Implementation and Evaluation of Multi-Mode Real-Time Tasks under Different Scheduling Algorithms

Anas Toma, Vincent Meyers and Jian-Jia Chen
TU Dortmund University, Germany

Supported by DFG as part of the Collaborative Research Center SFB876

03.07.2018
Introduction - Automotive Systems

Electronic Control Units (ECUs)

- Control and improve functionalities, performance and safety

- Continuous interaction with components
  - Doors, lights, **engine**, etc.

- Should react within a specific amount of time
  - A delayed reaction may affect the safety
Engine Control

- **Tasks:**
  - Adjusting the fuel flow
  - Calculating the time of the spark signal
  - Minimizing fuel consumption and emissions

- **Angular synchronous tasks**
  - Linked to the rotation of the crankshaft
  - Increasing rotation speed → Shorter period/deadline
    - Drop some non-critical functions to meet the deadline
  - Releases jobs depending on the engine’s rotation speed
    - Different execution modes → *Multi-Mode Task Model*
      - Digraph Real-Time model (DRT)
      - Variable Rate-dependent Behavior (VRB) task model
Multi-Mode Tasks

An example of a multi-mode task with three different execution modes

<table>
<thead>
<tr>
<th>Rotation Speed (rpm)</th>
<th>Mode Type</th>
<th>Executed Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>[0, 3000]</td>
<td>A</td>
<td>f1, f2 and f3</td>
</tr>
<tr>
<td>(3000, 6000]</td>
<td>B</td>
<td>f1 and f2</td>
</tr>
<tr>
<td>(6000, 9000]</td>
<td>C</td>
<td>f1</td>
</tr>
</tbody>
</table>

- Different modes: \((C^1, T^1, D^1)\) \((C^2, T^2, D^2)\) \((C^3, T^3, D^3)\)
  - \(C^j\): worst-case execution time (WCET)
  - \(T^j\): period
  - \(D^j\): relative deadline

- Implicit deadline \(T^j = D^j\)

- The mode changes based on an external interrupt or any other event
The FreeRTOS Kernel

- A Real Time Operating System (RTOS) for microcontrollers and small microprocessors
- Supports many different architectures
- Open source RTOS
- Low ROM and RAM usage
- Simple and easy to use
- Can be also used for educational purposes

[https://www.freertos.org]
Contribution

- Modifying the FreeRTOS real-time operating system to consider the multi-mode real-time tasks

- Implementing the Rate-Monotonic (RM) and the Earliest Deadline First (EDF) scheduling algorithms

- Empirical evaluation of the multimode tasks under EDF and RM algorithms in a real environment
  - FreeRTOS running on Raspberry Pi B+ board
Multi-Mode Task Model Implementation

**Periodic tasks**
- Expanding the task control block (TCB)
  - Period, worst-case execution time, relative deadline and the previous wake time
- `vTaskDelayUntil()` function to delay the task for the specified period

**Modes**
- TCB fields with array data structure
- Additional attributes
  - number of the modes
  - threshold values for each mode level
- Global variable for the external input
  - Any changes will be applied starting from the next release
Rate-Monotonic (RM) Algorithm Implementation

- Tasks with a shorter period have a higher priority

- Assign priorities before starting the scheduler

- Doubly linked list to sort the tasks according to their periods

- The priorities are assigned for each task for all the modes
  - Array of priorities for each task

- The tasks are moved to their corresponding ready lists
Earliest Deadline First (EDF) Algorithm Implementation

- Assign the highest priority to the job with the earliest absolute deadline

- A doubly linked list for the ready jobs
  - Instead of the array of linked lists provided by FreeRTOS
  - Apply binary heap

- Once a job is added to the ready list
  - The absolute deadline is calculated
  - The job with the earliest absolute deadline is scheduled for execution
Scheduling in FreeRTOS

- Shared Processor Behavior (round-robin)
  - Context switching for every system tick \(\sim 4\mu s\) → additional overhead!
    - Two tasks with the same priority
    - one ready task

Additional Modifications

+ Tasks with the same priority are scheduled according to their insertion order in the ready list

+ Perform context switching only if
  - a new job with a higher priority arrives, or
  - the current job under execution is blocked
Experimental Evaluation - Synthetic Workload

- Utilizations and computation segments: [10%-100%]
  - Uniform distribution according to *UUniFast*

- Periods: [1-100ms]
  - Log uniform distribution

- For the multi-mode tasks, the WCET and the period values for the remaining modes were scaled by the factor of 1.5
  - $C_{i,m+1} = 1.5 \times C_{i,m}$
  - $T_{i,m+1} = 1.5 \times T_{i,m}$

- 100 task sets with 50% multi-mode tasks and cardinality of 10

Experimental Evaluation - Synthetic Workload

- 5 modes

![Graph showing the success ratio (%) against total utilization (%). The graph compares two methods: RM (red crosses) and EDF (blue circles). As total utilization increases, the success ratio for RM decreases significantly, while EDF maintains a high success ratio throughout.]
Experimental Evaluation – Scheduling overhead

- Cardinality: the number of tasks per a set
Experimental Evaluation – Realistic Workload

- Shared the characteristics of an automotive software system*
  - The distribution of the tasks among the periods
  - The typical number of the tasks
  - The average execution time
  - Factors for determining the best- and worst-case execution times

<table>
<thead>
<tr>
<th>Period</th>
<th>Share</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 ms</td>
<td>3 %</td>
</tr>
<tr>
<td>2 ms</td>
<td>2 %</td>
</tr>
<tr>
<td>5 ms</td>
<td>2 %</td>
</tr>
<tr>
<td>10 ms</td>
<td>25 %</td>
</tr>
<tr>
<td>20 ms</td>
<td>25 %</td>
</tr>
<tr>
<td>50 ms</td>
<td>3 %</td>
</tr>
<tr>
<td>100 ms</td>
<td>20 %</td>
</tr>
<tr>
<td>200 ms</td>
<td>1 %</td>
</tr>
<tr>
<td>1000 ms</td>
<td>4 %</td>
</tr>
<tr>
<td>angle-synchronous ms</td>
<td>15 %</td>
</tr>
</tbody>
</table>

Task distribution among periods

6 modes ranging from 0 to 6000 rpm with their periods in milliseconds

<table>
<thead>
<tr>
<th>Mode</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min.</td>
<td>0</td>
<td>1001</td>
<td>2001</td>
<td>3001</td>
<td>4001</td>
<td>5001</td>
</tr>
<tr>
<td>Max</td>
<td>1000</td>
<td>2000</td>
<td>3000</td>
<td>4000</td>
<td>5000</td>
<td>6000</td>
</tr>
<tr>
<td>Period</td>
<td>30</td>
<td>15</td>
<td>10</td>
<td>7.5</td>
<td>6</td>
<td>5</td>
</tr>
</tbody>
</table>

[S. Kramer, D. Ziegenbein, and A. Hamann. Real world automotive benchmarks for free]
Experimental Evaluation – Realistic Workload

![Graph showing the success ratio (%) against total utilization (%). The graph compares RM and EDF scheduling algorithms.](image-url)
Conclusion

- Multi-mode tasks were evaluated under the EDF and the RM scheduling algorithms in a real environment
  - FreeRTOS real-time operating system was modified
  - Raspberry Pi B+ board
  - Synthetic and realistic data sets

- **Synthetic workload:** The EDF algorithm was able to find more feasible schedules than the RM algorithm
  - for high utilization values

- **Realistic workload:** EDF performed poorly
  - Scheduling overhead of EDF
  - Tasks with shorter periods