Rate Monotonic vs EDF -Judgment Day

Work by: GIORGIO C. BUTTAZZO, University of Pavia, Italy

Real-Time Systems, 29, 5–26, 2005 c 2005 Springer Science + Business Media, Inc. Manufactured in The Netherlands.

What is this project about?

- Periodic Tasks
- Scheduling Algorithms
- Misconceptions

Why do we need this comparison?

Implementation Complexity

EDF implementation is not easy?

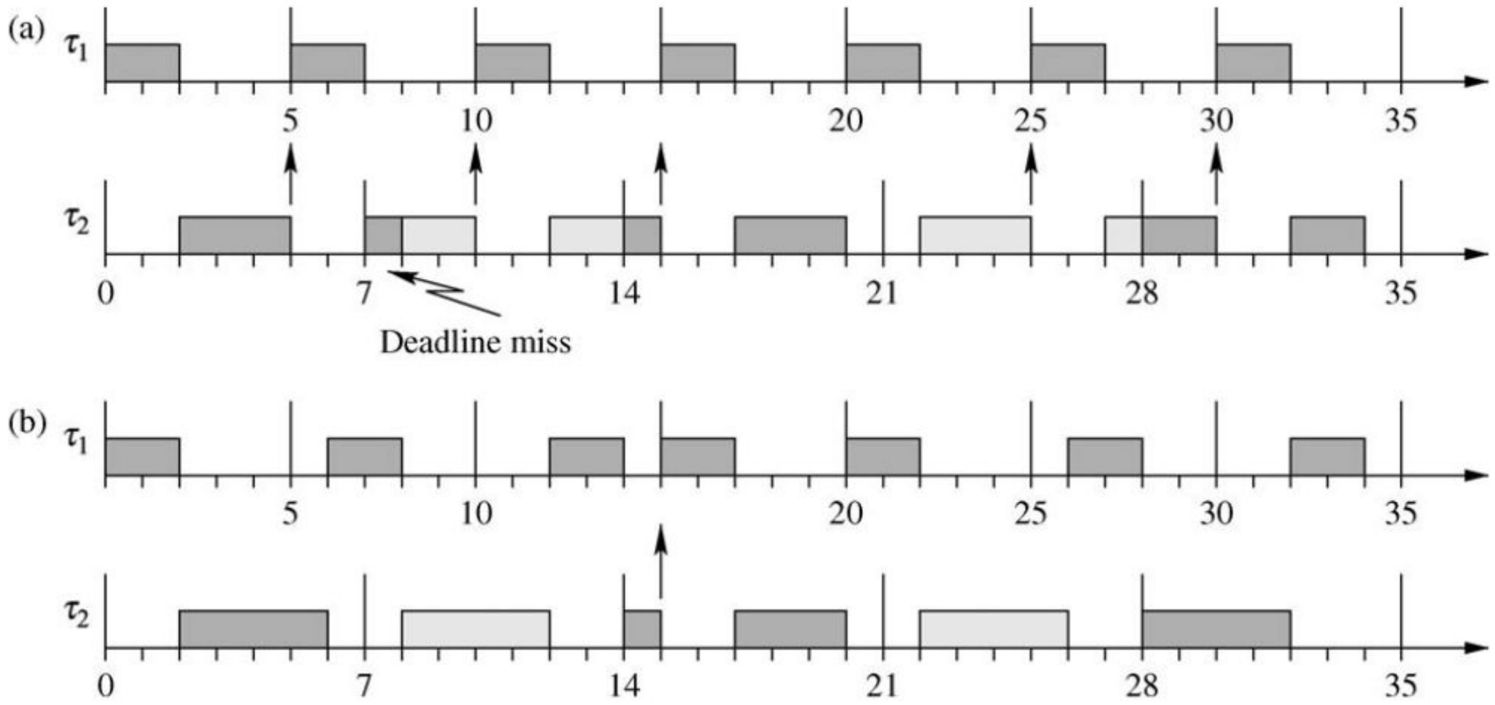
1. Developing on top on generic priority based OS (kernel settings are given)
   a. Kernel with fixed priority level: **True**
   b. Kernel which allows priority change during runtime: **Mostly true/Case dependent**
2. Algorithm is developed from scratch: Kernel using a list for ready queue : **Very close to RM**
   a. RM: ready queue is ordered by decreasing fixed priority levels
   b. EDF: ready queue has to be ordered by increasing absolute deadline

Runtime Overhead

EDF introduces a larger runtime overhead than RM?

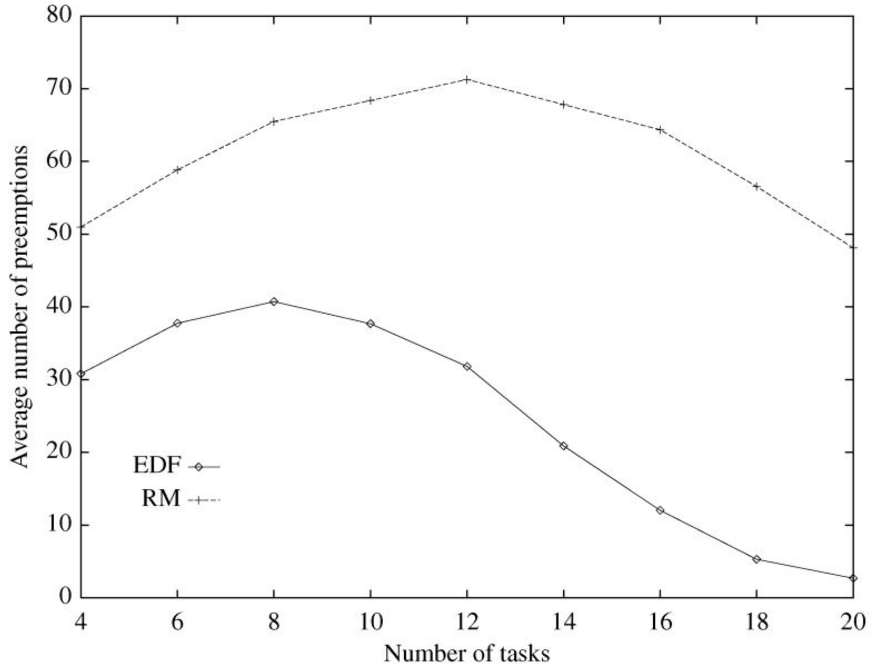
True that, under EDF, deadlines need to be updated by the kernel at each job activation

However, EDF introduces **less runtime overhead** than RM, when **context switches** are taken into account. In fact, to enforce the fixed priority order, the number of preemptions that typically occur under RM is much higher than under EDF.



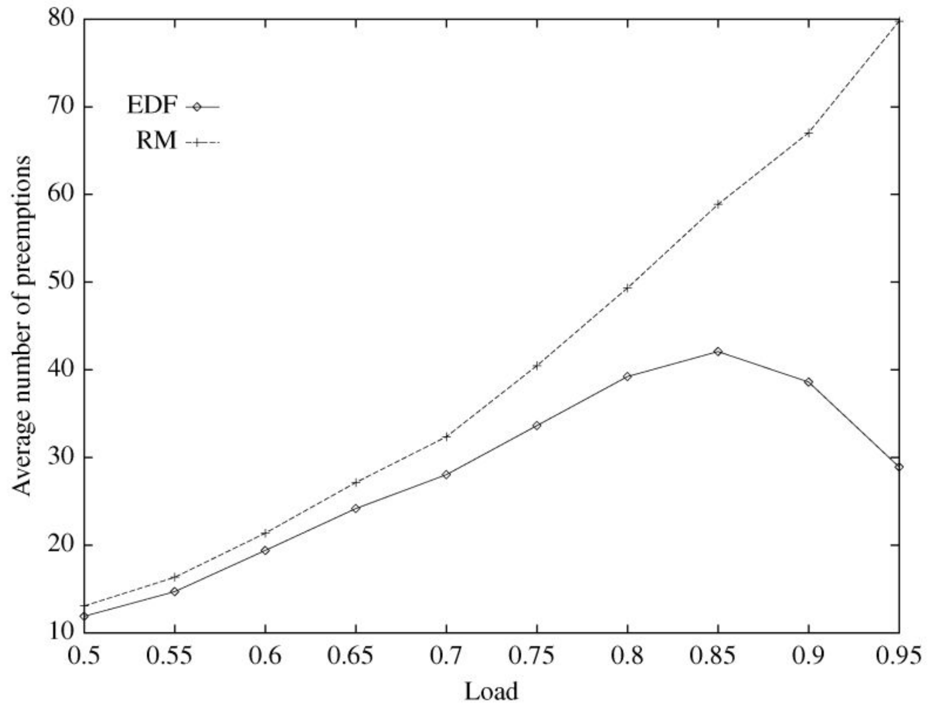
Preemptions introduced by **(a) RM** and **(b) EDF** on a set of two periodic tasks

*Adjacent jobs of τ 2 are depicted with different colours to better distinguish them



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

Each point in the graph, the average was computed over 1000 independent simulations, each running for 1000 units of time. In each simulation, periods were generated as random variables with uniform distribution in the range of 10 to 100 units of time, whereas execution times were computed to create a total processor utilization U = 0.9



Preemptions introduced by RM and EDF as a function of the load

The behavior of RM and EDF as a function of the processor load, for a fixed number of tasks. Figure shows the average number of preemptions as a function of the load for a set of **10 periodic tasks**. Periods and computation times were generated with the same criterion used in the previous experiment, but to create an average load ranging from 0.5 to 0.95

Schedulability Analysis

Assumptions(Liu and Layland, 1973): Taskset of n periodic tasks,where all tasks start simultaneously at time t = 0, relative deadlines are equal to periods and tasks are independent (that is, they do not have resource constraints, nor precedence relations)

A set of n periodic tasks is schedulable by the RM algorithm if

$$\sum_{i=1}^{n} U_i \leq n\,(2^{1/n} - 1)$$

A set of n periodic tasks is schedulable by the EDF algorithm if and only if

$$\sum_{i=1}^{n} U_i \leq 1$$

Schedulability Analysis

For RM, The worst-case response time Ri of a task can be computed using the Response Time Analysis' (RTA) iterative formula:

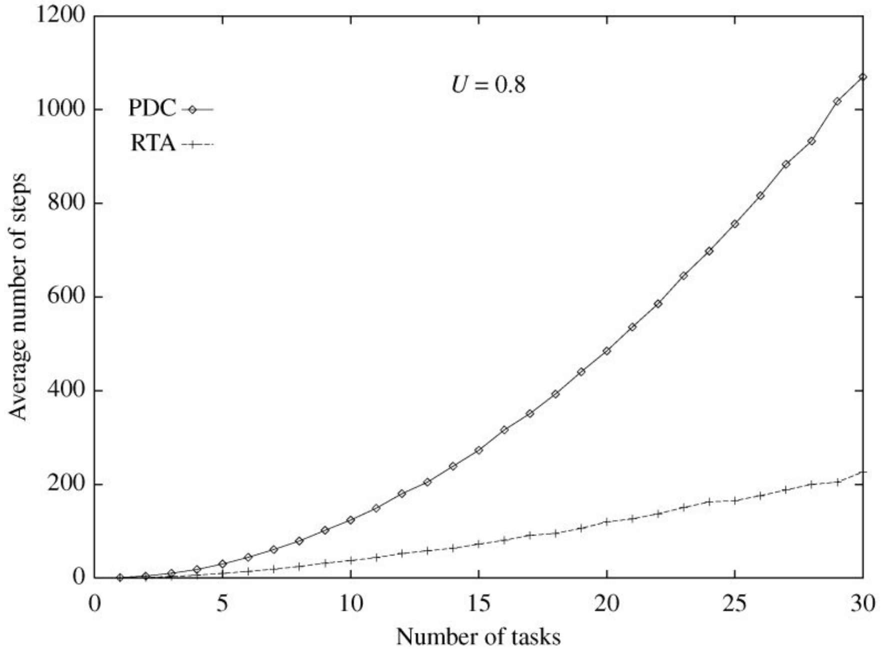
$$\begin{cases} R_i^{(0)} = C_i \\ R_i^{(k)} = C_i + \sum_{j:D_j<D_i} \left\lceil \frac{R_i^{(k-1)}}{T_j} \right\rceil C_j \end{cases}$$

For EDF, the schedulability analysis of periodic tasks with relative deadlines less than periods can be performed using the Processor Demand Criterion PDC

$$\forall L > 0, \quad \sum_{i=1}^{n} \left\lfloor \frac{L + T_i - D_i}{T_i} \right\rfloor C_i \le L$$

$$\forall L \in \mathcal{D}, \quad \mathcal{D} = \{d_k : d_k < \min(L^*, H)\}$$

$$L^* = \frac{\sum_{i=1}^{n} U_i(T_i - D_i)}{1 - U}.$$



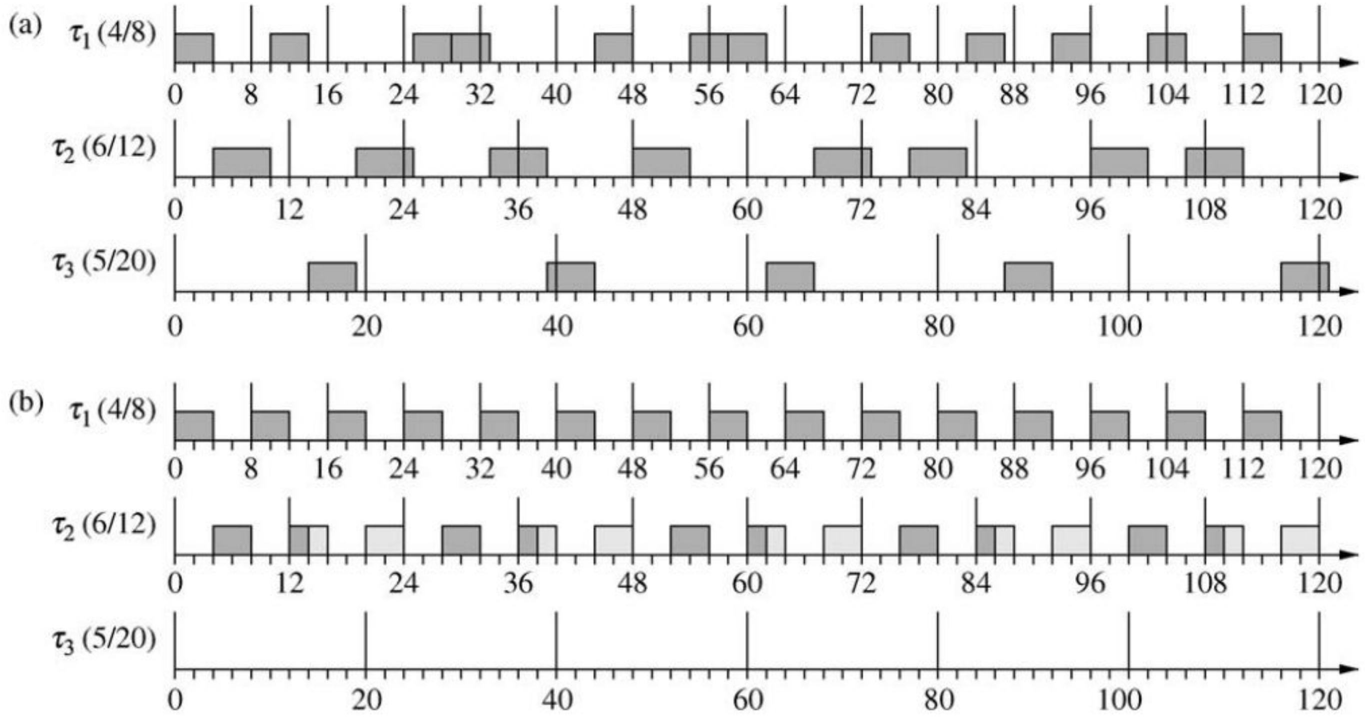
Average number of steps required for the RTA and for the PDC as a function of the number of tasks

The tests were performed on randomly generated task sets with periods T i uniformly distributed in [10, 200], relative deadlines uniformly distributed in [T i /2, T i], and utilization factor U = 0.8. The average was computed on 1000 samples.

Robustness: Permanent Overload

Theorem[Cervin]- *Assume a set of n periodic tasks, where each task is described by a fixed period T_i, a fixed execution time C_i, a relative deadline D_i, and a release offset i. If $U > 1$ and tasks are scheduled by EDF, then, in stationarity, the average period T_i of each task τ_i is given by $T_i = T_i U$.*

It says, EDF during permanent overload automatically performs a period rescaling, and tasks start behaving as they were executing at a lower rate



Schedules produced by **(a)EDF** and **(b)RM** for a set of 3 periodic tasks in a permanent overload condition

In conclusion, under permanent overload conditions both the behaviors of RM and EDF are **predictable**, but, deciding which one is better is highly **application dependent**

Robustness: Transient Overload

For RM, in the presence of transient overload conditions, deadlines are missed predictably?

NOT TRUE! Neither for RM nor for EDF

In conclusion, under RM, if the system becomes overloaded, any task, except the highest priority task, can miss its deadline, independently of its period.

The only difference between RM and EDF is that, under RM, an overrun in task Ti cannot cause tasks with higher priority to miss their deadlines, whereas under EDF any other task could miss its deadline. However, such a property of RM can be of little use if we do not know a priori which task is going to overrun.

Jitter

Variation in finish time of job(from start of period) is called Jitter. It is classified into two types:

- Relative Response time Jitter (RRJ)

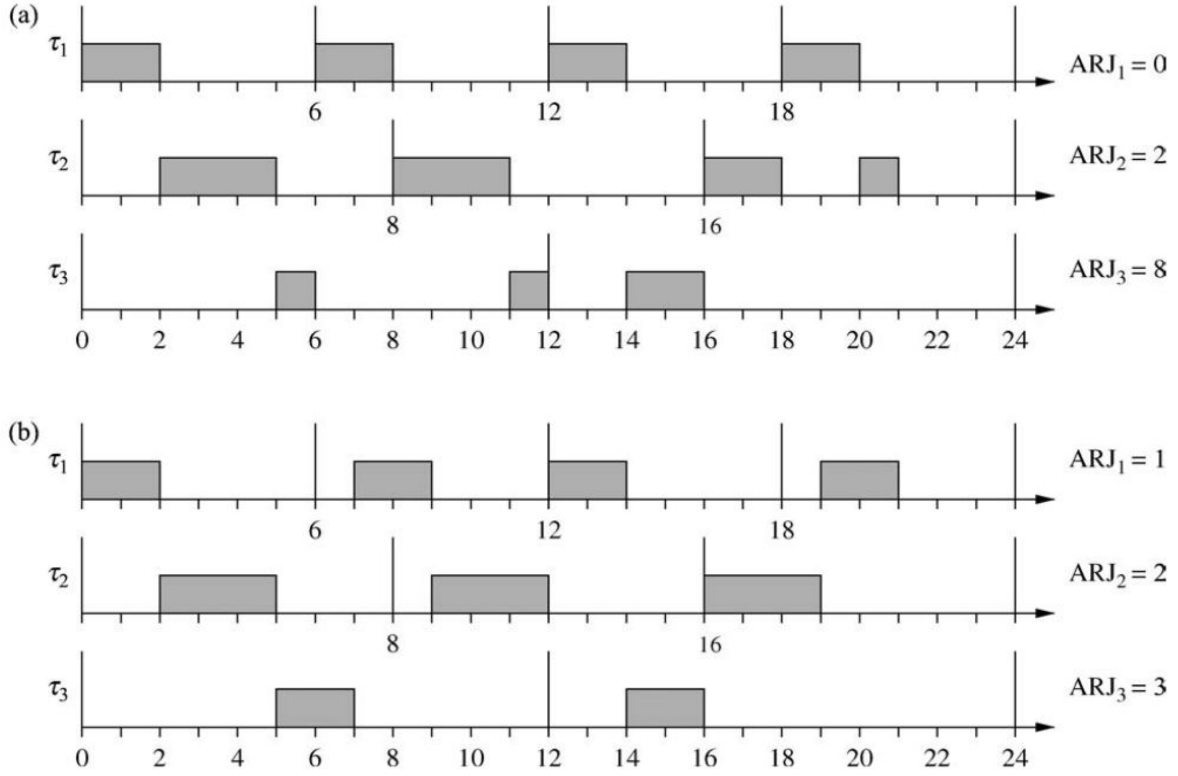
  $$RRJ_i = \max |R_{i,k+1} - R_{i,k}|$$

- Absolute Response time Jitter (ARJ_i)

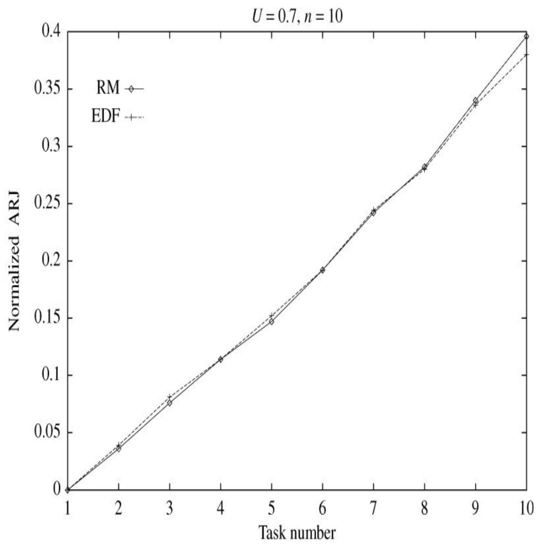
  $$ARJ_i = \max R_{i,k} - \min R_{i,k}$$

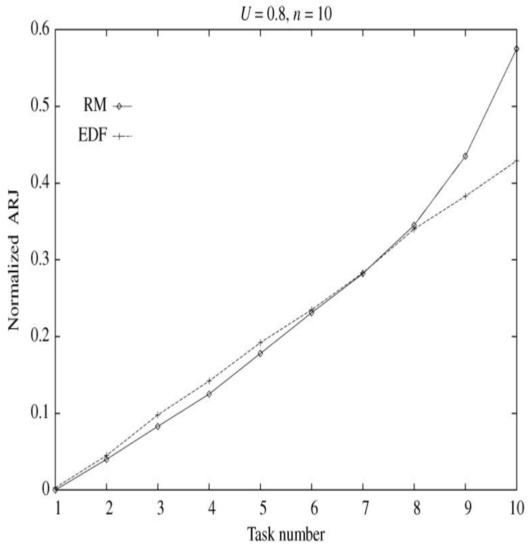
The fixed priority assignment used in RM reduces the jitter during task execution, more than under EDF?

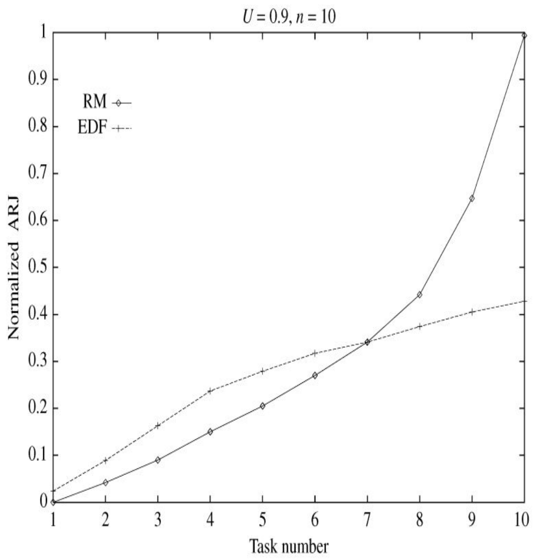
Clearly, this is true for the highest priority task, but this property does **not** hold in general.



Response time jitter introduced under **(a)RM** and under **(b)EDF**







Average normalized jitter for (a)U = 0.7, (b)U = 0.8, and (c)U = 0.9

The experiment has been performed to verify the jitter behavior under the two scheduling algorithms. Task sets of 10 periodic tasks were randomly generated with periods uniformly distributed in [10,200] and fixed total utilization U . The results refer to the Absolute Response-time Jitter (ARJ), which has been normalized with respect to task periods. Hence a value of 1 on task τ i corresponds to a jitter equal to its period T i .
The results reported in figures,show the jitter introduced by the algorithms for each individual task (tasks are ordered by increasing periods).

Latency

Time taken between input and output is called latency

$$L_i = \max(f_{i,k} - s_{i,k})$$

Theorem *[Cervin]- Given a set of n periodic control tasks performing input at the beginning of each job and output at the end, the maximum input–output latency of each task under EDF is shorter than or equal to the corresponding maximum latency under RM.*

In other words, EDF can **always** achieve a shorter input–output latency than RM, for any task.

Other issues: Resource Sharing

Only RM can be predictably used and analyzed in the presence of shared resources? **NO!**

If critical sections are accessed through classical semaphores, then tasks may experience long delays due to a phenomenon known as priority inversion

The problem can be solved by using concurrency control protocols, such as the PIP or the PCP.

These were initially designed for RM only.

Protocols for accessing shared resources:

1. RM
   a. PIP : Priority Inheritance Protocol
   b. PCP : Priority Ceiling Protocol
2. EDF
   a. DPI : Dynamic Priority Inheritance
   b. DCP : Dynamic Priority Ceiling
   c. DDM : Dynamic Deadline Modification
   d. SRP : Stack Resource Policy

Other issues: Aperiodic Task Handling

The objective is to reduce the aperiodic response times as much as possible, still guaranteeing that all periodic tasks complete within their deadlines.

RM	: Deferrable Server, Sporadic Server, and Slack Stealing Algorithm
EDF	: Total Bandwidth Server

The **superior performance of EDF**-based methods in aperiodic task handling comes from the higher processor utilization bound.

Other issues: Aperiodic Task Handling

***Theorem** [Tia-Liu-Shankar]. For any set of periodic tasks ordered on a given fixed-priority scheme and aperiodic requests ordered according to a given aperiodic queueing discipline, there does not exist any valid algorithm that <u>minimizes the response time</u> of every soft aperiodic request.*

***Theorem** [Tia-Liu-Shankar]. For any set of periodic tasks ordered on a given fixed-priority scheme and aperiodic requests ordered according to a given aperiodic queueing discipline, there does not exist any online valid algorithm that <u>minimizes the average response time</u> of the soft aperiodic requests.*

Conclusions

The real advantage of RM with respect to EDF is its simpler implementation in commercial kernels that do not provide explicit support for timing constraints, such as periods and deadlines. Other properties typically claimed for RM, such as predictability during overload conditions, or better jitter control, only apply for the highest priority task, and do not hold in general.

On the other hand, EDF allows a full processor utilization, which implies a more efficient exploitation of computational resources and a much better responsiveness of aperiodic activities. These properties become very important for embedded systems working with limited computational resources, and for multimedia systems, where quality of service is controlled through resource reservation mechanisms that are much more efficient under EDF. In fact, most resource reservation algorithms are implemented using service mechanisms similar to aperiodic servers, which have better performance under EDF.

Finally, both RM and EDF are not very well suited to work in overload conditions and to achieve jitter control.

My thoughts

Nature of scheduler should be kept in mind while presenting inferences

Thanks!

Presented by:

Sridhar M
2018111021

sridhar.m@research.iiit.ac.in