Paper “RM/EDF”

• The correctness of the system
  – Logical/functional
  – Temporal
• RT computing
  – The objective of “fast computing” is to minimize the average response time
  – The objective of real-time computing is to meet the individual timing requirement of each task
Paper “RM/EDF”

- Hard vs. soft real-time
- Closed-loop control

![Diagram of a control system with A/D and D/A converters, sensor, plant, and actuator]

Outside effects

The system being controlled

Paper “RM/EDF”

- Job
  - Each unit of work that is scheduled and executed by the system
- Task
  - A set of related jobs
  - For example, a periodic task Ti consists of jobs J1, J2, J3, ... coming at every period
- Release time
  - Time instant at which a job becomes available for execution
  - It can be executed at any time at or after the release time
- Deadline
  - Time instant by which a job should be finished
  - Relative deadline: Maximum allowable response time
  - Absolute deadline = release time + relative deadline
Paper “RM/EDF”

- Periodic task $T_i$
  - Period $P_i$
  - Worst case execution time $C_i$
  - Relative deadline $D_i$

- Job $J_{ik}$
  - Absolute deadline = release time + relative deadline
  - Response time = finish time – release time

- Deadline miss if
  - Finish time > absolute deadline
  - Response time of $J_{ik}$ > $D_i$

**Periodic Task Model**

Task = \{T, C, D\}

jobs (j1, j2, j3, ...)

Period = T

Deadline = D

Computation time WCET = C

Release Time
Paper “RM/EDF”

- Table-driven scheduling
- Jitter
- Hyperperiods

Paper “RM/EDF”

- A scheduling algorithm S is optimal if S cannot schedule a real-time task set T, no other scheduling algorithm can schedule T
- E.g., Rate Monotonic & EDF
Common Assumptions

- Single processor
- Every task is periodic
- Deadline = period
- Tasks are independent
- WCET of each task is known
- Zero context switch time

Paper “RM/EDF”

- Fixed priority system
  - Assign the same priority to all the jobs in each task
  - Rate monotonic (RMS)
- Dynamic priority system
  - Assign different priorities to the individual jobs in each task
  - Earliest Deadline First (EDF)
Paper “RM/EDF”

- RMS: optimal *fixed* priority scheduling algorithm
- Shorter period $\rightarrow$ Higher priority
  - Higher rate $\rightarrow$ higher priority
- Utilization bound

$$U = \sum_{i=1}^{n} C_i / T_i \leq n(\sqrt{2} - 1)$$

$$\lim_{n \to \infty} n(\sqrt{2} - 1) = \ln 2 \approx 0.693147 \ldots$$

RMS (Rate Monotonic Scheduling)

Process $P_1$: service time = 20, period = 50, deadline = 50
Process $P_2$: service time = 35, period = 100, deadline = 100
Missed Deadlines with RMS

Process $P_1$: service time = 25, period = 50, deadline = 50
Process $P_2$: service time = 35, period = 80, deadline = 80

RMS is guaranteed to work if

$N = \text{number of processes}$

sufficient condition

$$\sum_{i \in P} \frac{f_i}{P_i} \leq N \left( \sqrt{2} - 1 \right);$$

$$\lim_{N \to \infty} N \left( \sqrt{2} - 1 \right) = \ln 2 \approx 0.693147$$

<table>
<thead>
<tr>
<th>$N$</th>
<th>$N \left( \sqrt{2} - 1 \right)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.828427</td>
</tr>
<tr>
<td>3</td>
<td>0.779763</td>
</tr>
<tr>
<td>4</td>
<td>0.756828</td>
</tr>
<tr>
<td>5</td>
<td>0.743491</td>
</tr>
<tr>
<td>10</td>
<td>0.717734</td>
</tr>
<tr>
<td>20</td>
<td>0.705298</td>
</tr>
</tbody>
</table>

Paper “RM/EDF”

- EDF: shorter absolute deadline $\rightarrow$ Higher priority
- Utilization bound $U_b = 1$
- $U_b$ is necessary and sufficient
**EDF (Earliest Deadline First)**

Process $P_1$: service time = 25, period = 50, deadline = 50  
Process $P_2$: service time = 35, period = 80, deadline = 80

---

**Paper “RM/EDF”**

- **RMS**
  - RMS may not guarantee schedulability even when $U < 1$
  - Low overhead: priorities do not change for a fixed task set
- **EDF**
  - EDF guarantee schedulability as long as $U \leq 1$
  - High overhead: task priorities may change dynamically
Paper “RM/EDF”

- Implementation complexity
  - Modifying systems vs. from scratch
  - Periods for newly arriving tasks
  - Fixed vs. infinite number of priority levels
  - EDF runtime overheads (priorities change)

  - Winner: RMS

Paper “RM/EDF”

- Run-time overhead
  - Updating deadlines costly
  - EDF: fewer context switches (preemptions)

![Diagram of preemptions introduced by RM (a) and EDF (b) on a set of two periodic tasks. Adjacent jobs of t_3 are depicted with different colours to better distinguish them.](image)
Paper “RM/EDF”

• Run-time overhead

![Graph 1](image1.png)  
![Graph 2](image2.png)

Figure 2: Preemptiveness introduced by RM and EDF as a function of the number of tasks.

Figure 3: Preemptiveness introduced by RM and EDF as a function of the load.

Paper “RM/EDF”

• Run-time overhead

• Winner: EDF
Paper “RM/EDF”

• Schedulability analysis
  – EDF (d=p): simple
  – RMS: U <= 0.69; simple, but resources wasted
    • Hyperbolic bound (higher acceptance ratio for large n)
  – Exact for EDF:
    • Processor Demand Criterion (PDC) for d<p
  – Exact for RMS:
    • Response Time Analysis (RTA)

Paper “RM/EDF”

• Schedulability analysis

• Winner: Tie?
Paper “RM/EDF”

- Robustness during overloads
  - Permanent

  ![Image](image.png)

  Figure 8. Schedules produced by EDF (a) and RM (b) for 4 out of 12 periodic tasks in a permanent overload condition.

  — Winner: RMS

Paper “RM/EDF”

- Robustness during overloads
  - Transient

  ![Image](image.png)

  Figure 9. Under overloads, only the highest priority task is protected under RM, but nothing can be ensured for the other tasks.

  — Winner: Tie
Paper “RM/EDF”

• Jitter and Latency

![Image](image.png)

• Winner: Tie?

Paper “RM/EDF”

• Resource sharing
  – Solutions for EDF and RMS exist

• Aperiodic tasks
  – Periodic servers (EDF has higher utilization bounds)

• Resource reservations
  – Reservation protocols exist for EDF and RMS