Towards and Exact analysis

From the previous lecture we found that :

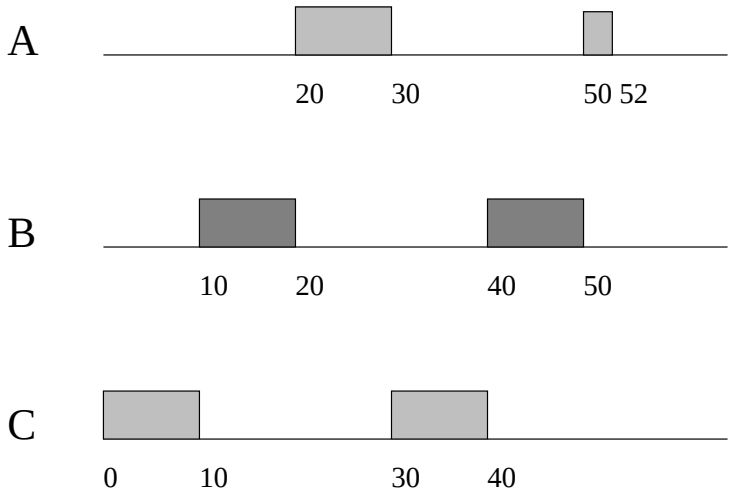
Slide 1

Name	T	C	Priority	Utilization
A	52	12	1	0.24
B	40	10	2	0.25
C	30	10	3	0.33

Total utilization = 0.81. But since 0.81 is more than 0.78 Liu and Layland's test does not say if it is possible to schedule the task set.

But we can.

Slide 2



An exact analysis

Slide 3

- In this lecture we are going to look at priority based scheduling and try and give a more exact analysis.
- It will turn out that it is harder to calculate than the Liu and Layland test, but we will be able to schedule more things.

Most Difficult invocation

Slide 4

If we can satisfy the deadlines for the most difficult invocation of the tasks, then the tasks will always meet their deadline. Where "most difficult" refers to the situation where all tasks are released simultaneously.

In '86 Joseph and Pandya[a] 1986 formalized the most difficult invocation into an easy to apply test.

[a] M. Joesph and P.Pandya, "The Computer Journal(British Computer Society)"

Exact Analysis

We need some more notation:

Slide 5

Notation	Description
C_i	Worst-case computation time
T_i	Period
D_i	Deadline
R_i	Worst-case response time

If we can calculate the worst-case response time of a task and show that it is earlier than the deadline then we can decide scheduling.

Worst-Case Response Time

How do we calculate the worst-case response time?

Slide 6

Remember we are using fixed-priority scheduling. So tasks with shorted periods get given higher priority. So the response time of a task should depend on the times and computations of all the higher priority tasks.

Given a task i define $\mathrm{hp}(i)$ to be the set of all tasks with a higher priority than task i.

Worst-Case Response Time

Joseph and Pandya derive the following equation:

Slide 7

$$R_i = C_i + \sum_{\forall j \in \mathrm{hp}(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

where $\lceil x \rceil$ is the smallest integer greater than x. For example $\lceil 3.1 \rceil = 4$. The equation has R_i on both sides, we are looking for the smallest value of R_i which satisfies the equation.

Calculating R_i

Slide 8

To solve the equation for R_i we have to form a recurrence relation should converge on correct value for R_i if there is a solution.

$$R_i^{n+1} = C_i + \sum_{\forall j \in \mathrm{hp}(i)} \left\lceil \frac{R_i^n}{T_j} \right\rceil C_j$$

This equation either converges or it will grow to exceed D_i in which case the task set has failed the scheduling test.

Example

Back to our example:

Slide 9

Name	T	C
A	52	12
B	40	10
C	30	10

We first want to calculate the response time R_A, since A has the longest period every task is at a higher priority.

So

$$\text{hp}(A) = \{B, C\}$$

Calculating R_A

We start be setting R_A^0 to be 0.

Slide 10

$$R_A^1 = C_A + \left\lceil \frac{R_A^0}{T_B} \right\rceil C_B + \left\lceil \frac{R_A^0}{T_C} \right\rceil C_C = 12$$

$$R_A^2 = C_A + \left\lceil \frac{R_A^1}{T_B} \right\rceil C_B + \left\lceil \frac{R_A^1}{T_C} \right\rceil C_C = 32$$

$$R_A^3 = C_A + \left\lceil \frac{R_A^2}{T_B} \right\rceil C_B + \left\lceil \frac{R_A^2}{T_C} \right\rceil C_C = 42$$

$$R_A^4 = C_A + \left\lceil \frac{R_A^3}{T_B} \right\rceil C_B + \left\lceil \frac{R_A^3}{T_C} \right\rceil C_C = 52$$

$$R_A^5 = C_A + \left\lceil \frac{R_A^4}{T_B} \right\rceil C_B + \left\lceil \frac{R_A^4}{T_C} \right\rceil C_C = 52$$

Calculating R_B and R_C

Slide 11

$$R_B^1 = C_B + \left\lceil \frac{0}{T_C} \right\rceil C_C = 10$$

$$R_B^2 = C_B + \left\lceil \frac{10}{T_C} \right\rceil C_C = 20$$

$$R_B^3 = C_B + \left\lceil \frac{20}{T_C} \right\rceil C_C = 20$$

So R_B has converge.

Since hp(C) is empty (C is the highest priority task)

$$R_C = C_C = 10$$

Slide 12

For our three tasks $R_i \leq D_i$ it is possible to schedule the tasks using a rate-monotonic assignment.

Summary so far:

- Liu and Layland's test provides a sufficient but not necessary test which is very easy to apply. If we don't know if it is possible to schedule the task set.
- When $D_i = T_i$ Joseph and Pandya's test is exact, if rate monotonic assignment works then the test will say yes.

Deadline Monotonic vs. Rate Monotonic

If we have a task set where $D < T$ then rate-monotonic assignment might not give the best assignment.

Slide 13

For example an infrequent but urgent task might be given a low-priority because its T is large and will in many cases miss its deadline.

There is a better assignment if we use *deadline monotonic assignment*, where the shorter the deadline the higher the priority. Then if the task set is unschedulable deadline monotonic assignment then it is unschedulable with all other orderings.

Deadline Monotonic Scheduling (DMA)

Slide 14

- **Task Model** Periodic tasks with the deadline within the period.
- **Deadline monotonic Assignment** - shorter the deadline higher the priority
- **Deadline monotonic Assignment is optimal** - if there is a priority based schedule that will schedule the tasks the DMA will.

Rate monotonic assignment is a special case of deadline monotonic.

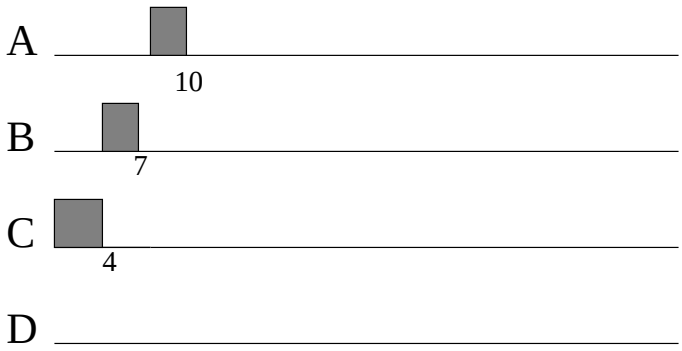
Example of DMS

Slide 15

	P	D	C	R	RMS Priority	DMS Priority
A	20	5	3	3	3	1
B	15	7	3	6	2	2
C	10	10	4	10	1	3
D	20	20	3	20	4	4

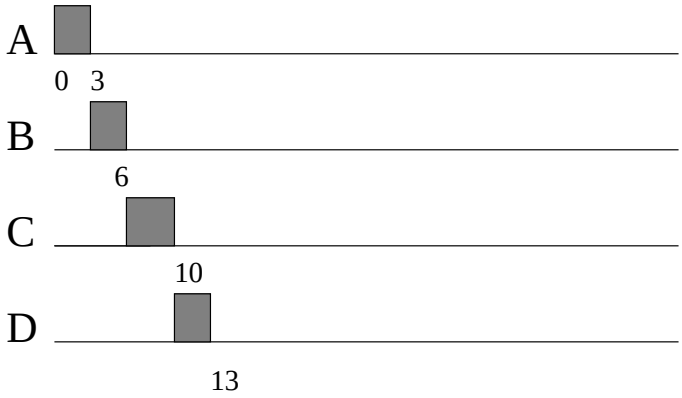
This task set won't work if we use simple rate-monotonic scheduling. But if we use deadline monotonic it will schedule.

Slide 16



Task A misses its deadline.

Slide 17



All tasks meet their deadline (continue the schedule to find out!).

Deadline Monotonic

Slide 18

Deadline Monotonic is optimal in the following sense.

> If it is possible to schedule the task set with fixed priority scheduling then it is possible to schedule it with deadline monotonic.