Solutions

Exercise 1: Polling

If we assume the simple case with $D = T$, the monitoring task may be excuted at any time within its period.

Worst case with regard to event detection occurs when the task in one period is run immediately at the period start, but is delayed until the very end in the next period.

In that case, if the event occurs at the start of the first period, it may just miss being detected there and then not detected and handled until the end of the next period. By inspecting a timeline, we see that this gives a reaction time for the event of $2T$ (indluding the event handling). Therefore we should have $2T \leq D_e$, where D_e is the deadline of the event handling. I.e. the polling task should run with a period not exceeding $D_e/2$ or in words:

> *The event polling task should have a period less than or equal to than half of the event deadline.*

The worst case computation time for the monitoring task will be that of the event handling C_e.

This may then be applied to the figures in [BW] Table 11.29, where the deadlines are halved to get the periods, but the computation times remain.

If we allow for lower deadlines than the period, we similarly see that $T + D \leq D_e$ should be satisfied for the monitoring process.

Exercise 2: Get acquianted with the Times tool

No solutions provided.

Exercise 3: Controlled Task

Question 3.1: See the file `Smallburst.xml`. Notice how the role of the two clocks are swapped in the two bottom states. A solution which resets both clocks to start a new round does not prevent three releases in a row and thus does not meet the specification.

[In the solution, an extra periodic task has been added in order to avoid some faulty behaviour of the Timestool Simulator.]

Question 3.2: In order to meet its deadline the burst task must receive top priority or a priority just below the b-task. This may be verified using `SmallburstExt.xml`

Since the verification may take some time on a standard computer, the burst task may be represented by a deterministic, worst-case task like `Control1ext.xml`. This represents the most condensed behaviour of the burst task, and thus gives a maximum load of the system.

From Exam F13: Problem 4

PROBLEM 3

Question 4.1

(a) Individual and total loads (utilization):

	T	C	U
a	5	2	40 %
b	10	3	30 %
c	25	4	16 %
Total			86 %

(b) For $N = 3$ the Liu-Layland criterion is 78.0 % which is exceeded by the total load. However, both 10 and 25 are multiples of 5 and hence task a may form a *family* with either task b or c (but not both). In either case, there will be two families (one being singleton) and hence the criterion should be tested for $N = 2$ where the limit is 82.8 %. As this is still exceeded, schedulability cannot be guaranteed by the Liu-Layland criterion.

[For this exam, the Bini criterion was not used. According to this, the product of $1 + U_i$ should not exceed 2. For the given tasks, the procduct becomes $(1.4 \times 1.3 \times 1.16) = 2.1112$. Hence schedulability of the task set cannot be guaranteed by this criterion either.]

Question 4.2

For the given tasks, rate monotonic priority assignment yields the order a. b, and c. The response times are then found using the iterative formula:

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

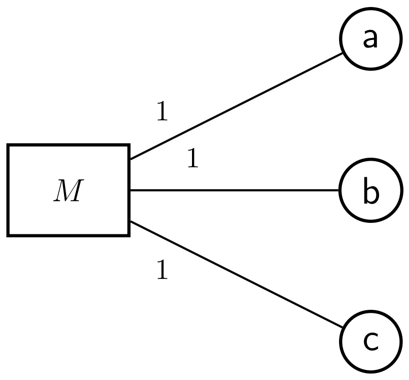
as presented in the following scheme

Task	T	C	R^0	R^1	R^2	R^3	R^4	...
a	5	2	2	$\underline{2}$				
b	10	3	3	5	$\underline{5}$			
c	25	4	4	9	11	16	18	$\underline{18}$

Hence, $R_a = 2$, $R_b = 5$, and $R_c = 18$.

Question 4.3

(a) The system may be illustrated by:

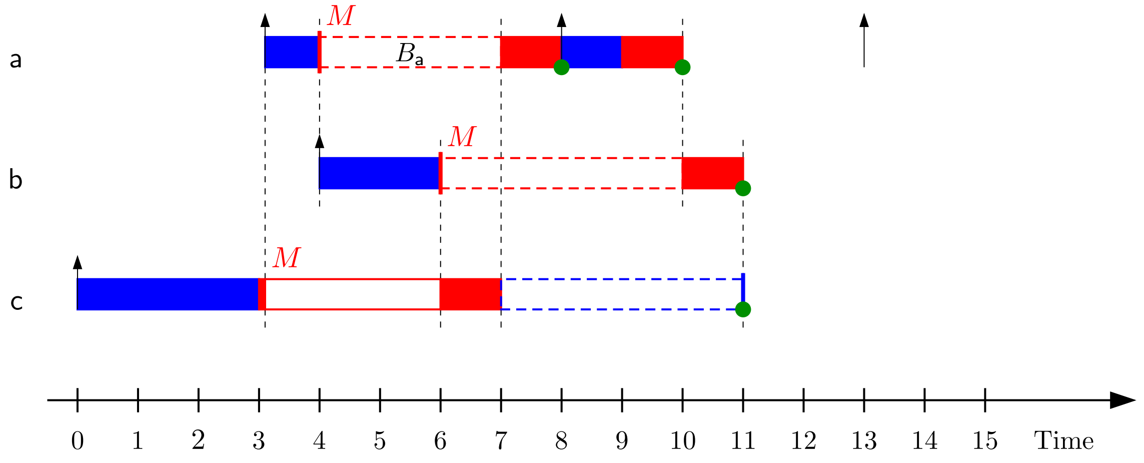


where the numbers on the connections indicate the maximum use of the given resource by the given task.

From this we conclude that $B_c = 0$ as the lowest priority task can never be blocked by any lower priority task.

Also $B_b = 1$ as task b can only be blocked by task c and only for at most 1 time unit.

(b) The worst case blocking time suffering from priority inversion can occur in the following scenario where tasks a is released right after task c has acquired the resource M and task b is released just when task a is blocked on M.



Here, task a is blocked until task b reaches the use of M and then further while task c finishes its use of M. The firt period is $C_b - 1 = 2$ and the second period is 1 giving a blocking time $B_a = 2 + 1 = 3$.

(c) Inserting the blocking times the response times are found by:

Task	T	C	B		R^0	R^1	R^2	R^3	R^4	...
a	5	2	3	5	$\underline{5}$					
b	10	3	1	4	6	8	$\underline{8}$			
c	25	4	0	4	9	11	16	18	$\underline{18}$	

As can be seen, the system is (just) schedulable.

Question 4.4

(a) By applying the *immediate ceiling priority protocol*, the blocking time B_a is now the maximum of any lower priority operations in M, hence $B_\mathsf{a} = 1$. The blocking times B_b and B_c remain 1 and 0 respectively. Inserting these into the response time scheme we get:

Task	T	C	B		R^0	R^1	R^2	R^3	R^4	...
a	5	2	1	3	$\underline{3}$					
b	10	3	1	4	6	8	$\underline{8}$			
c	25	4	0	4	9	11	16	18	$\underline{18}$	

As can bee seen, the response time R_a is significantly reduced by applying ICPP.

(b) As the lowest priority process by definition does not suffer from any blocking time, its response is not affected by ICPP (or any other priority inheritance mechanism).