Generalized Mixed-Criticality Scheduling based on RUN

Romain GRATIA  IRT SystemX
Thomas ROBERT  Télécom ParisTech
Laurent PAUTET  Télécom ParisTech

Automotive Electronics and Software Project
Real-Time Networks and Systems 2015
Embedded system design
Embedded system design
Design revisited as:

- Many functions => High conception complexity
- Uniprocessor replaced by multiprocessor
Design revisited as:

- Many functions => High conception complexity
- Uniprocessor replaced by multiprocessor
Design revisited as:
- Many functions => High conception complexity
- Uniprocessor replaced by multiprocessor

Mixed safety requirements
- Classification in several criticality levels (CL)
- First approach: isolation => partitioning based on CL... but inefficient
Design revisited as:

- Many functions => High conception complexity
- Uniprocessor replaced by multiprocessor

Mixed safety requirements

- Classification in several criticality levels (CL)
- First approach: isolation => partitioning based on CL... but inefficient

Mixed-Criticality (MC) scheduling
Mixed-criticality systems

- MC task model = Vestal periodic task model with implicit deadlines
- Execution time budget estimation depends on criticality level:

Example:
In a system with L criticality level a CL-3 Task has a budget for each criticality level

- Functionalities modeled by set of MC tasks
- Degraded execution mode = shutdown lower criticality tasks
Execution mode $X =$

Execute tasks of criticality level $\geq X$ ($\Gamma \geq X$) with budgets $C(X)$
Mixed-criticality systems

Execution Modes

- Execution mode $X =$

  Execute tasks of criticality level $\geq X \ (\Gamma \geq X)$ with budgets $C(X)$

**Mode $X=1$:**

- Fully-featured mode

- Active

- System starts in mode 1
Mixed-criticality systems

Execution Modes

- Execution mode \( X = \)

Execute tasks of criticality level \( \geq X \) (\( \Gamma \geq X \)) with budgets \( C(X) \)

Mode \( X=1: \) Fully-featured mode

- \( T_1 \)
- \( T_2 \)
- \( T_3 \)
- \( T_4 \)
- \( T_5 \)
- \( T_6 \)

Mode \( X=2: \) Important functionalities

- \( T_2 \)
- \( T_3 \)
- \( T_4 \)
- \( T_6 \)

Active

- System starts in mode 1
Execution mode $X = \Gamma \geq X$ with budgets $C(X)$

- **Mode $X=1$:** Fully-featured mode
  - $T1$, $T2$, $T3$, $T5$, $T4$, $T6$

- **Mode $X=2$:** Important functionalities
  - $T2$, $T3$, $T4$, $T6$

- **Mode $X=3$:** Vital functionalities
  - $T3$, $T6$

Active

- System starts in mode 1
Mixed-criticality systems

Execution Modes

- Execution mode $X =$

  Execute tasks of criticality level $\geq X$ ($\Gamma \geq X$) with budgets $C(X)$

  Mode $X=1$: Fully-featured mode
  - T1
  - T2
  - T3
  - T4
  - T5
  - T6

  Mode $X=2$: Important functionalities
  - T2
  - T3
  - T4
  - T6

  Mode $X=3$: Vital functionalities
  - T3
  - T6

Active

- System starts in mode 1
  Timing Failure Event
Mixed-criticality systems

Execution Modes

- Execution mode $X =$

Execute tasks of criticality level $\geq X$ ($\Gamma \geq X$) with budgets $C(X)$

Mode $X=1$: Fully-featured mode
- T1
- T2
- T3
- T5
- T4
- T6

Mode $X=2$: Important functionalities
- T2
- T3
- T4
- T6

Mode $X=3$: Vital functionalities
- T3
- T6

Active

- System starts in mode 1

Timing Failure Event $\rightarrow$ Mode change
Mixed-criticality systems

Execution Modes

- Execution mode $X = \Gamma \geq X$ with budgets $C(X)$

Mode $X=1$: Fully-featured mode
- T1
- T2
- T3
- T5
- T4
- T6

Mode $X=2$: Important functionalities
- T2
- T3
- T4
- T6

Mode $X=3$: Vital functionalities
- T3
- T6

System starts in mode 1
Timing Failure Event $\rightarrow$ Mode change

Active
Mixed-criticality systems

Execution Modes

- **Execution mode X =**

  Execute tasks of criticality level ≥ X (Γ ≥ X) with budgets C(X)

**Execution Modes**

<table>
<thead>
<tr>
<th>Mode X=1: Fully-featured mode</th>
<th>Mode X=2: Important functionalities</th>
<th>Mode X=3: Vital functionalities</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>T2</td>
<td>T3</td>
</tr>
<tr>
<td>T2</td>
<td>T3</td>
<td>T3</td>
</tr>
<tr>
<td>T3</td>
<td>T4</td>
<td>T6</td>
</tr>
<tr>
<td>T4</td>
<td>T6</td>
<td></td>
</tr>
<tr>
<td>T5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **System starts in mode 1**

  Timing Failure Event → Mode change

**Active**
Several mixed-criticality algorithms for multi-processor exist

Standard real-time scheduling algorithms are adapted to MC model

Problem:

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Schedulability performances</th>
<th>Number of Preemptions</th>
<th>Number of criticality levels managed</th>
</tr>
</thead>
<tbody>
<tr>
<td>fpEDF-VD [Baruah14]</td>
<td>Average</td>
<td>Few</td>
<td>2</td>
</tr>
<tr>
<td>MC-Fluid [Lee14]</td>
<td>Best known</td>
<td>Many</td>
<td>2</td>
</tr>
<tr>
<td>MC²[Mollison10]</td>
<td>Restricted</td>
<td>Few</td>
<td>5</td>
</tr>
</tbody>
</table>
Several mixed-criticality algorithms for multi-processor exist

Standard real-time scheduling algorithms are adapted to MC model

Problem:

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Schedulability performances</th>
<th>Number of Preemptions</th>
<th>Number of criticality levels managed</th>
</tr>
</thead>
<tbody>
<tr>
<td>fpEDF-VD [Baruah14]</td>
<td>Average</td>
<td>Few</td>
<td>2</td>
</tr>
<tr>
<td>MC-Fluid [Lee14]</td>
<td>Best known</td>
<td>Many</td>
<td>2</td>
</tr>
<tr>
<td>MC²[Mollison10]</td>
<td>Restricted</td>
<td>Few</td>
<td>5</td>
</tr>
</tbody>
</table>

Can we design a scheduling algorithm be good in all areas?
Mixed-criticality systems

Several mixed-criticality algorithms for multi-processor exist

Standard real-time scheduling algorithms are adapted to MC model

Problem:

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Schedulability performances</th>
<th>Number of Preemptions</th>
<th>Number of criticality levels managed</th>
</tr>
</thead>
<tbody>
<tr>
<td>fpEDF-VD [Baruah14]</td>
<td>Average</td>
<td>Few</td>
<td>2</td>
</tr>
<tr>
<td>MC-Fluid [Lee14]</td>
<td>Best known</td>
<td>Many</td>
<td>2</td>
</tr>
<tr>
<td>MC²[Mollison10]</td>
<td>Restricted</td>
<td>Few</td>
<td>5</td>
</tr>
</tbody>
</table>

Can we design a scheduling algorithm be good in all areas?

We present our MC scheduling algorithm:

GMC-RUN
The RUN algorithm

Principles

- **RUN** schedules periodic tasks with implicit deadlines
- **Extensively use servers** = task used to schedule tasks or other servers
  - Server utilization no greater than 1
  - Put each task into a server = Task Server (TS)
- **RUN mechanics based on two principles**
  1. Multiprocessor scheduling problem transformed in several uniprocessor ones
     - Uniprocessor problem modelled with servers scheduling task servers
     - Servers may need less than a processor => slack time
  2. Gather slack time to spare processors
The RUN algorithm

**Principles**

- **RUN** schedules periodic tasks with implicit deadlines
- Extensively use servers = task used to schedule tasks or other servers
  - Server utilization no greater than 1
  - Put each task into a server = Task Server (TS)

**RUN mechanics based on two principles**

1. Multiprocessor scheduling problem transformed in several uniprocessor ones
   - Uniprocessor problem modelled with servers scheduling task servers
   - Servers may need less than a processor => slack time
2. Gather slack time to spare processors
The RUN algorithm

Principles

- RUN schedules periodic tasks with implicit deadlines
- Extensively use servers = task used to schedule tasks or other servers
  - Server utilization no greater than 1
  - Put each task into a server = Task Server (TS)
- RUN mechanics based on two principles
  1. Multiprocessor scheduling problem transformed in several uniprocessor ones
     - Uniprocessor problem modelled with servers scheduling task servers
     - Servers may need less than a processor => slack time
  2. Gather slack time to spare processors
- Two kinds of servers:
The RUN algorithm

**Principles**

- **RUN** schedules periodic tasks with implicit deadlines
- Extensively use servers = task used to schedule tasks or other servers
  - Server utilization no greater than 1
  - Put each task into a server = Task Server (TS)
- **RUN mechanics based on two principles**
  1. Multiprocessor scheduling problem transformed in several uniprocessor ones
     - Uniprocessor problem modelled with servers scheduling task servers
     - Servers may need less than a processor => slack time
  2. Gather slack time to spare processors
- **Two kinds of servers:**
  1. Primal server $S$ = uniprocessor scheduling problem
RUN schedules periodic tasks with implicit deadlines

Extensively use servers = task used to schedule tasks or other servers
- Server utilization no greater than 1
- Put each task into a server = Task Server (TS)

RUN mechanics based on two principles
1. Multiprocessor scheduling problem transformed in several uniprocessor ones
   - Uniprocessor problem modelled with servers scheduling task servers
   - Servers may need less than a processor => slack time
2. Gather slack time to spare processors

Two kinds of servers:
1. Primal server $S$ = uniprocessor scheduling problem
   - Leaf primal servers schedule task servers
The RUN algorithm

**Principles**

- **RUN** schedules periodic tasks with implicit deadlines
- Extensively use servers = task used to schedule tasks or other servers
  - Server utilization no greater than 1
  - Put each task into a server = Task Server (TS)
- **RUN** mechanics based on two principles
  1. Multiprocessor scheduling problem transformed in several uniprocessor ones
     - Uniprocessor problem modelled with servers scheduling task servers
     - Servers may need less than a processor => slack time
  2. Gather slack time to spare processors
- **Two kinds of servers:**
  1. Primal server \( S \) = uniprocessor scheduling problem
     - Leaf primal servers schedule task servers
  2. Dual server \( S^* \) = schedule slack time
The RUN algorithm

**Principles**

- **RUN** schedules periodic tasks with implicit deadlines
- **Extensively use servers** = task used to schedule tasks or other servers
  - Server utilization no greater than 1
  - Put each task into a server = Task Server (TS)
- **RUN mechanics based on two principles**
  1. Multiprocessor scheduling problem transformed in several uniprocessor ones
     - Uniprocessor problem modelled with servers scheduling task servers
     - Servers may need less than a processor => slack time
  2. Gather slack time to spare processors
- **Two kinds of servers:**
  1. Primal server \( S \) = uniprocessor scheduling problem
     - Leaf primal servers schedule task servers
  2. Dual server \( S^* \) = schedule slack time
- **Hierarchy of servers is formed offline**
Objective: leave task servers unchanged to reuse RUN

- Preserve RUN good schedulability performances
- Preserve RUN properties concerning the number of pre-emptions

First consider systems with only two criticality levels

Problem:
- One execution time budget per Task Server used in whole RUN structure
- MC model several execution time budgets per task

RUN adaptation
RUN model and MC model
Problem:
- One execution time budget per Task Server used in whole RUN structure
- MC model several execution time budgets per task

Objective: leave task servers unchanged to reuse RUN
- Preserve RUN good schedulability performances
- Preserve RUN properties concerning the number of pre-emption

First consider systems with only two criticality levels

Schedule CL-2 tasks using budgets of mode 2
Objective: leave task servers unchanged to reuse RUN

- Preserve RUN good schedulability performances
- Preserve RUN properties concerning the number of pre-emption

First consider systems with only two criticality levels

Schedule CL-2 tasks using budgets of mode 2

Introduce modal servers for CL-2 tasks
Modal server $MS_i$ for $\tau_i$ a CL-2 task:

- $C(1)$
- $C(2)-C(1)$
- $\tau_i^{\text{start}}$
- $MS_i^2$
- Task server
Modal server $MS_i$ for $\tau_i$ a CL-2 task:

- In mode 2: $\tau_i^{\text{start}}$ and $\tau_i^{\text{finish}}$ scheduled

If mode 2 active
Modal server MS\textsubscript{i} for \(\tau\textsubscript{i}\) a CL-2 task:

- In mode 2: \(\tau\textsubscript{i}\)\textsuperscript{start} and \(\tau\textsubscript{i}\)\textsuperscript{finish} scheduled
- In mode 1: \(\tau\textsubscript{i}\)\textsuperscript{start} and \(\Gamma\textsubscript{X<2}\) scheduled

If mode 2 active

Otherwise
Modal server $MS_i$ for $\tau_i$ a CL-2 task:

- In mode 2: $\tau_i^{\text{start}}$ and $\tau_i^{\text{finish}}$ scheduled
- In mode 1: $\tau_i^{\text{start}}$ and $\Gamma_{X<2}$ scheduled

How to define $\Gamma_{X<2}$?
Allocation problem and schedulability conditions

- Allocation problem: for each modal server, find the lower criticality task set that it can schedule correctly
- MS schedules correctly $\Gamma_{x<2}$ if one schedulability condition fulfilled
  - SC 1: if and only if MS period divides each $\Gamma_{x<2}$ task period, $U(\text{MS}) \geq U(\Gamma_{x<2})$
  - SC 2 (always applicable): compare Demand Bound Function and Supply Bound Function on a hyperperiod (not trivial)
- If no conditions fulfilled promote CL-1 task to a CL-2 one
  - Mode 1: execution of the CL-1 task
  - Mode 2: nothing executed
Aggregated modal server
Gather several unused budgets

- Issue: each modal server has a budget too small to schedule a task
- But overall budget large enough to schedule lower criticality tasks
- Issue: each modal server has a budget too small to schedule a task
- But overall budget large enough to schedule lower criticality tasks
- Gather modal server budgets to schedule same set of tasks

\[ \text{Aggregated modal server} \]

\[ \text{Gather several unused budgets} \]
- **Issue:** each modal server has a budget too small to schedule a task
- **But overall budget large enough to schedule lower criticality tasks**
- **Gather modal server budgets to schedule same set of tasks**
- **Aggregation condition:** execute **sequentially** modal servers

### Diagram:

- **$\tau_1^{\text{start}}$**
- **$\tau_2^{\text{start}}$**
- **$\Gamma_{x<2}$**
- **$\tau_1^{\text{finish}}$**
- **$\tau_2^{\text{finish}}$**

- **If Timing Failure Event**
- **Otherwise**
- **If Timing Failure Event**
Issue: each modal server has a budget too small to schedule a task
But overall budget large enough to schedule lower criticality tasks
Gather modal server budgets to schedule same set of tasks
Aggregation condition: execute sequentially modal servers
Schedule corresponding task servers in a same leaf primal server

\[
\begin{align*}
\tau_1^{\text{start}} & \quad MS_1^2 \\
\tau_2^{\text{start}} & \quad MS_2^2 \\
\tau_1^{\text{finish}} & \quad \Gamma_{x<2} \\
\tau_2^{\text{finish}} & \\
\end{align*}
\]

\[U_1(2) + U_2(2) \leq 1\]
Best allocation: allocation requiring the fewest number of processors
Finding the best allocation is a hard optimization problem

1. Many allocations possible
   - Many possible aggregated modal servers
   - Many possible task allocation in modal servers

2. Assessing schedulability conditions not trivial
   - Schedulability conditions are non-linear (SBF/DBF) and two possible tests
   - Aggregated modal servers prevent handling task allocation in isolation

Our choice: evolutionary algorithm for its flexibility
1. Find a representation to describe all possible allocations:
   - Implemented as matrix
   - Presented as tree manipulation
1. Find a representation to describe all possible allocations:
   - Implemented as matrix
   - Presented as tree manipulation

2. Define operators to explore the set of solutions
   - Mutation: (de-)allocation of a task to a modal server
   - Crossover: form two new allocations by mixing two existing ones
1. Find a representation to describe all possible allocations:
   - Implemented as matrix
   - Presented as tree manipulation

2. Define operators to explore the set of solutions
   - Mutation: (de-)allocation of a task to a modal server
   - Crossover: form two new allocations by mixing two existing ones

3. Evaluate allocations and conserve only best ones
Mutation: (de-)allocation of a task to a modal server
Mutation: (de-)allocation of a task to a modal server
Mutation: (de-)allocation of a task to a modal server
Mutation: (de-)allocation of a task to a modal server
Mutation: (de-)allocation of a task to a modal server

Evolutionary algorithm

Mutation operator
Crossover: form two new allocations by mixing two existing ones
Crossover: form two new allocations by mixing two existing ones
- Crossover: form two new allocations by mixing two existing ones
Experimental assessment

Two criticality levels system: schedulability performances

- X-axis: task set utilization normalized by the number of CPU (here 2)
- Y-axis: acceptance ratio for 500 task sets
- GMC-RUN close to MC-DP-Fair
- fpEDF-VD outperformed

![Graph showing acceptance ratio vs normalized utilization for different schedulers]
Experimental assessment

Two criticality levels system: number of pre-emptions

- Consider only task sets schedulable by GMC-RUN and MC-DP-Fair
- GMC-RUN outperforms MC-DP-Fair
- RUN pre-emption performances preserved

![Graph showing average number of preemption per jobs vs. number of tasks]

- GMC-RUN
- MC-DP-Fair

Number of Tasks vs. Average number of preemption per jobs
Actual systems are generally composed of at least 4 criticality levels.

Approach to schedule systems with 2 criticality levels:
- Schedule CL-1 task into CL-2 task unused budgets
- Or promote CL-1 task to a CL-2 task one

Remove the lowest criticality level: criticality level reduction

Approach to deal with n criticality levels:
- Recursively apply criticality level reduction from lowest to highest criticality levels
Scheduling more than two criticality levels

Generalization rules: modal servers

- Tasks $\tau_i$ of criticality $X > 2$ has $X$ subtasks:
Tasks $\tau_i$ of criticality $X > 2$ has $X$ subtasks:

- Each subtask $\tau_i^l$ can be scheduled by a modal server $MS^l$
  - Completes $\tau_i$ execution when mode $L$ or higher is active
  - Schedules tasks of criticality $< L$, otherwise

Scheduling more than two criticality levels

Generalization rules: modal servers
Tasks \( \tau_i \) of criticality \( X > 2 \) has \( X \) subtasks:

- Each subtask \( \tau_i^l \) can be scheduled by a modal server \( MS^l \)
  - Completes \( \tau_i \) execution when mode \( L \) or higher is active
  - Schedules tasks of criticality \( < L \), otherwise
Tasks $\tau_i$ of criticality $X > 2$ has $X$ subtasks:

- Each subtask $\tau_i^l$ can be scheduled by a modal server $MS^l$
  - Completes $\tau_i$ execution when mode $L$ or higher is active
  - Schedules tasks of criticality $< L$, otherwise

Schedulability condition
Scheduling more than two criticality levels

**Generalization rules: modal servers**

- **Tasks** $\tau_i$ **of criticality** $X > 2$ **has** $X$ **subtasks:**
  - $\tau_i$ execution when mode $L$ or higher is active
  - Schedules tasks of criticality $< L$, otherwise

- **Each subtask** $\tau_i^L$ **can be scheduled by a modal server** $MS^L$
  - Completes $\tau_i$ execution when mode $L$ or higher is active
  - Schedules tasks of criticality $< L$, otherwise

**Schedulability condition**
Each modal server $MS^L$
- either completes $\tau_i$ execution when mode $L$ or higher is active
- or schedules tasks of criticality $< L$, otherwise

Scheduling more than two criticality levels

Generalization rules: modal servers
Each modal server $MS^L$
- either completes $\tau_i$ execution when mode $L$ or higher is active
- or schedules tasks of criticality $< L$, otherwise
Each modal server $MS^L$
- either completes $\tau_i$ execution when mode $L$ or higher is active
- or schedules tasks of criticality $< L$, otherwise
Scheduling more than two criticality levels

**Generalization rules: modal servers**

- **Each modal server** $MS^L$
  - either completes $\tau_i$ execution when mode $L$ or higher is active
  - or schedules tasks of criticality $< L$, otherwise
- **Aggregate MS when they fit in a primal leaf server**
Each modal server $MS^L$
- either completes $\tau_i$ execution when mode $L$ or higher is active
- or schedules tasks of criticality $< L$, otherwise

Aggregate $MS$ when they fit in a primal leaf server

Schedulability condition: use timing parameters of $MS$ and task at its CL
Each modal server $MS^L$
- either completes $\tau_i$ execution when mode $L$ or higher is active
- or schedules tasks of criticality $< L$, otherwise

Aggregate $MS$ when they fit in a primal leaf server

Schedulability condition: use timing parameters of $MS$ and task at its CL

Aggregation condition: sum TS utilizations at max CL (here CL-3)

$U_i(3) + U_j(3) \leq 1$
Scheduling more than two criticality levels

Worst case allocation: promoting task criticality only
Worst case allocation: promoting task criticality only

- Worst case allocation: no allocation possible
Worst case allocation: no allocation possible

Promote task criticality level
Scheduling more than two criticality levels

Worst case allocation: promoting task criticality only

<table>
<thead>
<tr>
<th>Task</th>
<th>Period</th>
<th>Criticality</th>
<th>U(1)</th>
<th>U(2)</th>
<th>U(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\tau_1)</td>
<td>5</td>
<td>3</td>
<td>0.2</td>
<td>0.4</td>
<td>0.7</td>
</tr>
<tr>
<td>(\tau_2)</td>
<td>5</td>
<td>3</td>
<td>0.2</td>
<td>0.3</td>
<td>0.5</td>
</tr>
<tr>
<td>(\tau_3)</td>
<td>5</td>
<td>2</td>
<td>0.1</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>(\tau_4)</td>
<td>10</td>
<td>1</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
</tr>
<tr>
<td>(\tau_5)</td>
<td>20</td>
<td>1</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
</tr>
<tr>
<td>(\tau_6)</td>
<td>40</td>
<td>1</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
</tr>
</tbody>
</table>

- **Worst case allocation:** no allocation possible
- **Promote task criticality level**
Scheduling more than two criticality levels

Worst case allocation: promoting task criticality only

<table>
<thead>
<tr>
<th>Task</th>
<th>Period</th>
<th>Criticality</th>
<th>U(1)</th>
<th>U(2)</th>
<th>U(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>τ₁</td>
<td>5</td>
<td>3</td>
<td>0.2</td>
<td>0.4</td>
<td>0.7</td>
</tr>
<tr>
<td>τ₂</td>
<td>5</td>
<td>3</td>
<td>0.2</td>
<td>0.3</td>
<td>0.5</td>
</tr>
<tr>
<td>τ₃</td>
<td>5</td>
<td>2</td>
<td>0.1</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>τ₄</td>
<td>10</td>
<td>1</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
</tr>
<tr>
<td>τ₅</td>
<td>20</td>
<td>1</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
</tr>
<tr>
<td>τ₆</td>
<td>40</td>
<td>1</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
</tr>
</tbody>
</table>

- **Worst case allocation:** no allocation possible
- **Promote task criticality level**

Number of criticality levels: 3
Scheduling more than two criticality levels

Worst case allocation: promoting task criticality only

<table>
<thead>
<tr>
<th>Task</th>
<th>Period</th>
<th>Criticality</th>
<th>U(1)</th>
<th>U(2)</th>
<th>U(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\tau_1)</td>
<td>5</td>
<td>3</td>
<td>0.2</td>
<td>0.4</td>
<td>0.7</td>
</tr>
<tr>
<td>(\tau_2)</td>
<td>5</td>
<td>3</td>
<td>0.2</td>
<td>0.3</td>
<td>0.5</td>
</tr>
<tr>
<td>(\tau_3)</td>
<td>5</td>
<td>2</td>
<td>0.1</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>(\tau_4)</td>
<td>10</td>
<td>1</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
</tr>
<tr>
<td>(\tau_5)</td>
<td>20</td>
<td>1</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
</tr>
<tr>
<td>(\tau_6)</td>
<td>40</td>
<td>1</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
</tr>
</tbody>
</table>

- Worst case allocation: no allocation possible
- Promote task criticality level

Number of criticality levels: 3
Scheduling more than two criticality levels

Worst case allocation: promoting task criticality only

Worst case allocation: no allocation possible

Promote task criticality level

Number of criticality levels: 2

<table>
<thead>
<tr>
<th>Task</th>
<th>Period</th>
<th>Criticality</th>
<th>U(1)</th>
<th>U(2)</th>
<th>U(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>τ₁</td>
<td>5</td>
<td>3</td>
<td>0.2</td>
<td>0.4</td>
<td>0.7</td>
</tr>
<tr>
<td>τ₂</td>
<td>5</td>
<td>3</td>
<td>0.2</td>
<td>0.3</td>
<td>0.5</td>
</tr>
<tr>
<td>τ₃</td>
<td>5</td>
<td>2</td>
<td>0.1</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>τ₄</td>
<td>10</td>
<td>1</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
</tr>
<tr>
<td>τ₅</td>
<td>20</td>
<td>1</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
</tr>
<tr>
<td>τ₆</td>
<td>40</td>
<td>1</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
</tr>
</tbody>
</table>

Task Criticality level

- 3
  - τ₁
  - τ₂

- 2
  - τ₃
  - τ₄⁺
  - τ₅⁺
  - τ₆⁺
Scheduling more than two criticality levels

Worst case allocation: promoting task criticality only

<table>
<thead>
<tr>
<th>Task</th>
<th>Period</th>
<th>Criticality</th>
<th>U(1)</th>
<th>U(2)</th>
<th>U(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>τ₁</td>
<td>5</td>
<td>3</td>
<td>0.2</td>
<td>0.4</td>
<td>0.7</td>
</tr>
<tr>
<td>τ₂</td>
<td>5</td>
<td>3</td>
<td>0.2</td>
<td>0.3</td>
<td>0.5</td>
</tr>
<tr>
<td>τ₃</td>
<td>5</td>
<td>2</td>
<td>0.1</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>τ₄</td>
<td>10</td>
<td>1</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
</tr>
<tr>
<td>τ₅</td>
<td>20</td>
<td>1</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
</tr>
<tr>
<td>τ₆</td>
<td>40</td>
<td>1</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
</tr>
</tbody>
</table>

- Worst case allocation: no allocation possible
- Promote task criticality level

Number of criticality levels: 1

Task Criticality level

3

τ₁  τ₂  τ₃⁺  τ₄++  τ₅++  τ₆++
Scheduling more than two criticality levels

Worst case allocation: promoting task criticality only

- Worst case allocation: no allocation possible
- Promote task criticality level

Number of criticality levels: 1

<table>
<thead>
<tr>
<th>Task</th>
<th>Period</th>
<th>Criticality</th>
<th>U(1)</th>
<th>U(2)</th>
<th>U(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>τ₁</td>
<td>5</td>
<td>3</td>
<td>0.2</td>
<td>0.4</td>
<td>0.7</td>
</tr>
<tr>
<td>τ₂</td>
<td>5</td>
<td>3</td>
<td>0.2</td>
<td>0.3</td>
<td>0.5</td>
</tr>
<tr>
<td>τ₃</td>
<td>5</td>
<td>2</td>
<td>0.1</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>τ₄</td>
<td>10</td>
<td>1</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
</tr>
<tr>
<td>τ₅</td>
<td>20</td>
<td>1</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
</tr>
<tr>
<td>τ₆</td>
<td>40</td>
<td>1</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
</tr>
</tbody>
</table>

Task Criticality level

- 3

- ✗

- ✗
Scheduling more than two criticality levels

Worst case allocation: promoting task criticality only

<table>
<thead>
<tr>
<th>Task</th>
<th>Period</th>
<th>Criticality</th>
<th>U(1)</th>
<th>U(2)</th>
<th>U(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>5</td>
<td>3</td>
<td>0.2</td>
<td>0.4</td>
<td>0.7</td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>5</td>
<td>3</td>
<td>0.2</td>
<td>0.3</td>
<td>0.5</td>
</tr>
<tr>
<td>$\tau_3$</td>
<td>5</td>
<td>2</td>
<td>0.1</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>$\tau_4$</td>
<td>10</td>
<td>1</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
</tr>
<tr>
<td>$\tau_5$</td>
<td>20</td>
<td>1</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
</tr>
<tr>
<td>$\tau_6$</td>
<td>40</td>
<td>1</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
</tr>
</tbody>
</table>

- Worst case allocation: no allocation possible
- Promote task criticality level
- 3 processors required
- Only promoting ensures a valid but inefficient scheduling
Feasible allocation with better performances

Reduction of criticality level 1

<table>
<thead>
<tr>
<th>Task</th>
<th>Period</th>
<th>Criticality</th>
<th>U(1)</th>
<th>U(2)</th>
<th>U(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>5</td>
<td>3</td>
<td>0.2</td>
<td>0.4</td>
<td>0.7</td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>5</td>
<td>3</td>
<td>0.2</td>
<td>0.3</td>
<td>0.5</td>
</tr>
<tr>
<td>$\tau_3$</td>
<td>5</td>
<td>2</td>
<td>0.1</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>$\tau_4$</td>
<td>10</td>
<td>1</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
</tr>
<tr>
<td>$\tau_5$</td>
<td>20</td>
<td>1</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
</tr>
<tr>
<td>$\tau_6$</td>
<td>40</td>
<td>1</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
</tr>
</tbody>
</table>

More realistic case: some allocations are possible
Feasible allocation with better performances

Reduction of criticality level 1

More realistic case: some allocations are possible

<table>
<thead>
<tr>
<th>Task</th>
<th>Period</th>
<th>Criticality</th>
<th>U(1)</th>
<th>U(2)</th>
<th>U(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>τ₁</td>
<td>5</td>
<td>3</td>
<td>0.2</td>
<td>0.4</td>
<td>0.7</td>
</tr>
<tr>
<td>τ₂</td>
<td>5</td>
<td>3</td>
<td>0.2</td>
<td>0.3</td>
<td>0.5</td>
</tr>
<tr>
<td>τ₃</td>
<td>5</td>
<td>2</td>
<td>0.1</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>τ₄</td>
<td>10</td>
<td>1</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
</tr>
<tr>
<td>τ₅</td>
<td>20</td>
<td>1</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
</tr>
<tr>
<td>τ₆</td>
<td>40</td>
<td>1</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
</tr>
</tbody>
</table>
Feasible allocation with better performances

Reduction of criticality level 1

More realistic case: some allocations are possible
Feasible allocation with better performances

Reduction of criticality level 1

<table>
<thead>
<tr>
<th>Task</th>
<th>Period</th>
<th>Criticality</th>
<th>U(1)</th>
<th>U(2)</th>
<th>U(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>5</td>
<td>3</td>
<td>0.2</td>
<td>0.4</td>
<td>0.7</td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>5</td>
<td>3</td>
<td>0.2</td>
<td>0.3</td>
<td>0.5</td>
</tr>
<tr>
<td>$\tau_3$</td>
<td>5</td>
<td>2</td>
<td>0.1</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>$\tau_4$</td>
<td>10</td>
<td>1</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
</tr>
<tr>
<td>$\tau_5$</td>
<td>20</td>
<td>1</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
</tr>
<tr>
<td>$\tau_6$</td>
<td>40</td>
<td>1</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
</tr>
</tbody>
</table>

More realistic case: some allocations are possible
### Feasible allocation with better performances

#### Reduction of criticality level 1

<table>
<thead>
<tr>
<th>Task</th>
<th>Period</th>
<th>Criticality</th>
<th>U(1)</th>
<th>U(2)</th>
<th>U(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\tau_1)</td>
<td>5</td>
<td>3</td>
<td>0.2</td>
<td>0.4</td>
<td>0.7</td>
</tr>
<tr>
<td>(\tau_2)</td>
<td>5</td>
<td>3</td>
<td>0.2</td>
<td>0.3</td>
<td>0.5</td>
</tr>
<tr>
<td>(\tau_3)</td>
<td>5</td>
<td>2</td>
<td>0.1</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>(\tau_4)</td>
<td>10</td>
<td>1</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
</tr>
<tr>
<td>(\tau_5)</td>
<td>20</td>
<td>1</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
</tr>
<tr>
<td>(\tau_6)</td>
<td>40</td>
<td>1</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
</tr>
</tbody>
</table>

- More realistic case: some allocations are possible
Feasible allocation with better performances

Reduction of criticality level 2

More realistic case: some allocations are possible

<table>
<thead>
<tr>
<th>Task</th>
<th>Period</th>
<th>Criticality</th>
<th>U(1)</th>
<th>U(2)</th>
<th>U(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>5</td>
<td>3</td>
<td>0.2</td>
<td>0.4</td>
<td>0.7</td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>5</td>
<td>3</td>
<td>0.2</td>
<td>0.3</td>
<td>0.5</td>
</tr>
<tr>
<td>$\tau_3$</td>
<td>5</td>
<td>2</td>
<td>0.1</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>$\tau_5^+$</td>
<td>20</td>
<td>2</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
</tr>
<tr>
<td>$\tau_6^+$</td>
<td>40</td>
<td>2</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
</tr>
</tbody>
</table>

More realistic case: some allocations are possible
### Feasible allocation with better performances

#### Reduction of criticality level 2

<table>
<thead>
<tr>
<th>Task</th>
<th>Period</th>
<th>Criticality</th>
<th>U(1)</th>
<th>U(2)</th>
<th>U(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\tau_1)</td>
<td>5</td>
<td>3</td>
<td>0.2</td>
<td>0.4</td>
<td>0.7</td>
</tr>
<tr>
<td>(\tau_2)</td>
<td>5</td>
<td>3</td>
<td>0.2</td>
<td>0.3</td>
<td>0.5</td>
</tr>
<tr>
<td>(\tau_3)</td>
<td>5</td>
<td>2</td>
<td>0.1</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>(\tau_5^+)</td>
<td>20</td>
<td>2</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
</tr>
<tr>
<td>(\tau_6^+)</td>
<td>40</td>
<td>2</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
</tr>
</tbody>
</table>

- More realistic case: some allocations are possible
Feasible allocation with better performances

Reduction of criticality level 2

More realistic case: some allocations are possible
### Feasible allocation with better performances

**Reduction of criticality level 2**

- More realistic case: some allocations are possible

<table>
<thead>
<tr>
<th>Task</th>
<th>Period</th>
<th>Criticality</th>
<th>U(1)</th>
<th>U(2)</th>
<th>U(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>5</td>
<td>3</td>
<td>0.2</td>
<td>0.4</td>
<td>0.7</td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>5</td>
<td>3</td>
<td>0.2</td>
<td>0.3</td>
<td>0.5</td>
</tr>
<tr>
<td>$\tau_3$</td>
<td>5</td>
<td>2</td>
<td>0.1</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>$\tau_5^+$</td>
<td>20</td>
<td>2</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
</tr>
<tr>
<td>$\tau_6^+$</td>
<td>40</td>
<td>2</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
</tr>
</tbody>
</table>

Diagram:

- $\tau_1$: $MS_1^2$
- $\tau_2$: $MS_2^2$
- $\tau_3$: $MS_3^2$
- $\tau_5^+$
- $\tau_6^+$

- $\tau_3^+$
- $\tau_3$
- $\tau_5^+$
- $\tau_6^+$

- Additional notes:
  - $\tau_1$, $\tau_2$, $\tau_3$, $\tau_5^+$, $\tau_6^+$
Feasible allocation with better performances

<table>
<thead>
<tr>
<th>Task</th>
<th>Period</th>
<th>Criticality</th>
<th>U(1)</th>
<th>U(2)</th>
<th>U(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>5</td>
<td>3</td>
<td>0.2</td>
<td>0.4</td>
<td>0.7</td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>5</td>
<td>3</td>
<td>0.2</td>
<td>0.3</td>
<td>0.5</td>
</tr>
<tr>
<td>$\tau_3^+$</td>
<td>5</td>
<td>3</td>
<td>0.1</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>$\tau_6^{++}$</td>
<td>40</td>
<td>3</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
</tr>
</tbody>
</table>

Final result

- More realistic case: some allocations are possible
### Feasible allocation with better performances

<table>
<thead>
<tr>
<th>Task</th>
<th>Period</th>
<th>Criticality</th>
<th>U(1)</th>
<th>U(2)</th>
<th>U(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>5</td>
<td>3</td>
<td>0.2</td>
<td>0.4</td>
<td>0.7</td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>5</td>
<td>3</td>
<td>0.2</td>
<td>0.3</td>
<td>0.5</td>
</tr>
<tr>
<td>$\tau_3^+$</td>
<td>5</td>
<td>3</td>
<td>0.1</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>$\tau_6^{++}$</td>
<td>40</td>
<td>3</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
</tr>
</tbody>
</table>

- More realistic case: some allocations are possible
- Task set schedulable on:
  - 2 processors with GMC-RUN
  - One processor saved

**Final result**

**RUN**

- $\tau_1$
- $\tau_2$
- $\tau_3^+$
- $\tau_6^{++}$
Feasible allocation with better performances

<table>
<thead>
<tr>
<th>Task</th>
<th>Period</th>
<th>Criticality</th>
<th>U(1)</th>
<th>U(2)</th>
<th>U(3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>5</td>
<td>3</td>
<td>0.2</td>
<td>0.4</td>
<td>0.7</td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>5</td>
<td>3</td>
<td>0.2</td>
<td>0.3</td>
<td>0.5</td>
</tr>
<tr>
<td>$\tau_3^+$</td>
<td>5</td>
<td>3</td>
<td>0.1</td>
<td>0.2</td>
<td>0.2</td>
</tr>
<tr>
<td>$\tau_6^{++}$</td>
<td>40</td>
<td>3</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
</tr>
</tbody>
</table>

Final result

- More realistic case: some allocations are possible
- Task set schedulable on:
  - 2 processors with GMC-RUN
  - One processor saved

RUN

More realistic case: some allocations are possible
- Task set schedulable on:
  - 2 processors with GMC-RUN
  - One processor saved

RUN
**Conclusion and future work**

**GMC-RUN:**
- Handles more than two criticality levels
- RUN working preserved: pre-emption performance preserved
- Performs better than fpEDF-VD and is close to MC-Fluid

**Future work:**
- Experiments on systems with more than two criticality levels
- Implementation in Litmus$^{\text{RT}}$
Thank You !


