=4$
Different numbers of errors in each interval $\leq D_{\text{max}}$

- $f$ and $F$ where $f \leq F$

What does it mean by particular number errors in each interval $\leq D_{\text{max}}$?

- $f=1$, $F=3$, $D_{\text{max}}=4$
Different numbers of errors in each interval \( \leq D_{\text{max}} \)

✓ f and F where \( f \leq F \)

What does it mean by particular number errors in each interval \( \leq D_{\text{max}} \)?

\[
f = 1 \quad F = 3 \quad D_{\max} = 4
\]
Different numbers of errors in each interval $\leq D_{\text{max}}$

✓ $ f $ and $ F $ where $ f \leq F $ 

What does it mean by particular number errors in each interval $ \leq D_{\text{max}} $?

$f=1\quad F=3\quad D_{\text{max}}=4$
Task Model

• Total $n$ sporadic tasks
  – Task $\tau_i \equiv (L_i, C_i, B_i, D_i, T_i)$

✓ dual-criticality  $L_i \in \{\text{LO}, \text{HI}\}$

✓ Different WCETs of primary and backups of a task
  ▪ Primary: $C_i = <C_{i,\text{LO}}, C_{i,\text{HI}}>$ where $C_{i,\text{LO}} \leq C_{i,\text{HI}}$
  ▪ Backups: $B_i = <B_1, B_2, \ldots B_f, \ldots B_F>$
    • where $B_1 = <B_{1,\text{LO}}, B_{1,\text{HI}}>$ where $B_{1,\text{LO}} \leq B_{1,\text{HI}}$
    • where $B_2 = <B_{2,\text{LO}}, B_{2,\text{HI}}>$ where $B_{2,\text{LO}} \leq B_{2,\text{HI}}$
    • $\ldots$
    • where $B_F = <B_{F,\text{LO}}, B_{F,\text{HI}}>$ where $B_{F,\text{LO}} \leq B_{F,\text{HI}}$
  ▪ relative deadline $\leq$ period, i.e., $D_i \leq T_i$
Task Model

• Total $n$ sporadic tasks
  – Task $\tau_i \equiv (L_i, C_i, B_i, D_i, T_i)$

• Tasks are given **fixed priorities**
  – Primary and backups of a task have the same priority

  – $hp(i)$: the set of higher priority tasks of task $\tau_i$
    ▪ Higher-priority and LO-Critical tasks
    ▪ Higher-priority and HI-Critical tasks

• Tasks are executed on **uniprocessor**
Scheduling Problem Statement

How to ensure that all the deadlines are met on uniprocessors?

- different freq. of errors for different assurance levels
- different WCETs of the primary and backups for different assurance levels
Outline

• Task model
  – Criticality Behaviors
• Scheduling Algorithm
  – Schedulability analysis and test
• Evaluation
• Conclusion
Outline

• Task model
  – Criticality Behaviors
• Scheduling Algorithm
  – Schedulability analysis and test
• Evaluation
• Conclusion
Criticality behavior

• Different assumptions regarding the two X factors
  ▪ Assumptions that hold at runtime determines criticality behavior

• Exhibits **LO-Crit behavior** as long as
  ▪ LO-Crit assumptions regarding **all** X factors hold

• Switches to **HI-Crit behavior** when
  ▪ LO-Crit assumptions regarding **at least one X factor** does not hold
    ○ Actual exec. time of some primary/backup exceeds $C^{LO}/B^{LO}$, or

![Diagram showing criticality behavior over time with intervals $C^{LO}$, $C^{HI-CLO}$, and $C^{HI}$.]
Criticality behavior

• Different assumptions regarding the two X factors
  ▪ Assumptions that hold at runtime determines criticality behavior

• Exhibits **LO-Crit behavior** as long as
  ▪ LO-Crit assumptions regarding *all* X factors hold

• Switches to **HI-Crit behavior** when
  ▪ LO-Crit assumptions regarding *at least one X factor* does not hold
    o Actual exec. time of some primary/backup exceeds $C^{LO}/B^{LO}$, or
    o The $(f+1)^{th}$ error is detected in an interval $\leq D_{max}$

  \[
  f=1 \quad F=3 \quad D_{max}=4
  \]
Criticality behavior

• Different assumptions regarding the two X factors
  ▪ Assumptions that hold at runtime determines criticality behavior

• Exhibits LO-Crit behavior as long as
  ▪ LO-Crit assumptions regarding all X factors hold

• Switches to HI-Crit behavior when
  ▪ LO-Crit assumptions regarding at least one X factor does not hold
    o Actual exec. time of some primary/backup exceeds $C^L / B^L$, or
    o The $(f+1)^{th}$ error is detected in an interval $\leq D_{\text{max}}$

• After criticality switches, the system exhibits HI-Crit behavior
Both LO and HI-crit tasks execute.
No task executes more than $C^{LO}/B^{LO}$.
At most $f=1$ errors in an interval $\leq D_{max} = 4$

LO-crit Behavior $[0, 12)$

HI-crit Behavior $[12, \alpha)$

Only HI-crit tasks execute.
Task executes at most $C^{HI}/B^{HI}$.
At most $F=3$ errors in an interval $\leq D_{max} = 4$

Criticality behavior switches from LO to HI at $t=12$
Criticality Behavior (X Factors = WCET, freq. of errors)

\[ f = 1 \quad F = 3 \quad D_{\text{max}} = 4 \]

LO-crit Behavior \([0, 12)\)
- Both LO and HI-crit tasks execute.
- No task executes more than \(C^{LO}/B^{LO}\).
- At most \(f = 1\) errors in an interval \(\leq D_{\text{max}} = 4\).

HI-crit Behavior \([12, \alpha)\)
- Only HI-crit tasks execute.
- Task executes at most \(C^{HI}/B^{HI}\).
- At most \(F = 3\) errors in an interval \(\leq D_{\text{max}} = 4\).

At time \(t = 12\), the \((f+1)^{\text{th}} = 2^{\text{nd}}\) error is detected in an interval \(\leq D_{\text{max}} = 4\).

Criticality behavior switches from LO to HI at \(t = 12\).
Outline

• Task model
  – Criticality Behaviors

• **Scheduling Algorithm**
  – Schedulability analysis and test

• Evaluation

• Conclusion
FTMC: Fault-Tolerant Mixed-Criticality Scheduling

FTMC scheduling is same as FP scheduling on uniprocessor

+ execute a backup if an error is detected

+ criticality-switch detection
  - if the \((f+1)\)th error is detected in an interval \(\leq D_{\text{max}}\)
  - if some primary/backup executes \(\geq C^{\text{LO}}/B^{\text{LO}}\)

+ drop all LO-crit tasks after switching
Outline

• Task model
  – Criticality Behaviors
• Scheduling Algorithm
  – Schedulability analysis and test
• Evaluation
• Conclusion
FTMC: Schedulability Analysis

• **Correctness**: The system is *schedulable* in all LO- and HI-criticality behaviors.

  ▪ **LO criticality**: *All (HI- and LO-critical) tasks* meet their deadlines in all LO-crit behaviors

  ▪ **HI criticality**: *All HI-critical tasks* meet their deadlines in all HI-crit behaviors
FTMC: Schedulability Analysis

Response-time analysis for LO- and HI-crit behaviors to find

\[ R_{i}^{\text{LO}} : \text{Response-time at LO-crit behavior} \]

\[ R_{i}^{\text{HI}} : \text{Response-time at HI-crit behavior} \]
Non-MC and Non-FT

Response-time analysis of task $\tau_i$:

$$t \leftarrow C_i + \sum_{\tau_k \in hp(i)} \left[ \frac{t}{T_k} \right] C_j$$

Set of jobs of $\tau_i \cup hp(i)$ during the busy period are

$$\text{JobSet}(t) = \bigcup_{\tau_k \in \tau_i \cup hp(i)} \left\{ J_{k,1}, J_{k,2}, \ldots, J_{k,\left[ \frac{t}{T_k} \right]} \right\}$$
Non-MC and Non-FT

Response-time analysis of task $\tau_i$:

$$t \leftarrow C_i + \sum \left\lfloor \frac{t}{T_k} \right\rfloor C_j$$

Set of jobs of $\tau_i \cup hp(i)$ during the busy period are

$$\text{JobSet}(t) = \bigcup_{\tau_k \in \tau_i \cup hp(i)} \{ J_{k,1}, J_{k,2}, \ldots, J_{k,\left\lfloor \frac{t}{T_k} \right\rfloor} \}$$

If these jobs recover $E$ errors, then what is the total workload in the busy period?

$$\text{Work}(\text{JobSet}(t), E) = ?$$
Example Task Set (F=2)

<table>
<thead>
<tr>
<th>(L_i)</th>
<th>(C_i^{LO})</th>
<th>(B_1^{LO})</th>
<th>(B_2^{LO})</th>
<th>(C_i^{HI})</th>
<th>(B_1^{HI})</th>
<th>(B_2^{HI})</th>
<th>(D_i)</th>
<th>(T_i)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\tau_1)</td>
<td>HI</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>(\tau_2)</td>
<td>LO</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>12</td>
</tr>
<tr>
<td>(\tau_3)</td>
<td>HI</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>14</td>
</tr>
</tbody>
</table>

Primary
### Example Task Set (F=2)

<table>
<thead>
<tr>
<th>( \tau )</th>
<th>( L_i )</th>
<th>( C_{i}^{\text{LO}} )</th>
<th>( B_{1}^{\text{LO}} )</th>
<th>( B_{2}^{\text{LO}} )</th>
<th>( C_{i}^{\text{HI}} )</th>
<th>( B_{1}^{\text{HI}} )</th>
<th>( B_{2}^{\text{HI}} )</th>
<th>( D_i )</th>
<th>( T_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \tau_1 )</td>
<td>HI</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>( \tau_2 )</td>
<td>LO</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>12</td>
<td>14</td>
</tr>
<tr>
<td>( \tau_3 )</td>
<td>HI</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>14</td>
<td>28</td>
</tr>
</tbody>
</table>

**First Backup**
Example Task Set (F=2)

<table>
<thead>
<tr>
<th>( \tau_1 )</th>
<th>( \tau_2 )</th>
<th>( \tau_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( L_i )</td>
<td>( C_i^{LO} )</td>
<td>( B_1^{LO} )</td>
</tr>
<tr>
<td>HI</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>LO</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>HI</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>

Second Backup
Example Task Set (F=2)

<table>
<thead>
<tr>
<th>Li</th>
<th>C_i^{LO}</th>
<th>B_1^{LO}</th>
<th>B_2^{LO}</th>
<th>C_i^{HI}</th>
<th>B_1^{HI}</th>
<th>B_2^{HI}</th>
<th>D_i</th>
<th>T_i</th>
</tr>
</thead>
<tbody>
<tr>
<td>τ_1</td>
<td>HI</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>τ_2</td>
<td>LO</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>12</td>
</tr>
<tr>
<td>τ_3</td>
<td>HI</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>14</td>
</tr>
</tbody>
</table>

**Work({J}, E) = total exec. by job J to recover E errors**

If J is a job of task τ_3 and E=2, then

\[
\text{Work}\left(\{J_{LO}\}, 2\right) = 3 + 3 + 3 = 9
\]

\[
\text{Work}\left(\{J_{HI}\}, 2\right) = 4 + 3 + 3 = 10
\]

**Work(JobSet(t), E)=?**
Steps to Compute $R_{i}^{LO}$ and $R_{i}^{HI}$

• Find **Jobset(t)**: the jobs that are eligible to execute in the busy period are determined.
• **Characterize** each job $J$ as $J_{LO}$ or $J_{HI}$.
• **Workload** is computed considering maximum number of errors in the busy period.
• A **recurrence** is formulated to find the response time.
Finding $R_i^{LO}$
We compute the workload in the level-\(i\) busy period.

Critical instant (Audsley et al. 1991) for sporadic tasks applies:
- when all tasks arrive simultaneously, and
- when jobs of the tasks arrive strictly periodically.
$R_i^{LO}$: Schedulability Analysis

We compute the workload in the level-$i$ busy period

$$\text{JobSet}(t) = \bigcup_{\tau_k \in \tau_i \cup h_p(i)} \{J_{k,1}, J_{k,2}, \ldots J_{k,\lfloor \frac{t}{T_k} \rfloor}\}$$

$$t \leftarrow \text{Work(\text{JobSet}(t), f)}$$
Finding $R_i^{HI}$
\( R_i^H \): Schedulability Analysis

LO-crit tasks execute ONLY during LO-criticality behavior
HI-crit tasks execute during LO- and HI-criticality behavior


\[ X = \text{set of LO-Crit jobs that execute in LO-Crit behavior} \]
\[ Y = \text{set of HI-Crit jobs that execute in HI-Crit behavior} \]
\[ Z = \text{set of HI-Crit jobs that execute in LO-Crit behavior} \]
**R_i^{HI}: Schedulability Analysis**

LO-crit tasks executes ONLY during LO-criticality behavior
HI-crit tasks executes during LO- and HI-criticality behavior


\[
\text{JobSet}(t, s) = X \cup Y \cup Z
\]
$R_i^{HI}$: Schedulability Analysis

Jobs in "JobSet$(t,s)$" are executed in the busy period where at most $F$ errors can occur

$R_{i,s}^{HI}$ is the solution of

$$R_i^{HI} = \max \{ R_{i,s}^{HI} \}$$

$$\text{Work}(\text{JobSet}(t,s), F)$$

Problem window of length $t$
Priority Assignment

- Deadline-monotonic is not optimal for uniprocessor MC system [Vestal, RTSS07]

How to assign the fixed-priorities for MC scheduling on uniprocessor?

Audsley’s Optimal Priority Assignment (OPA)
Audsley’ OPA algorithm

for each priority level k, lowest first

for each priority unassigned task $\tau_i$

If $R_{i}^{HI} \leq D_i$ and $R_{i}^{LO} \leq D_i$ assuming higher priorities for the other priority unassigned task, then

assign $\tau_i$ to priority k

break (continue outer loop)

return “unschedulable”

return “schedulable”
Evaluation
Schedulability Tests

Three tests are evaluated

- **DM-FTMC**: Response time tests with deadline-monotonic priority assignment
- **OPA-FTMC**: Response time tests with OPA
- **UBound test**: Necessary Test
  - This is an upper bound on the schedulable task sets by FTMC algorithm.
Simulation Parameters

• Random mixed-criticality task sets are generated
  ▪ \( U \) : total utilization of a task set \( (\sum C_{LO}/T) \)
  ▪ \( n \) : number of tasks in a task set
  ▪ \( f, F \) : Frequency of errors
  ▪ \( CF \) : \( C^{HI} = C^{LO} \times CF \)

• Backups are same as the primary (i.e., re-execution)
Results

\[ (n = 20, f = 0, F = 1, CF = 1) \]

No pessimism regarding WCET

\[ (n = 20, f = 1, F = 2, CF = 2) \]

Pessimism regarding WCET and freq. of errors
Conclusion

• FP scheduling of sporadic tasks on uniprocessor
  ▪ Real time, fault tolerance, and mixed criticality

• Priority assignment with Audsley’s OPA

• Applicable to more than two criticality levels

**Future work:** Apply it for multiprocessors, probabilistic analysis

**What are the other X factors?**
Thank You
Email: risat@chalmers.se