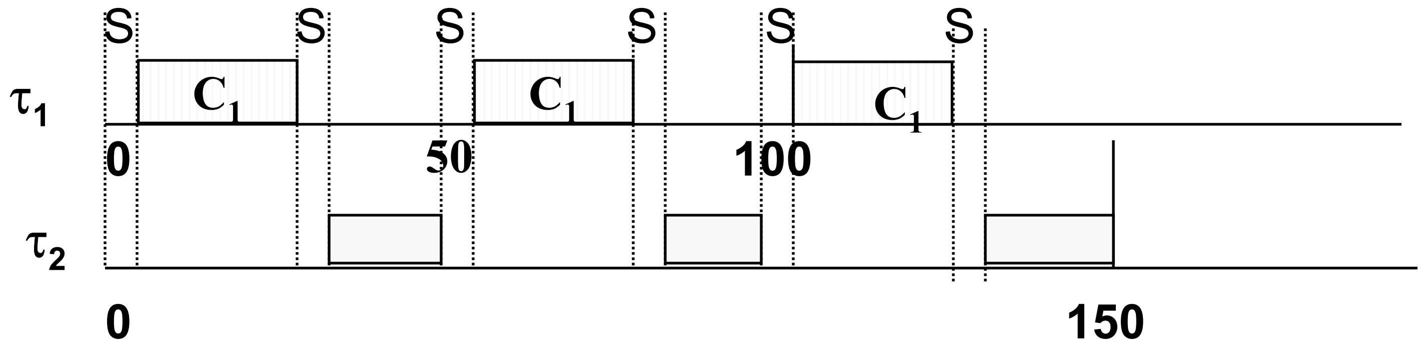
Overview

In this lecture, we will student more practical issues in the application of GRMS:

- Pre-period deadlines
- high priority I/O
- interrupts

Review: Context Switch Time

- An instance of a periodic task can cause at most two context switches. To simply the analysis, we shall assume that this is always the case.
- In the following example, the time left for τ_2 is $(150 - 3C_1 - 6S) = (150 - 3(C_1+2S))$. Thus we can take the effect of context switching into account by replacing C_1 with $(C_1 + 2S)$ and reuse analysis methods which assume that the context switching overhead is zero.

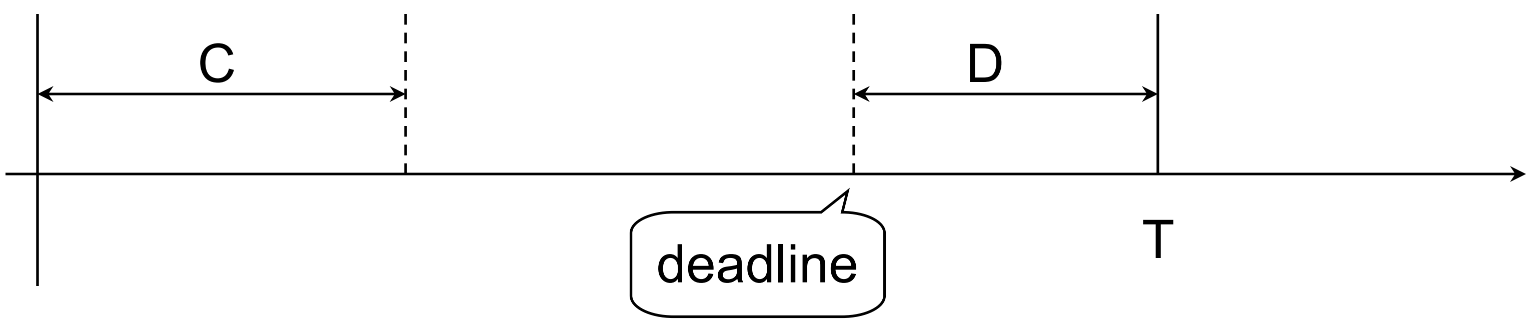


<u>The context overhead incurred by task τ_i is $2S/T_i$, while the net processor utilization by task τ_i is C_i/T_i.</u>

Modeling Preperiod Deadlines

Suppose task τ, with compute time C and period T, has a preperiod deadline D.

- In UB tests, pre-period deadline can be modeled as if the task has a longer execution time (C+D), because if the task has execution time (C+D) can finish before time T, then we know it must finish D units before T if it has only execution time C.
- In exact schedulability test, just move the deadline from T to (T - D)



Quiz

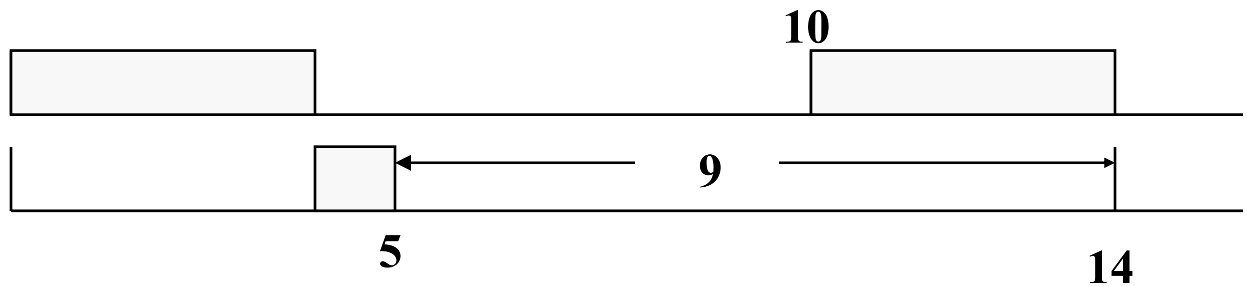
- We have modeled the effect of pre-period deadline by adding D to execution time C.

$$\frac{C_i + D_i}{T_i}$$

- It was suggested that we may also subtract D from the period T. What do you think? (Hint: are the 2 methods equivalent? Is there a counter example?)

$$\frac{C_i}{T_i - D_i}$$

Counter Example for (T – D)



- Consider the case of two tasks, {(C1= 4, T1= 10), (1, 14)}. If the pre-period is more than 9, then Task 2 will not be schedulable.
- However, 4/10 + 1/(14 – 10) = 0.4 + 0.25 = 0.625 < U(2). That is, it tells us that even if the pre-period deadline is at t = 4, it is still schedulable. This is obviously wrong

Task Switching and Pre-period Deadline

Suppose that task 2 has D_2 unit of preperiod deadline, we just add D_2 to task 2's execution time ***LOCALLY*** (Why?)

$$\tau_1 \qquad \frac{(C_1 + 2S)}{T_1} \le U(1)$$

$$\tau_2 \qquad \frac{(C_1 + 2S)}{T_1} + \frac{(C_2 + 2S + D_2)}{T_2} \le U(2)$$

$$\tau_3 \qquad \frac{(C_1 + 2S)}{T_1} + \frac{(C_2 + 2S)}{T_2} + \frac{(C_3 + 2S)}{T_3} \le U(3)$$

Example: Schedulability with Task Switching and Pre-period Deadline

Given the following tasks:

	C	T	D
Task $\tau 1$	1	4	
Task $\tau 2$	2	6	1
Task $\tau 3$	2	10	

Assume S = 0.05, are these 3 tasks schedulable?

Solution: Schedulability with Task Switching and Pre-period Deadline

$$\tau_1 \qquad \frac{(1 + 2(0.05))}{4} = 0.275 \leq U(1) = 1.00$$

$$\tau_2 \qquad \frac{(1 + 2(0.05))}{4} + \frac{(2 + 2(0.05) + 1)}{6} = 0.791 \leq U(2) = 0.828$$

$$\tau_3 \qquad 0.275 + 0.35 + \frac{(2 + 2(0.05))}{10} = 0.835 > U(3) = 0.779$$

a0 = 1.1 + 2.1 + 2.1 = 5.3
a1 = 2.1 + ceil(5.3/4) * 1.1 + ceil(5.3/6) * 2.1 = 6.4
a2 = 2.1 + ceil(6.4/4) * 1.1 + ceil(6.4/6) * 2.1 = 8.5
a3 = 2.1 + ceil(8.5/4) * 1.1 + ceil(8.5/6) * 2.1 = 9.6
a4 = 2.1 + ceil(9.6/4) * 1.1 + ceil(9.6/6) * 2.1 = 9.6
Task 3 is schedulable, it completes by 10.00

The Concept of Blocking in GRMS

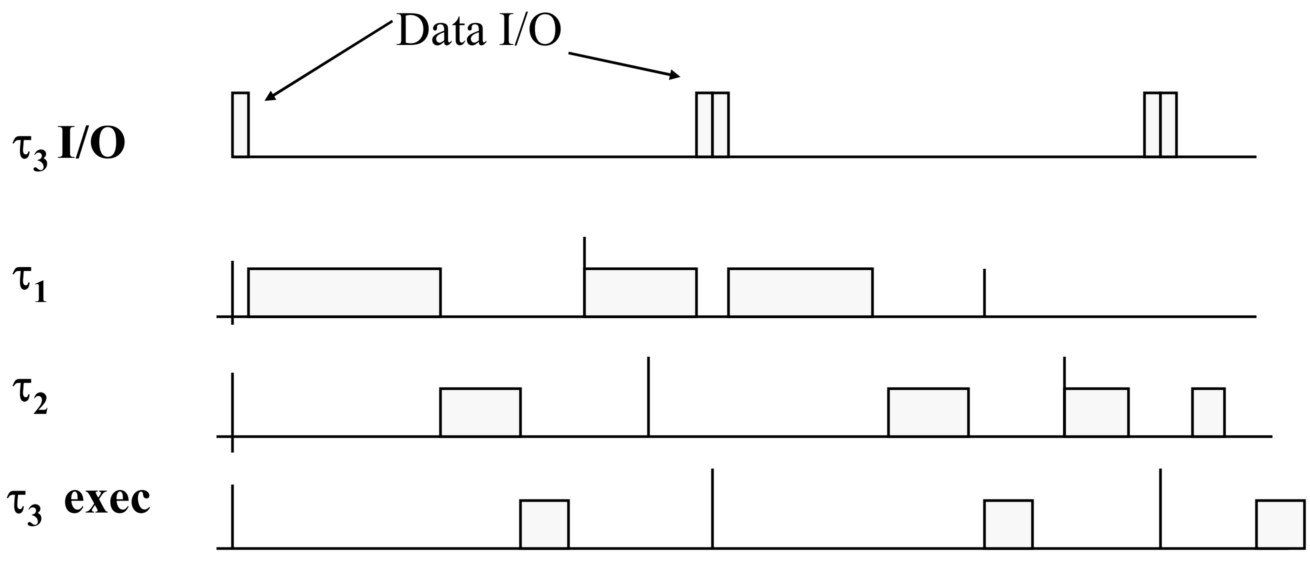
- In GRMS, short period tasks are given higher priorities. When a long period task is delayed by the execution of short period tasks, the long period task is said to be PREEMPTED by short period tasks.

- What if, for some reason, the long period task delays the execution of short period tasks? In this case, the short period task is said to be BLOCKED by long period tasks.

- (NOTE: in OS literature, if a higher priority delays a lower priority task, it is called preemption, independent of periods.)

Blocking Due to I/O and Interrupt Handling

In this example, τ_3 has the longest period. However its data I/O executes at top priority to reduce jitter. As a result, τ_3's data I/O blocks the execution of shorter period tasks , τ_1 and τ_2.



•Similarly, if we use interrupts to perform the data I/O, the ISR for data I/O will have higher priority even if it is done for a longer period task

Task Switching and Pre-period Deadline and Blocking

Suppose that a task has D unit of preperiod deadline and blocking time B, we just add them to its execution time for each task in UB test. In exact schedulability test, we will move the deadline from T to (T - D - B)

$$\tau_1 \qquad \frac{(C_1 + 2S + B_1 + D_1)}{T_1} \leq U(1)$$

$$\tau_2 \qquad \frac{(C_1 + 2S)}{T_1} + \frac{(C_2 + 2S + D_2 + B_2)}{T_2} \leq U(2)$$

$$\tau_3 \qquad \frac{(C_1 + 2S)}{T_1} + \frac{(C_2 + 2S)}{T_2} + \frac{(C_3 + 2S + D_3)}{T_3} \leq U(3)$$

Note that B_3 is always zero. It is the task with the longest period and therefore it cannot be blocked by a task with longer period.

Example: Interrupt and I/O

	C	T	B	D
Task $\tau 1$	1	4	?	
Task $\tau 2$	2	6	?	1
Task $\tau 3$	4	13	?	

Suppose that S=0.0. However, task 3 has two parts. part 1 is execution time C_3 = 1 and part 2 is high priority I/O that is 3 units long.

Fill in the blocking times in the table and determine if all three tasks are schedulable?

Example: I/O and Interrupts

	C	T	B	D
Task τ1	1	4	3	
Task τ2	2	6	3	1
Task τ3	4	13	0	

τ_1 $$\frac{(1+3)}{4} = 1.00 = U(1) = 1.00$$

τ_2 $\mathbf{a_0}$ **=** $1 + 2 = 3 > (6 - 1 - 3) = 2$. Not schedulable

τ_3 $\mathbf{a_0}$ **=** $1+2+4 = 7$,

$\mathbf{a_1}$ = ceil(7/4)*1 + ceil(7/6)*2 + 4 = 10

$\mathbf{a_2}$ = ceil(10/4)*1 + ceil(10/6)*2 + 4 = 11

$\mathbf{a_3}$ = ceil(11/4)*1 + ceil(11/6)*2 + 4 = 11<13

(The lowest priority one is ok!!)

Summary

- In this lecture, we learned
  - how to model pre-period deadline
  - the notion of blocking caused by giving longer period task I/O higher priorities
- We will study real time task synchronization next.