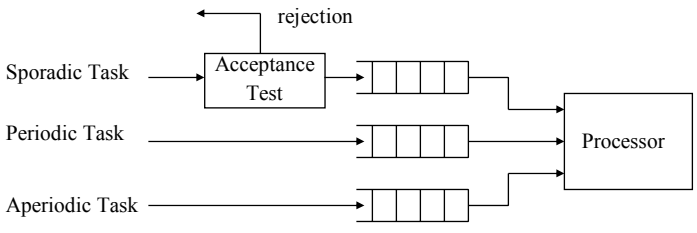
Scheduling of Aperiodic and Sporadic Jobs in Priority-Driven Systems
- Chapter 7 –

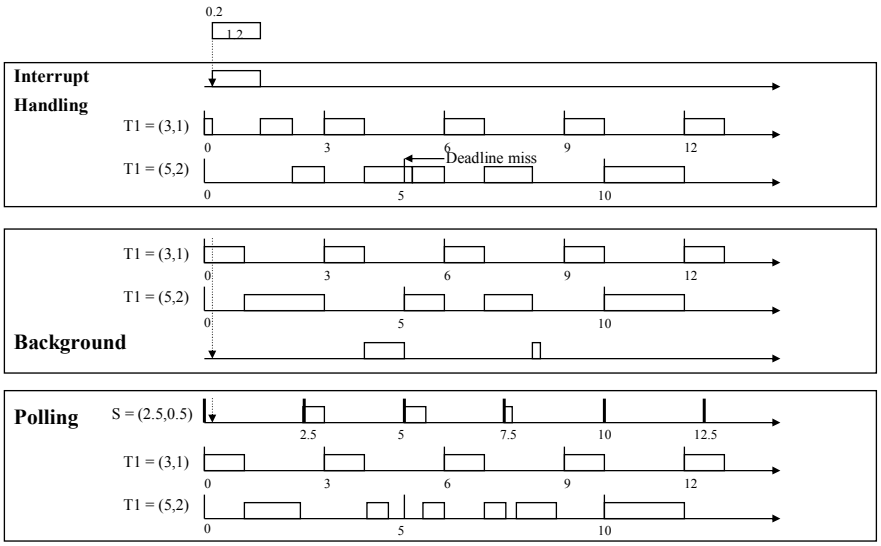
(Fixed-Priority Framework)

Types of Aperiodic Requests

- The jobs of an aperiodic task have random release times
  - Soft aperiodic tasks:
    - random arrivals such as a Poisson distribution:
    - the execution time can also be random such as exponential distribution
    - typically it models users' requests.
  - Firm aperiodic tasks (Sporadic tasks):
    - there is a minimal separation between 2 consecutive arrivals
    - there is a worst-case execution time bound
    - models emergency requests such as the warning of engine overheat



Interrupt Handling, Background, Polling



Interrupt Handling or Background Service

- One way to serve aperiodic requests is handle them right at the interrupt handler.
  - This gives the best response time but can greatly interrupt the hard real-time periodic tasks.
  - Use it as last resort only such as pending power failure exception handling
- Another simple method is to give background class priority to aperiodic requests. This works as well but the response time is not too good. For example:
  - Priority levels 1 to 246 for periodic tasks
  - Priority levels 247 to 256 for aperiodic tasks

Polling - 1

- The simplest form of integrated aperiodic and periodic service is polling.
  - For each aperiodic task, we assign a periodic service with budget e_s and period p_s. This creates a server (e_s, p_s)
  - The aperiodic requests are buffered into a queue
  - When polling server starts,
    - Resumes the existing job if it was suspended in last cycle.
    - it checks the queue.
  - The polling server runs until
    - All the requests are served
    - Or suspends itself when the budget is exhausted.
  - Remark: a small improvement is to run the tasks in background priority instead of suspend. This background mode can be applied to all the servers discussed later.

Polling - 2

- A polling server is just a periodic task and thus the schedulability of periodic tasks is easy to analyze. For example, if we use L&L bound,

$$\sum_{i=1}^{n} \frac{e_i}{p_i} + \frac{e_s}{p_s} \le (n+1)\left(2^{1/(n+1)} - 1\right)$$

- Quiz: How can we analyze the aperiodic performance for each polling server?
- Answer: Using M/M/1 for 1st cut analysis. Server size is e_s/p_s. Service time is ($0.5p_s$ + average request service time)…

Average response time In M/M/1 queue

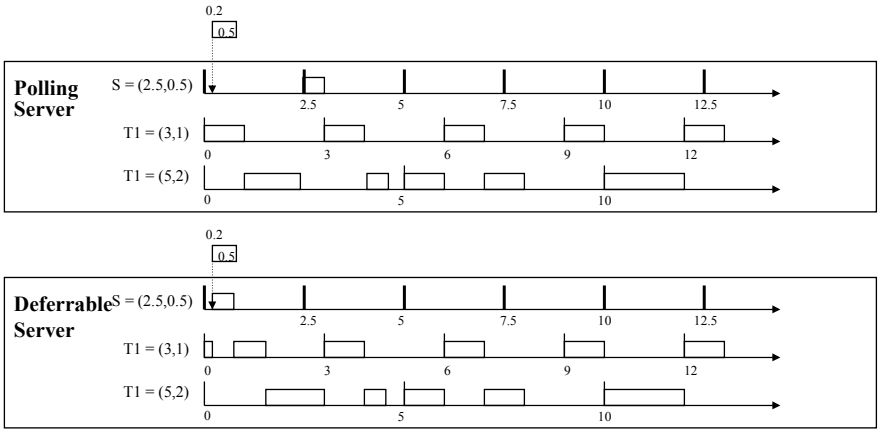
$$w = \frac{E[S]}{1-\rho}$$

where E[S] is the average service time and ρ is the average utilization of the bandwidth assigned to aperiodic jobs

Deferrable Server - 1

- Comparing polling with interrupt handling, interrupt handling serves aperiodic requests right away whereas the Polling Server creates an average of half a period waiting time.
- Deferrable Server is the 1st attempt to simulate interrupt handling service but bounds the service time of aperiodics so that it ensures periodic tasks are schedulable.
- The idea is to let the budget float, just like getting a monthly salary. The salary allocation is periodical, but one can spend it anytime he likes.

Deferrable Server - 2



Deferrable Server - 3

- Example: e = 50 ms; p = 250 ms.

- Every 250ms, the budget is RESET to 50 ms (no savings of unused budget!)
- Aperiodic requests arrive at a queue.
- The head of queue request checks if there is budget available.

- If there is budget left,
- the aperiodic request runs until either the request is served
- or the budget is exhausted
- and therefore the aperiodic request is suspended until there is new budget available
- else the aperiodic request is suspended and it waits until there is new budget available

Deferrable Server - 4

- When the budget >> requests workload, requests seldom suspend. It has interrupt like service if the deferrable server is running at a high priority.

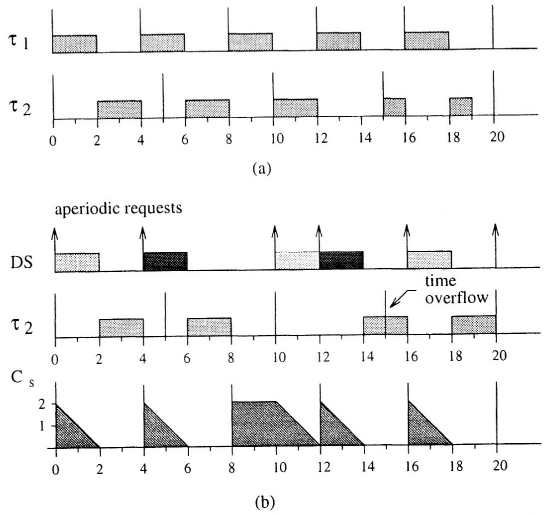
- We can model it with M/M/1 with service time = 0*p + average request service time.

- When the budget << requests workload, it behaves just like polling.

- In other cases, you need to do simulation to determine the performance.

Deferrable Server vs. periodic task

- A Deferrable Server is not equivalent to a periodic task!



Deferrable Server - 5

- Schedulability of periodic tasks using RMS. Let the period of the sever be p. For any lower priority task with period p_i, it generates at most ceiling (p_i/p) times preemption, if it was a regular periodic task. However, it can generate (1 + ceiling(p_i/p)) times the preemption.

• Note that task 1 originally starts at t = 0 and the interval for the preemption is [0, 10]. In the second example, a 1 unit shifting lets the deferred unit to come in. The starting time is now 1 and the interval for preemption is still [0, 10]

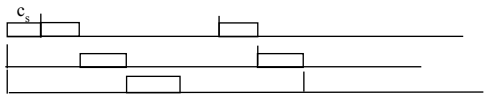
Deferrable Server - 6

- Scheduling bound under RMS: Considering the 1 additional unit of preemption, we will get the following bound

$$U_{\text{lub}} = n\left[\left(\frac{U_s + 2}{2U_s + 1}\right)^{1/n} - 1\right]$$

where U_s is the utilization of the Deferrable Server (e/p). – see textbook exercise 7.7

- It is worth noting that the tasks' pattern that provides the worst-case condition for the periodic tasks under the RM algorithm is:



Deferrable Server - 7

- Given a set of n periodic tasks and a Deferrable Server with utilization factors U_p and U_s, respectively, the schedulability of the periodic task set is guaranteed under RM if:

$$U_p \le U_{\text{lub}} = n\left[\left(\frac{U_s + 2}{2U_s + 1}\right)^{1/n} - 1\right]$$

where U_s is the utilization of the Deferrable Server (e/p).

Deferrable Server - 8

- Time demand analysis: since there *could* be an additional preemption, a sufficient condition is to use the old time demand analysis and add 1 to the preemption of the deferrable task's term.

$$a_i(t) = e_i + (1 + \left\lceil \frac{t}{p} \right\rceil) e + \sum_{j=1}^{i-1} \left\lceil \frac{t}{p_j} \right\rceil e_j$$

An improvement can be made by noting that we can substract e out of t in the ceiling function for the deferrable server, since we shift the starting time to the right by e units to let the deferred units to come in.

$$a_i(t) = e_i + (1 + \left\lceil \frac{t-e}{p} \right\rceil) e + \sum_{j=1}^{i-1} \left\lceil \frac{t}{p_j} \right\rceil e_j$$

if we have m servers have shorter periods than task *i*

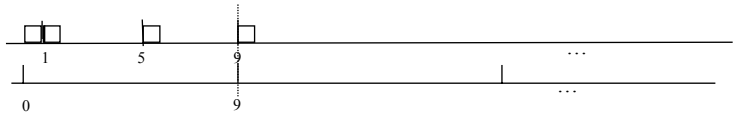
$$a_i(t) = e_i + \sum_{s=1}^{m} (1 + \left\lceil \frac{t-e_s}{p_s} \right\rceil) e_s + \sum_{j=1}^{i-1} \left\lceil \frac{t}{p_j} \right\rceil e_j$$

Remark: the textbook has an addition b term for blocking. We assume it is 0 for now

Deferrable Server - 9

- The effect of shifting to the right can best be illustrated as follows.

- 1 + ceiling(9/4) = 4; over count 1 unit
- 1 + ceiling ((9-1)/4) = 3; exact.



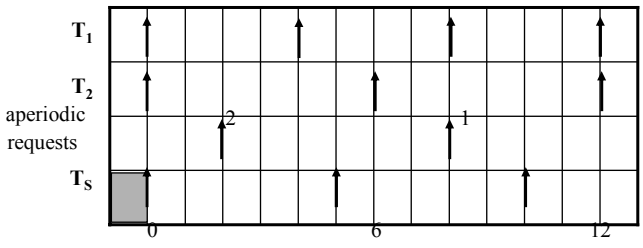
Class exercise (1)

- Consider the following task set
  - T_1 {e_1=1, p_1= 4}
  - T_2 {e_2=2, p_2= 6}
  - T_s {e_s=1, p_s= 5}
- Are the periodic task set and the deferrable server T_s schedulable?

Use the time demand analysis ➔ $$a_i(t) = e_i + (1 + \left\lceil \frac{t-e}{p} \right\rceil)e + \sum_{j=1}^{i-1} \left\lceil \frac{t}{p_j} \right\rceil \cdot e_j$$

Class exercise (2)

- Consider the following task set
  - T_1 {e_1=1, p_1= 4}
  - T_2 {e_2=2, p_2= 6}
  - T_s {e_s=1, p_s= 5}
- Schedule the following aperiodic activities by using the deferrable server



Sporadic Server - 1

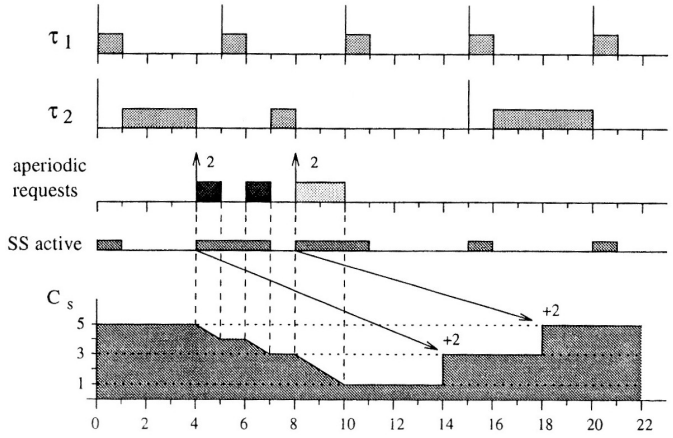
- The deferrable server has this one additional preemption and reduces the schedulability of periodic tasks. So we tried to get rid of this additional preemption.
- The SS differs from DS in the way it replenishes its capacity. Whereas DS periodically replenishes its capacity at the beginning of each server period, SS replenishes its capacity only after it has been consumed by aperiodic task execution.
- Idea: Spread the budget replenishment at least P time units
- We will see that Sporadic Server can be treated as if it is a periodic task.

Sporadic Server - 2

- A Sporadic Server with priority $Prio_s$ is said to be *active* when it is executing or another task with priority $Prio_T \geq Prio_s$ is executing. Hence, the server remains active even when it is preempted by a higher priority task.
- If the server is not active, it is said to be *idle*
- **Replenishment Time (RT):** it is set as soon as "SS becomes active and the server capacity $C_s > 0$". Let T_A be such a time. The value of RT is set equal to T_A plus the server period ($RT = T_A + p_s$).
- **Replenishment Amount (RA):** The RA to be done at time RT is computed when "SS becomes idle or the server capacity C_s has been exhausted". Let T_I be such a time. The value of RA is set equal to the capacity consumed within the interval $[T_A, T_I]$.

Sporadic Server - 3

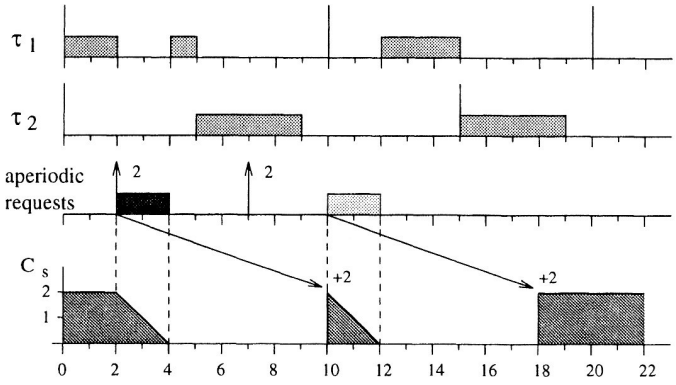
- Example of a medium-priority Sporadic Server.



	e	p
T_1	1	5
T_S	5	10
T_2	4	15

Sporadic Server - 4

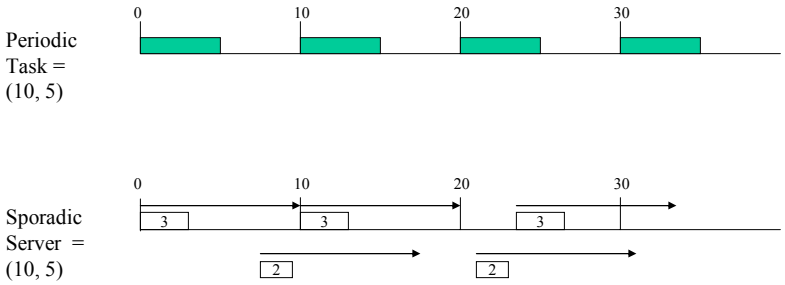
- Example of a high-priority Sporadic Server.



	e	p
T_S	2	8
T_1	3	10
T_2	4	15

Sporadic Server vs. periodic task

- A Sporadic Server ≤ a periodic task!



Sporadic Server - 5

- *A periodic task set that is schedulable with a task T_i is also schedulable if T_i is replaced by a Sporadic Server with the same period and execution time.*

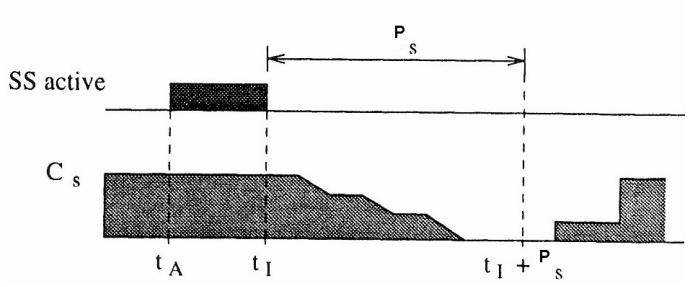
Proof:

Let's prove that SS exhibits in the worst case an execution behavior equivalent to one or more periodic tasks with period p_s and total execution time equal to e_s. Consider the execution behavior of the server during the interval $[T_A, T_I]$, we can analyze the following three cases:

1) No capacity is consumed
2) The server capacity is totally consumed
3) The server capacity is partially consumed

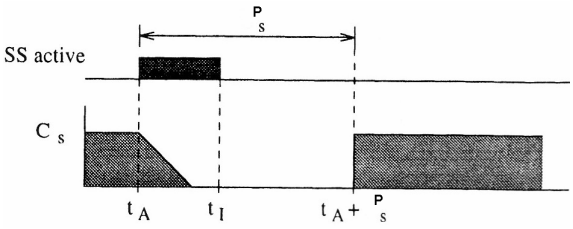
Sporadic Server - 6

Case 1: If no requests arrive in $[T_A, T_I]$, SS preserves its capacity and no replenishments can be performed before time $T_I + p_S$. This means that at most e_S units of aperiodic time can be executed in $[T_I, T_I + p_S]$. Hence, the SS behavior is identical to a periodic task $T_s(e_s, p_s)$ whose release time is delayed from T_A to T_I.



Sporadic Server - 6

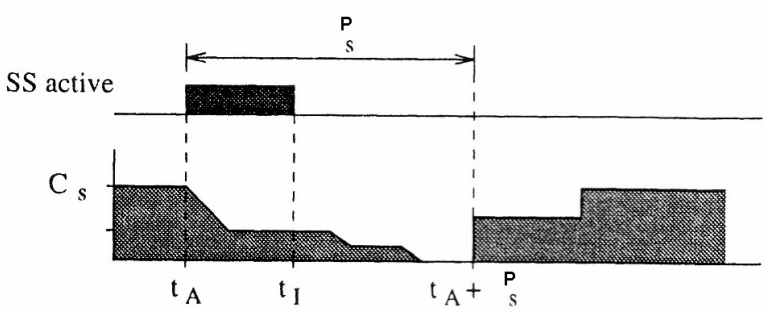
Case 2: If e_s is totally consumed in $[T_A, T_I]$, a replenishment of e_s units of time will occur at time $T_A + p_s$. Hence, the SS behavior is identical to a periodic task $T_s(e_s, p_s)$ released at time T_A.



Case 3: If e_s is partially consumed in $[T_A, T_I]$, a replenishment will occur at time $T_A + p_s$, and the remaining capacity is preserved for future requests. This case is just the combination of the previous two cases. As a consequence, the behavior of the server is equivalent to two periodic tasks T_x and T_y, released at T_A and T_I respectively. Both tasks have the same period p_s with executions times e_x and e_y such that $e_s = e_x + e_y$.

Sporadic Server - 7

Since in any servicing situation SS can be represented by one or more periodic tasks with period p_s and total execution time e_s, the contribution of SS in terms of processor utilization is equal to $U_s=e_s/p_s$. Hence, from a scheduling point of view, SS can be replaced by a periodic task having the same utilization factor. □



Class exercise (3)

- Consider the following task set
  - T_1 {e_1=1, p_1= 4}
  - T_2 {e_2=2, p_2= 7}
  - T_s {e_s=?, p_s= 5}
- What is the maximum possible e_s if it is deferrable server?
- What is the maximum possible e_s if it is sporadic server?