Overview of Scheduling a Mix of Periodic and Aperiodic Tasks

Minsoo Ryu

RTCC Lab.
College of Information and Communications

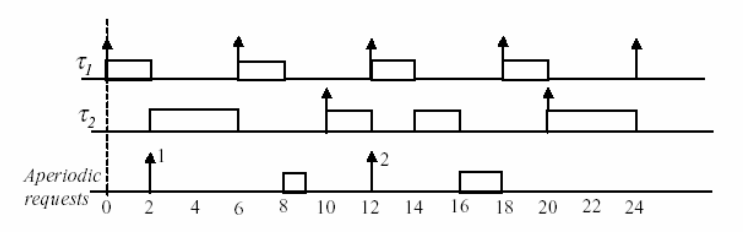
Algorithms

- Naive approach
  - Background scheduling
- Under static priority policy (RM)
  - Polling Server
  - Deferrable Server
  - Priority Exchange
  - Sporadic Server
  - Slack Stealing
- Under dynamic priority policy (EDF)
  - Dynamic Priority Exchange
  - Total Bandwidth Server

Background Scheduling

- Aperiodic tasks are executed when there is no periodic task to execute

task$_1$ (6, 2) and task$_2$ (10, 4)



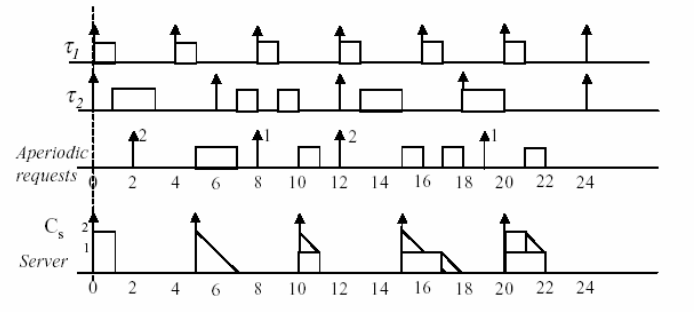
(period, execution time)

Polling Server

- A periodic task is created (period = T_0 , capacity = C_0)
  - It is assigned a priority according to the rate monotonic algorithm
  - It is subject to preemption by higher priority tasks
- Scheduling of PS (polling server)
  - PS is ready and its capacity is replenished at the start of its period
  - PS services aperiodic task arrivals until either it exhausts its execution time C_0 or there is no aperiodic work left to be executed
  - In the latter case, PS loses any of unused execution time and is unable to service aperiodic tasks until the start of its next period

Polling Server Example

Server (5, 2), task$_1$ = (4, 1), and task$_2$ = (6, 2)

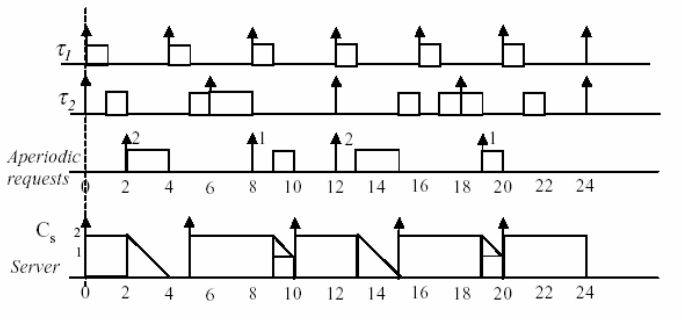


Deferrable Server

- A periodic task is created (period = T_0 , capacity = C_0)
  - In general, it is assigned a priority according to the rate monotonic algorithm
  - In practice, we often consider it at the highest priority level
- Scheduling of DS (deferrable server)
  - DS is ready and its capacity is replenished at the start of its period
  - DS services aperiodic task arrivals until it exhausts its execution time C_0 , or until the end of its period is reached
  - Unlike PS, DS does not lose its unused execution time and is able to service aperiodic tasks throughout its entire period
  - DS loses any of its unused execution time at the end of its period

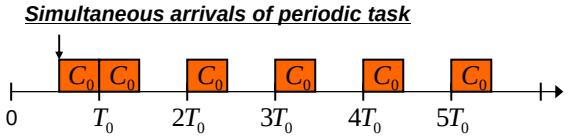
Deferrable Server Example

Server (5, 2), task$_1$ = (4, 1), and task$_2$ = (6, 2)



Deferrable Server

- Schedulability condition for periodic tasks with DS
  - The DS task violates one of Liu and Layland assumptions, namely that the task is ready at the start of the task period
  - The DS task can defer its execution time until later in its period
- Critical instant (worst case phasing)



Deferrable Server

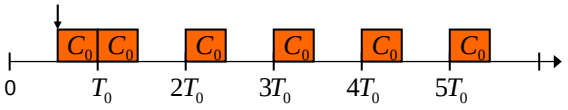
- Time demand analysis for periodic tasks only

$$\forall_i, \exists_t : 1 \le i \le n : \frac{C_i^t}{t} + \frac{I_i^t}{t} \le 1 \qquad I_i^t = \sum_{j=1}^{i-1} \left\lceil \frac{t}{T_j} \right\rceil \cdot C_j$$

- Total time demand with DS

$$C_0 + C_0 \cdot \left\lceil \frac{t - T_0}{T_0} \right\rceil + \sum_{j=1}^{i} \left\lceil \frac{t}{T_j} \right\rceil \cdot C_j$$

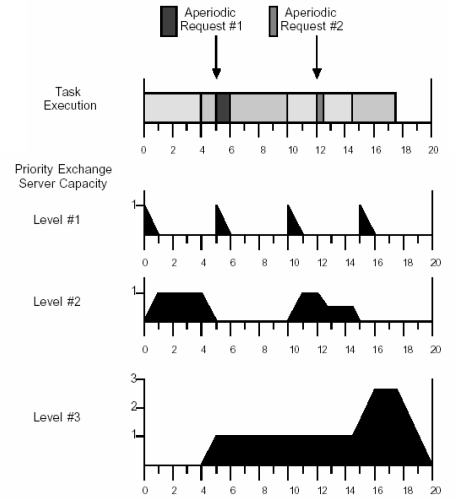
Simultaneous arrivals of periodic task



Priority Exchange

- A periodic server is created (period = T, capacity = C)
- Scheduling
  - If there are no aperiodic tasks, the server exchanges its priority with a lower priority periodic task allowing the periodic task to run
  - The server's capacity is not lost, but preserved at a low priority
- Replenishment
  - The computation time allowance for the server is replenished at the start of its period

Priority Exchange Example

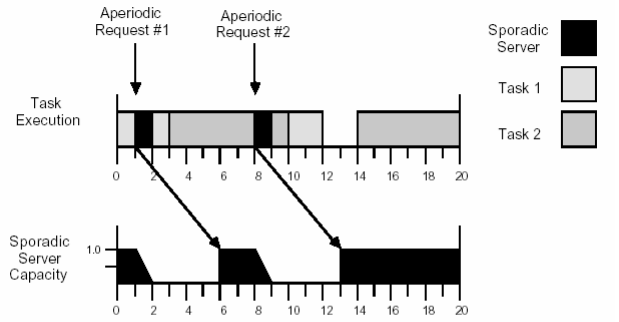


Server (5, 1)
$task_1$ = (10, 4)
$task_2$ = (20, 8)

Sporadic Server

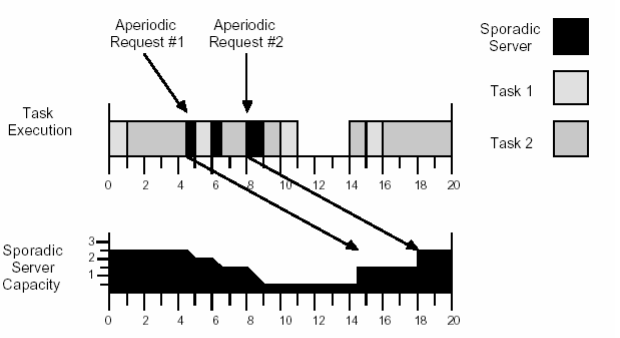
- Sporadic server retains the advantages of DS and PE algorithms
- Scheduling
  - The server preserves its server execution time at its priority level until an aperiodic request occurs
  - If there are enough aperiodic tasks in an invocation, it serves up to the server's capacity
- Replenishment
  - The computation time allowance for the server is replenished after some or all of the execution time is consumed by aperiodic task execution
- Sporadic server can be treated as a standard periodic task

Sporadic Server Example



Task	Exec Time	Period	Utilization
SS	1	5	20.0%
τ_1	2	10	20.0%
τ_2	6	14	42.9%

Sporadic Server Example

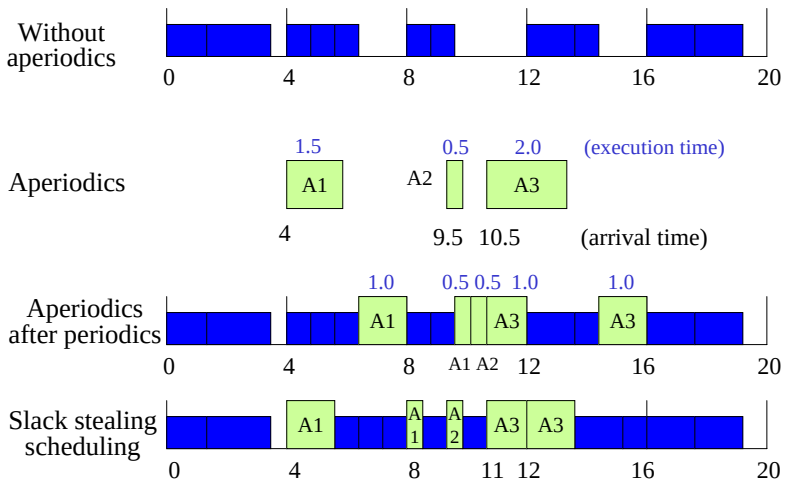


Task	Exec Time	Period	Utilization
τ_1	1.0	5	20.0%
SS	2.5	10	25.0%
τ_2	6.0	14	42.9%

Slack Stealing

- **Slack stealing**
  - **No periodic server**
  - **Steals time from periodic tasks without causing any deadline misses**
  - **Slack = deadline – current time – execution time**
- **Slack stealer**
  - **Relies on the exact schedulability conditions for RM and DM**
    - Find the slack available by mapping out the processor schedule for the hard periodic tasks over their hyperperiod
    - The slack values are stored in a table (off-line computation)
  - **Is optimal in the sense that it minimizes the response time of soft aperiodic tasks**

Slack Stealing Example



Total Bandwidth Server

- Used under EDF policy
- Algorithm
  - When the kth aperiodic request arrives at time $t = r_k$
  - It receives a deadline
    - $d_k = \max(r_k, d_{k-1}) + C_k/U_s$
    - C_k is the execution time of the request
    - U_s is the server's utilization factor
  - The request is scheduled as any other standard periodic request
- Schedulability condition
  - $U_p + U_s <= 1$
  - U_p : utilization of periodic tasks

Total Bandwidth Server Example

- Example: $U_p = 0.75,\ U_s = 0.25,\ U_p + U_s = 1$