Scheduling mixed tasks

Date	@October 24, 2024

Priority servers

In real application heterogeneous task sets are used (it meas both periodic and aperiodic tasks):

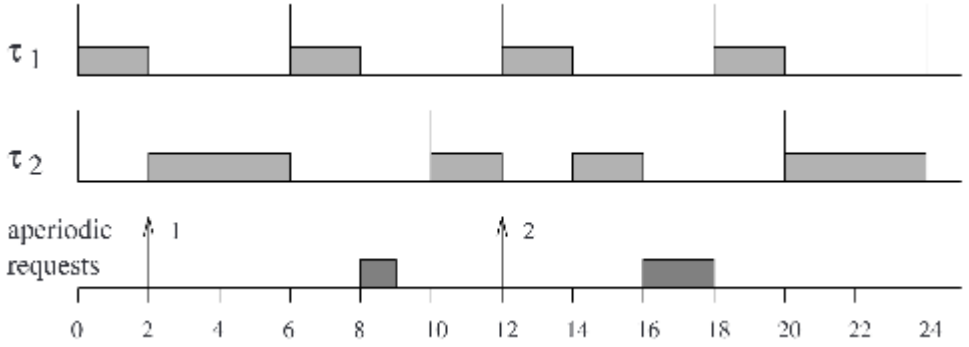
- periodic tasks are time-driven and execute critical control activities with hard timing constraints aimed at guaranteeing regular activation rates
- aperiodic tasks are event-driven and may have hard, soft or non-rel-time requirements depending on the specific application.

For real applications specific scheduling algorithms are used:

- background scheduling
- fixed priority servers, in which periodic tasks are scheduled based on a fixed-priority assignment and all start simultaneously at time t=0 with relative deadlines equal to their period; arrival times of aperiodic tasks are unknown instead, and the minimum interval arrival time of a sporadic task is assumed to be equal to its deadline; all tasks are pre-emptible.
- dynamic priority servers, in which periodic tasks are scheduled based on a dynamic-priority assignment working as before

Background scheduling

In background scheduling periodic tasks are scheduled using RM (rate monotonic scheduling), while aperiodic tasks are scheduled in background, when no periodic instance is ready.



This algorithm is simple and two ready queues are needed, one for periodic tasks, in wihch RM scheduling is used, and one for the aperiodic ones, in which FCFS scheduling is used. On the contrary, in case of high periodic workload, response time for aperiodic tasks can be very high and it is suitable only when aperiodic tasks are soft real-time.

Polling server

A special periodic tasks, the server, is created to serve aperiodic tasks as soon as possible. the server is characterized by:

- T_s: period of the server
- C_s: capacity/budget of the server

The server is scheduled using RM as other periodic tasks and it consumes its budget either running the aperiodic task or immediately if no aperiodic task is ready. Its budget is replenished at the beginning of each new period.

Considering n period tasks each with utilization U_i and a polling server with utilization $U_s = C_s/T_s$, the task set s feasible with RM if

$$\prod_{i=1}^{n}(U_i + 1) \leq \frac{2}{U_s + 1}$$

In order to set C_s and T_s so that the resulting scheduling is feasible, we look for the polling server maximum utilization factor

$$P \stackrel{\text{def}}{=} \prod_{i=1}^{n}(U_i + 1)$$

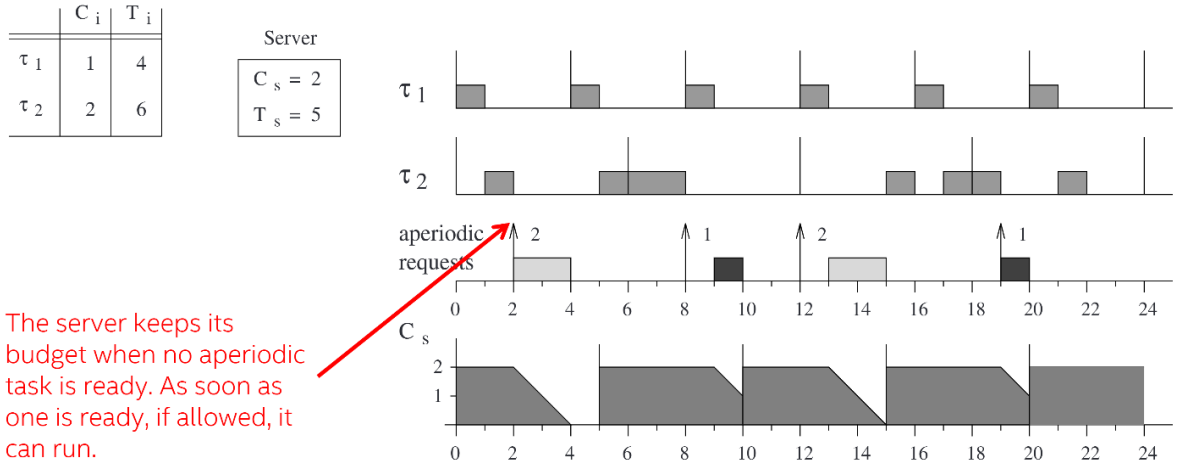
$$U_s^{max} = \frac{2 - P}{P}$$

Given U_s^{max} the rule of thumb is the following:

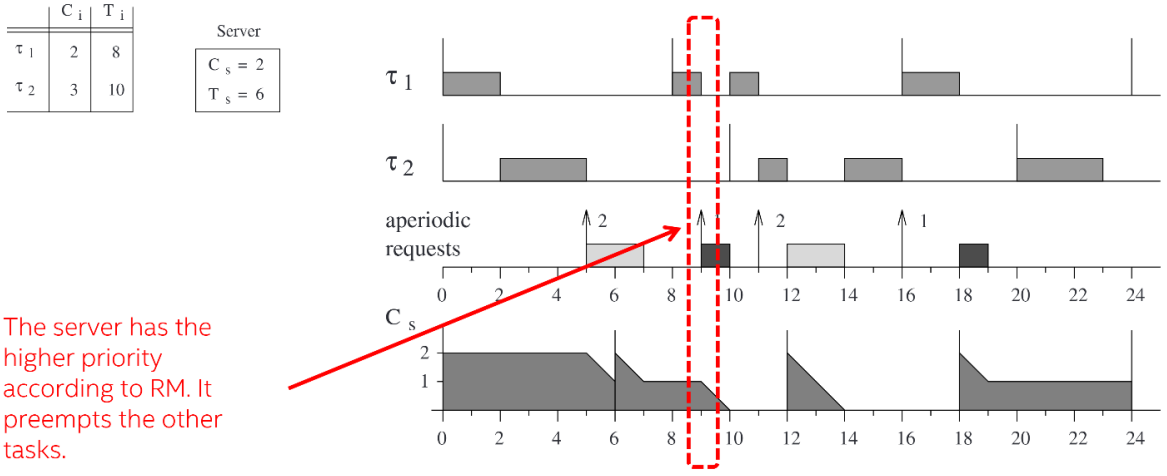
- set U_s at most equal to U^{max}
- set T_s as the period of the periodic task with the shortest period
- set $C_s = U_s T_s$

Deferrable server

As the polling server, the DS algorithm creates a periodic task for servicing aperiodic tasks and usually it has a high priority. DS preserves its capacity if no requests are pending and the capacity is maintained until the end of the period. Aperiodic requests can be serviced at the same server's priority at anytime, as long as the capacity has not been exhausted. At the beginning of any server period, the capacity is replenished at its full value.



The server keeps its budget when no aperiodic task is ready. As soon as one is ready, if allowed, it can run.



The server has the higher priority according to RM. It preempts the other tasks.

Considering n period tasks each with utilization U_i and a polling server with utilization $U_s = C_s/T_s$, the task set is feasible with RM if

$$\prod_{i=1}^{n}(U_i + 1) \leq \frac{U_s + 2}{2U_s + 1}$$

To set C_s and T_s so that the resulting scheduling is feasible, we have to look for the polling server maximum utilization factor U_s^{max}.

$$P \stackrel{\text{def}}{=} \prod_{i=1}^{n} (U_i + 1)$$

$$U_s^{max} = \frac{2 - P}{2P - 1}$$