

Deadlines and Aperiodic Servers

Raj Rajkumar
Lecture #8

Outline

- Dealing with Deadline \neq Period
 - Deadline-monotonic scheduling
 - Deadlines $>$ period
- Dealing with Interrupts
- Aperiodic Tasks
 - Deferrable Servers
 - Sporadic Servers

Modeling Pre-period Deadlines

- Suppose a task τ , with a worst-case computation time of C and a period of T , has a “pre-period” deadline D (i.e. $D < T$).
- Compare total utilization to modified bound:

$$U_{total} = \frac{C_1}{T_1} + \dots + \frac{C_n}{T_n} \leq U(n, \Delta_i)$$

where Δ_i is the ratio D_i / T_i .

$$U(n, \Delta_i) = \begin{cases} n((2\Delta_i)^{1/n} - 1) + 1 - \Delta_i, & \frac{1}{2} < \Delta_i \leq 1.0 \\ \Delta_i, & \Delta_i \leq \frac{1}{2} \end{cases}$$

Deadline-Monotonic Scheduler

- Assign fixed priority based on D (and not T)
 - Shorter the relative deadline, higher the priority.
- Optimal fixed-priority preemptive scheduling policy for periodic tasksets where $D \leq T$
 - i.e. the relative deadline of each task is not greater than the task period
- When $D = T$, RMS and DMS are one and the same
 - When $\Delta_i = (D_i / T_i)$ is constant across all tasks, RMS and DMS are also the same.
- When $D > T$, neither RMS nor DMS is the optimal fixed-priority scheduler!

Notes on Fixed-Priority Scheduling Policies

- Both RMS and DMS are fixed-priority scheduling policies. Hence, the exact response-time test can be used to verify the schedulability of tasksets using RMS, DMS or any other fixed-priority policy.
 - Compute the worst-case completion time and check whether the completion time is not greater than the deadline.
 - Does NOT work for case when $D > T$.
 - One has to check for a longer duration.
- Hence, the general set of principles for analyzing fixed-priority preemptive scheduling policies is called **RMA (Rate-Monotonic Analysis)**.

Schedulability with Interrupts

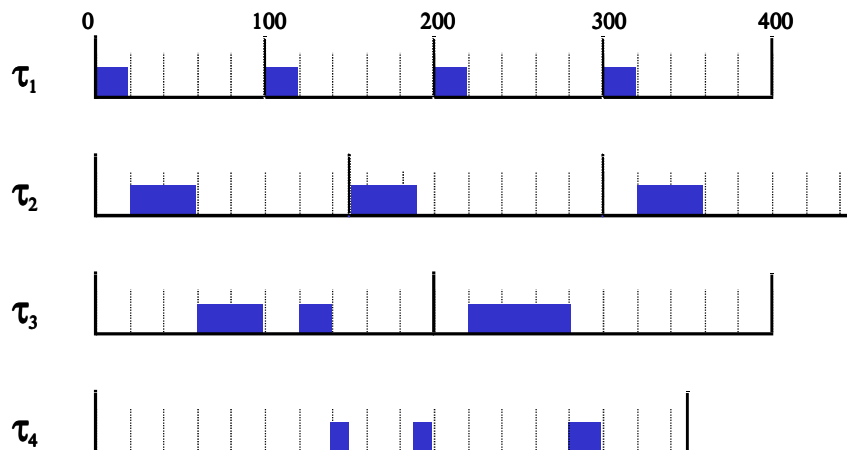
- Interrupt processing can be inconsistent with rate-monotonic priority assignment.
 - interrupt handler executes with high priority despite its longer period
 - interrupt processing may delay execution of tasks with shorter periods
- Effects of interrupt processing must be taken into account in schedulability model.
- Question is: how to do that?

Example: Determining Schedulability with Interrupts

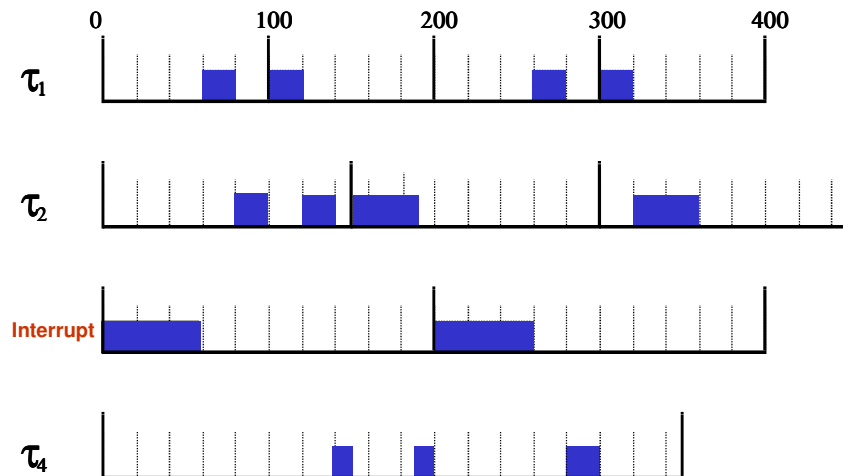
| Task | C | T | U |
|----------|-----|-----|-------|
| τ_1 | 20 | 100 | 0.200 |
| τ_2 | 40 | 150 | 0.267 |
| τ_3 | 60 | 200 | 0.300 |
| τ_4 | 40 | 350 | 0.115 |

τ_3 is an interrupt handler

Example: Execution with Rate-Monotonic Priorities



Example: Execution with an Interrupt Priority



Resulting Table for Example

| Task (i) | Period (T) | Execution Time (C) | Priority (P) | Deadline (D) |
|-------------|---------------|-----------------------|-----------------------|-----------------|
| τ_3 | 200 | 60 | Hardware (highest) | 200 |
| τ_1 | 100 | 20 | High | 100 |
| τ_2 | 150 | 40 | Medium | 150 |
| τ_4 | 350 | 40 | Low | 350 |

- For τ_1 , τ_3 introduces priority inversion $\rightarrow B_1 = 60$.
- For τ_2 , τ_3 introduces priority inversion $\rightarrow B_2 = 60$.
- For τ_3 , it must satisfy its own deadline ($60 < 200?$).
- For τ_4 , τ_3 looks like a normal higher priority task.

Concepts and Definitions

- Aperiodic task
 - runs at irregular intervals.
- Aperiodic deadline:
 - hard, minimum interarrival time
 - soft, best average response

Techniques

- Approaches to handle aperiodic tasks:
 - Background server:
 - Long response times
 - Tight guarantees are difficult
 - Slack stealing
 - Exploit the fact that jobs often do not execute up to their worst-case execution times
 - Periodic servers

Periodic Servers

- Polling Server
 - Simple
 - Commonly used
 - Executes at the “highest” priority
 - Provides a guaranteed fraction of the processor for dealing with aperiodic tasks
 - Worst-case response times can be long (why?)
- Deferrable Server
 - Improves on the response time of the polling server
 - Maintains its advantages
 - Cannot be generalized to multiple instances at different priority levels
- Sporadic Server
 - Improves upon the deferrable server and is very generalizable
 - Higher run-time complexity

Deferrable Server (DS)

- Create a high-priority “server” with an associated budget of C time-units and a period of T (i.e. $T_{DS} \leq$ shortest period of all the normal periodic tasks)
- When aperiodic tasks arrive, they check for any available server budget
 - If available, use the budget to execute at the highest priority
 - Budget is reduced correspondingly for every unit of server execution
 - If budget is depleted, aperiodic tasks can execute at background priority
- The budget of C gets replenished every T time-units
 - Unused budget does not get carried over
 - No “rollover” minutes

DS Example



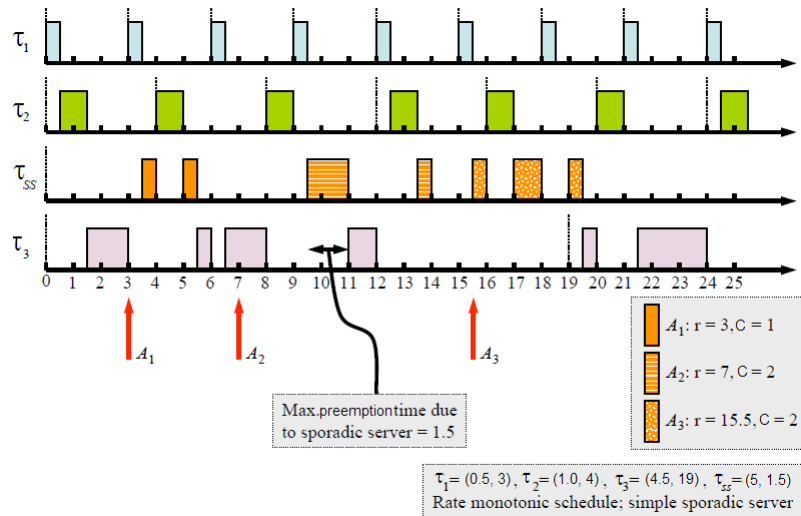
τ_1 preempted for 1.2 time-units even though C_{DS} is only 1.

- This is referred to as the “back-to-back execution” effect.
- Such an effect *cannot* happen with the RMS (Liu and Layland) model.

Sporadic Server (SS)

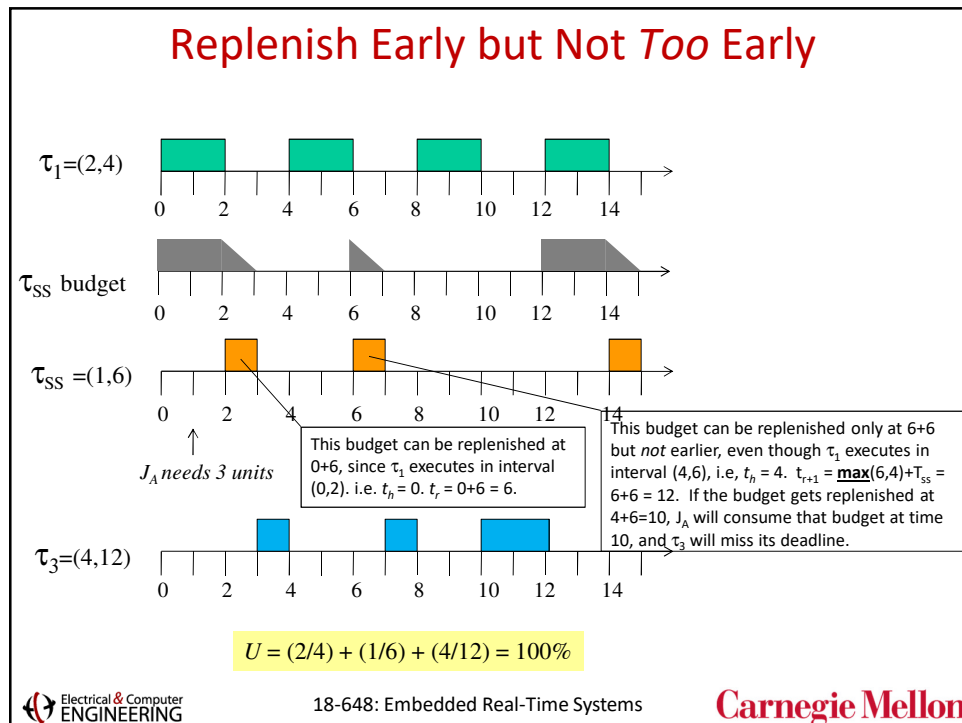
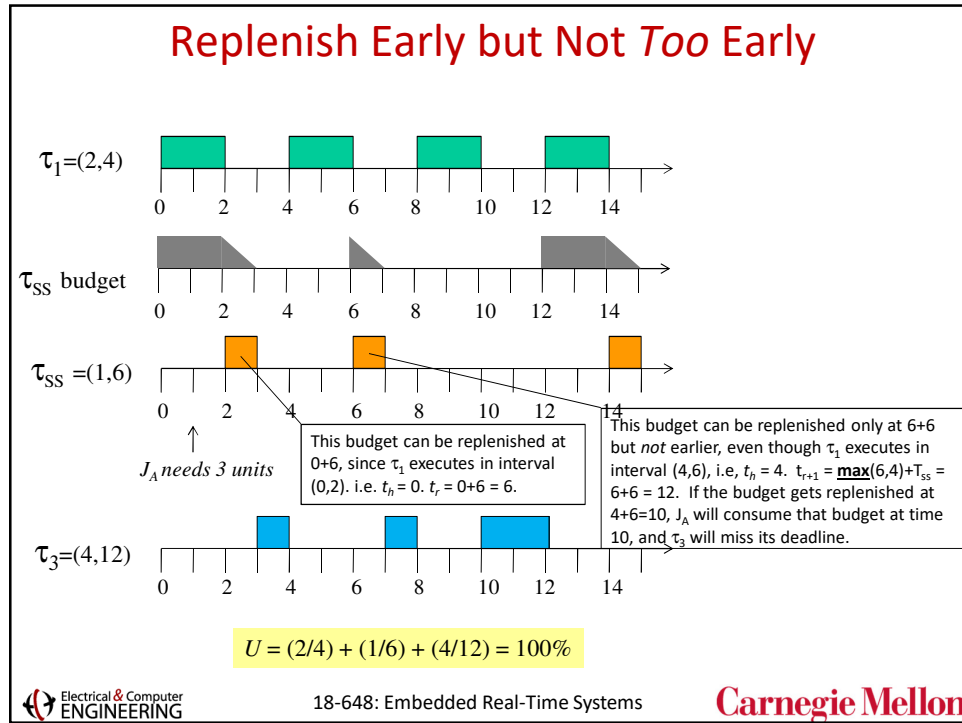
- A *sporadic server* is designed to eliminate the occurrence of “back-to-back” executions
 - More complex consumption and replenishment rules ensure that a sporadic server with period T_s and budget C_s never demands more processor time than a periodic task with the same parameters
- Server priority is based on T_s , just like periodic tasks.
- Budget replenishment occurs one “period” after **start** of use.
- Budget can be replenished earlier if sporadic server is preempted by a higher priority task
 - But not too early!
- Replenishment time $t_r = \max(t_{r-1}, t_h) + T_s$
 - where t_h is the start of a higher (or equal priority) task running just before the SS
- There can be multiple “pieces” of budget with different replenishment times (these budgets can be combined later)

SS Example



Another Optimization

- If the CPU ever becomes idle, any SS capacity can be immediately replenished (to its original allocated budget).



A Sample Taskset

Periodic Tasks

- $\tau_1 = (20, 100)$
- $\tau_2 = (40, 150)$
- $\tau_3 = (100, 350)$

Sporadic Task

- Emergency task: $C = 5$, $D = 6$, Minimum Inter-Arrival Time = 50

Create a sporadic server with $C=5$, $T=50$. Completion = $5 < D$

Aperiodic Task

User Input: $C = 2$, Minimum inter-arrival time = 40

Desired response time = 25 ms after arrival

Use simulation and queueing theory based on M/M/1 approximation:
response time ~20ms

Summary

- Dealing with Deadline \neq Period
 - Deadline-monotonic scheduling
 - Deadlines $>$ period
- Dealing with Interrupts
- Aperiodic Tasks
 - Deferrable Servers
 - Sporadic Servers