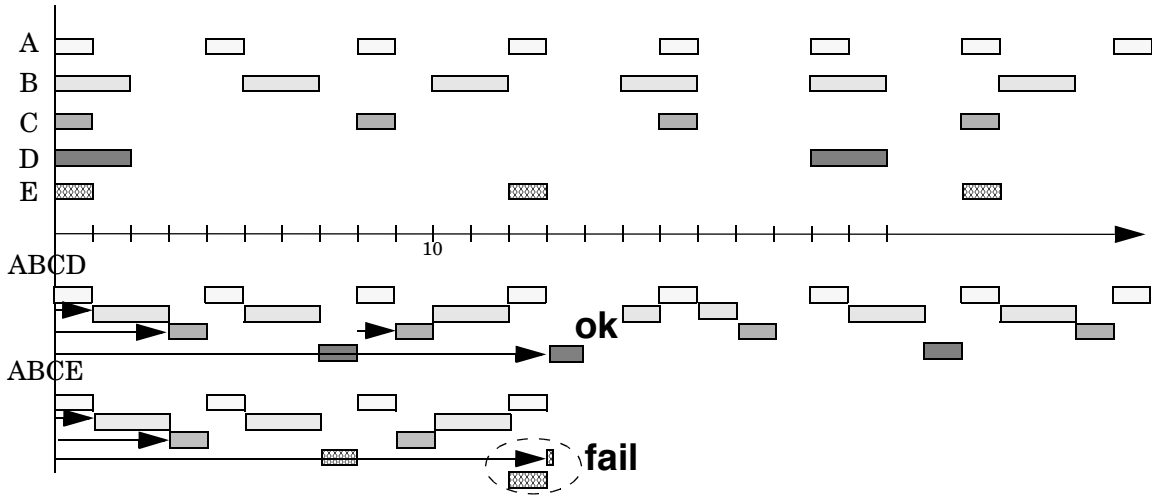
Exercise session 6 – Deadlock and scheduling : Solutions

1 Scheduling

a) The CPU-load due to the threads A, B, and C is 1/4+2/5+1/8 = 31/40 = 0.775 = 77.5%. This is above the general limit of 0.69 which ensures that the system can be scheduled. It is, however, below the specific limit of 0.78 for three periodic threads. Therefore, we know that these threads will all meet their deadline (which is equal to the period). To determine the actual response times, we need a more accurate analysis (the same analysis as we would need to tell if deadlines are met in the case of a CPU-load between 78% and 100%). Drawing the schedule (also for exercise b and c) gives:



When making such a schedule, we utilize that the worst case is when all threads are released at the same time. Here we use rate-monotonic scheduling (RMS) which means that shortest T runs first and so on. An actual schedule (see top three lines in the scheduling figure) then shows (as we already know) that the system is schedulable., and the worst-case response time R will be A: 1, B: 3, C: 4.

b) As the ABCD-part of the sheduling figure shows, the system is schedulable even with the thread D which gets the response time 14 time units.

c) The system ABCE is not possible to schedule because E cannot complete until t=13.2 which is past the deadline. Hence, not schedulable.

2 Scheduling with blocking

a) We obtain the following maximum blocking times:

A: max([M1:b+M2:t], [M1:c]) = [M1:b+M2:t] = 1.0+0.5 = 1.5

B: max([M2:t via A,C], [M1:c]) + [M3:y] = 0.5+2.0 = 2.5

C: [M3:y via B,D] = 2.0

Note that D will never be blocked since it does not share any resources with any lower priority thread. Further note that even if C does not directly share any resources with any lower priority thread, it can still be delayed by the time of the y operation; assume D has just entered y and C is preempted by B which will be blocked by D's call of y. We obtain a nominal maximum response time by adding the blocking times to the response time according to exercise 2. But additionally, higher priority threads might be scheduled to run again (possibly several times) within the nominal response time. Therefore, we iteratively have to add an appropriate (according the constructed schedule) number of execution time for additional invocations of higher priority threads. Hence:

	T	*C*	*R0*	*L*	*R*	
A	4	1	1	1.5	1+1.5 = 2.5	:-)
B	5	2	3	2.5	$3+2.5+n*C_A=5.5+1*1 = 6.5 >5$	**:-(**
C	8	1	4	2	$C_A+C_B+C_C+2+n_A*C_A+n_B*C_B=1+2+1+2+2+2=10 >8$	**:-(**
D	20	2	14	0	14	:-)

Where R0 denotes the response time without blocking, L denotes the worst case blocking time, and R denotes the worst-case response time we are trying to compute. Note that since the response time of thread B (in the first iteration) gets longer than the period of thread A, A will run a second time before B can complete its execution. Also note that when A executes the second time, it cannot be blocked by C since B is running, and it does not matter if A should be blocked by B since that is the thread we analyze. The maximum response time for B is 6.5 which is above the limit 5.0. Thus, we cannot ensure that the deadline is met.

Also C will not meet its deadline; the indirect blocking of 2 and the execution times will cause A to be invoked twice and B once in addition to their initial cycles. This gives the response time 10 which is too long.

b) After the rewriting of D and M3, the worst-case blocking time L for B will decrease to 0.5 + 0.5 = 1.0. The response time will then be 3.0+1.0=4.0 which means that B will meet its deadline. It is, however, quite sensitive since any additional computing time of A or B would cause A to be run once more before B completes and the response time would be slightly more than the deadline of 5. So even if the deadline in is met, having no margins means that we are very sensitive to errors in the estimated execution times. Anyway, B is formally OK, but what about C which did not meet its deadline either? With M3 rewritten we obtain the maximum blocking time of 0.5 (M3:y_k indirect via B,D). This gives the response time $R_C = C_A+C_B+C_C+L_C+n*C_A+m*C_B = 1+2+1+0.5+1*1+1*2 = 7.5 < 8$. Since D as the lowest priority thread will never be blocked waiting for resources, all deadlines are met.

c) To decrease the maximum blocking time for A one could merge the monitors M1 and M2, either by rewriting them or by using the priority ceiling or immediate inheritance protocols. That gives the response time 2 exactly. Another alternative is to rewrite M1 and B in such a way that the blocking time of 1.0 gets smaller. It is not clear from the facts given if that can be done in this application.