Graduate Operating Systems
(Embedded Systems & Scheduling)

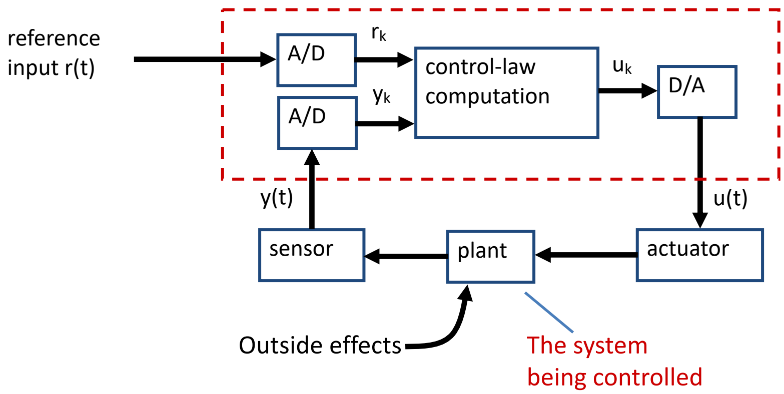
Fall 2020

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



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

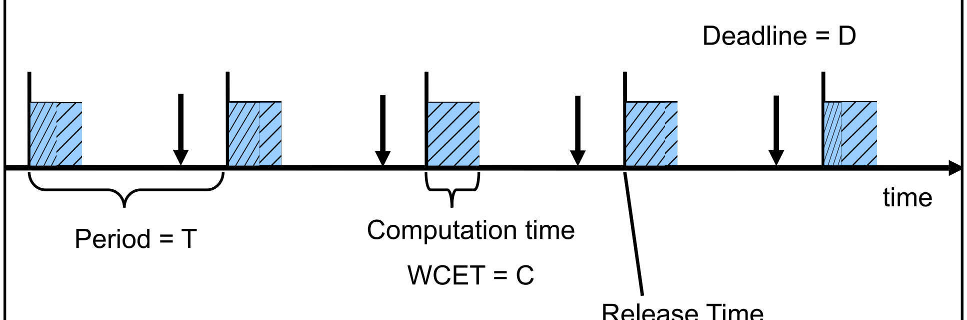
Paper "RM/EDF"

- 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 (j_1, j_2, j_3, …)



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 → Higher priority
  - Higher rate → higher priority
- Utilization bound

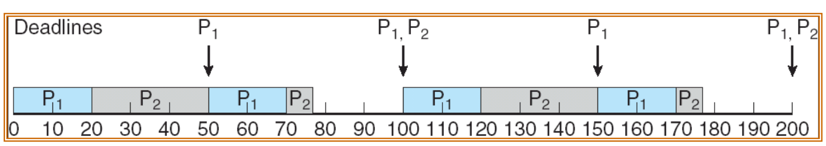
$$U = \sum_{i=1}^{n} C_i/T_i \leq n(\sqrt[n]{2} - 1)$$

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

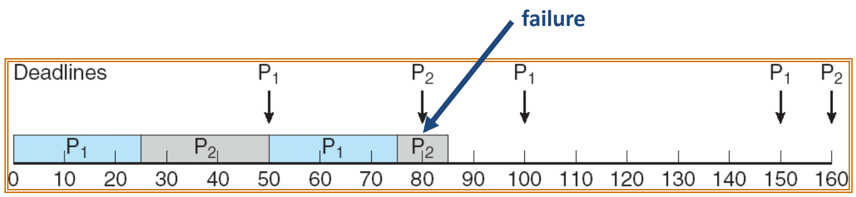
RMS (Rate Monotonic Scheduling)

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 = number of processes

sufficient condition

$$u = \sum_{i=1}^{N} \frac{t_i}{p_i} \le N\left(\sqrt[N]{2} - 1\right);$$

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

N	$N\left(\sqrt[N]{2} - 1\right)$
2	0,828427
3	0,779763
4	0,756828
5	0,743491
10	0,717734
20	0,705298

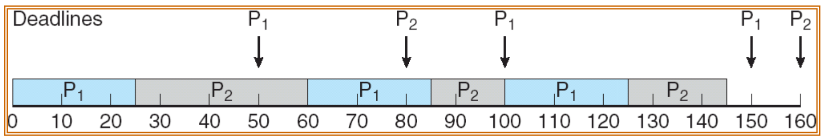
Paper “RM/EDF”

- EDF: shorter absolute deadline → 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 <= 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)

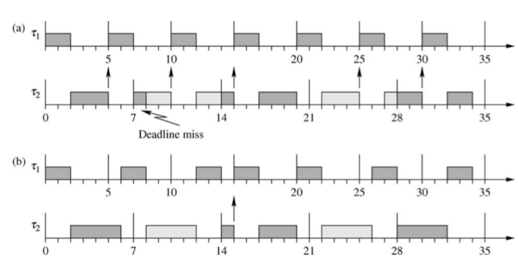


Figure 1. Preemptions introduced by RM (a) and EDF (b) on a set of two periodic tasks. Adjacent jobs of τ_2 are depicted with different colours to better distinguish them.

Paper “RM/EDF”

- Run-time overhead

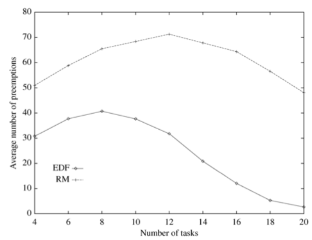


Figure 2. Preemptions introduced by RM and EDF as a function of the number of tasks.

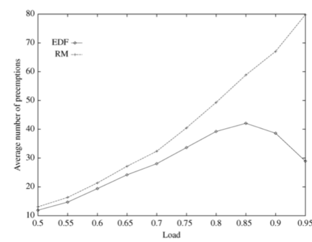


Figure 3. Preemptions introduced by RM and EDF on a set of 10 periodic tasks as a function of the load.

Paper “RM/EDF”

- Run-time overhead

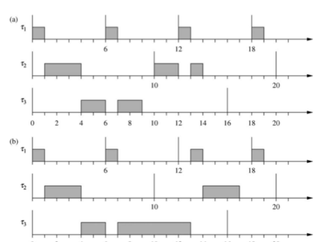


Figure 4. Under EDF, the number of preemptions may decrease when execution times increase: in case (a), where C_3 is small, τ_2 is preempted by τ_1, but this does not occur in case (b), where τ_3 has a higher execution time.

- 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

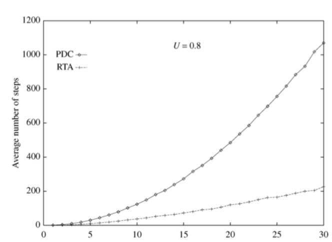


Figure 7. Average number of steps required for the RTA and for the PDC as a function of the number of tasks.

- Winner: Tie?

Paper "RM/EDF"

- Robustness during overloads
  - Permanent

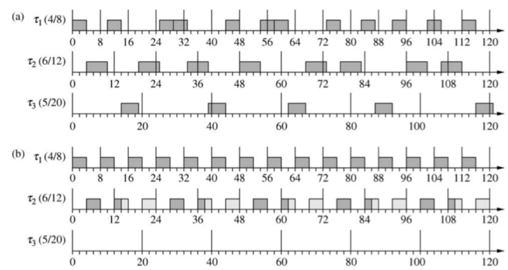


Figure 8. Schedules produced by EDF (a) and RM (b) for a set of three periodic tasks in a permanent overload condition.

  - Winner: RMS

Paper "RM/EDF"

- Robustness during overloads
  - Transient

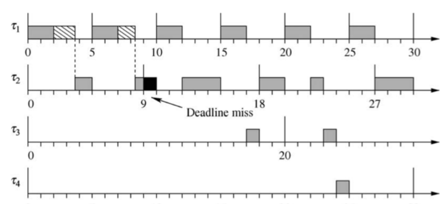


Figure 9. Under overloads, only the highest priority task is protected under RM, but nothing can be ensured for the other tasks.

  - Winner: Tie

Paper “RM/EDF”

- Jitter and Latency

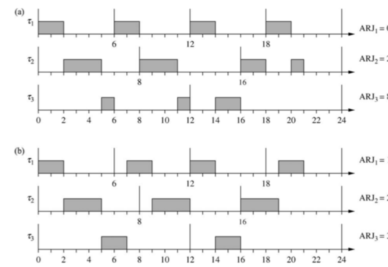


Figure 10. Response time jitter introduced under RM (a) and under EDF (b).

- 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