50.002 COMPUTATIONAL STRUCTURES

INFORMATION SYSTEMS TECHNOLOGY AND DESIGN

Scheduling Extra Question

Natalie Agus

Task	Service Time	Deadline	Frequency	Period
T	1 μs	10 μs	1000/s	1ms
N	?	25 μs	500/s	2ms
C	25 μs	–	333.333/s	3ms
A	10 μs	15 μs	250/s	4ms
J	15 μs	50 μs	200/s	5ms

Table 1: Schedules

NOTE: Unless stated otherwise, do NOT confuse yourselves between period and deadline. A task can happen multiple times and be in a queue repeated times. The most important matter is to meet its deadline

1. First thing first, if there is a *finite* deadline, check if period is **longer** than completion time. If period is shorter than completion time, that means the rate of input of task is faster than the rate of completion of task, and the queue will grow to infinity. No scheduling method can solve this with finite deadline. Secondly, check if period is **shorter** than the deadline, i.e: period is 2ms, deadline is 5ms. If period is shorter than the deadline, then it is as good as saying that the 'realistic' deadline is the period itself, even though the deadline is much longer than that because the existence of the deadline itself means that the task has to be completed. Maximizing the completion time to be closer to the deadline of 5ms, will cause the completion time to be longer than the period, which in turns violate the first condition above, rendering the scheduling problem unsolvable.

2. Before answering any questions regarding scheduling, it is also good to check the load % of this machine running these 5 tasks periodically:

   - The load from T is: $0.000001 \times 1000 = 0.1\%$
   - The load from N is: $x \times 500 = 5x\%$

- The load from C is: $0.000025 \times 333.33 = 0.8325\%$
- The load from A is: $0.000010 \times 250 = 0.25\%$
- The load from J is: $0.000035 \times 200 = 0.7\%$

The total load is 1.8825% + 5x%.

3. **Assuming weak priority scheduling**, what is the *maximum* service time of task N that still satisfies all constraints?

   - The intuition is to ensure that, since weak priority scheduling cannot interrupt ongoing process, if N happens to be the ongoing process, what is the longest service time N can have before it causes other tasks in the queue to miss the deadline?
   - Look at the task with the *shortest gap* between its service time and deadline, in this case its Task A
   - So, the maximum service time of N is limited by A's deadline and service time: 15 - 10 = 5 ms.

4. **Assuming strong priority scheduling of** $A > T > N > J > C$, what is the *maximum* service time of task N?

   - The intuition is to ensure two things. Firstly, that N can still meet its own deadline because it can be interrupted by either A or T, and that N is not *too long* such that it causes J to miss its deadline. We do not need to care about C since it has no deadline. We also do not need to care about A or T because they can always interrupt N, so N's service time doesn't concern A or T.
   - First lets consider N having to meet its own deadline despite being interrupted by A and T. At t=0, if task T, N, and A happens at the same time, the machine will choose A and T to be executed first, which in total takes 11 μs. The deadline for N is at t=25 μs, and between t=0 and t=25 μs, there will be no other A and T tasks invoked. Hence, this gives N a maximum service time of 25 - 11 = 14 μs.
   - Now lets consider N having to care about J meeting its deadline. At t=0, if tasks T, N, A, and J happens at the same time, the machine will choose A, T, and N to be executed first. It takes 11 μs to complete A and T. The deadline for J is at t = 50μs, and there will be no other T, A, and N tasks invoked within this time period. Since N will be executed first, it will have to finish within: 50 - 15 - 11 = 24 μs, otherwise J will miss its deadline.
   - So if both constraints have to be satisfied, which is J and N both to meet their deadlines, then, the maximum service time of task N is 14 μs.

5. **Assuming strong priority scheduling of** $A > T > N > J > C$, what is the:

   - **Worst case completion time for A?** A doesn't have to wait for anyone, it can interrupt all other task. Thus, 10 μs.

- **Worst case completion time for T?** T can only be interrupted by A. Now we need to consider how many times A can happen within 1 period of T. Turns out A can only happen once within 1ms. Hence, the answer is simply the service time of A plus the service time of T : 1 + 10 = 11 μs.
- **Worst case completion time for N?** N can be interrupted by A and T. Within 2ms (period of N), T can happen 2 times, and A can happen 1 time. N has to wait for A and T to complete first, taking 11 μs in total before starting. There is no other A or T task invoked after 11 μs. The next T task invoked will be at t=1ms. Hence, N can start immediately at t=11 μs. Taking the answer from above: 14μs for service time for N, the the worst case completion time for N is 11 + 14 = 25 μs.
- **Worst case completion time for J?** J can be interrupted by A, T, and N. Within 5ms, A can happen 5 times, T can happen 4 times, and N can happen 3 times. With the same logic, we consider how much time it takes to complete T, A, and N tasks that are all invoked at t=0: 1 + 10 + 14 = 25 μs. No other task will be invoked between 25 μ to 1ms (the next T task), so J can start immediately after 25 μs. Hence the worst case completion time for J is 25+15 = 40 μs.
- **Worst case completion time for C?** Following the same logic, the worst case completion time for C is 1+10+14+15+25 = 65 μs. This shows that the *minimum* deadline for C, if any, is 65 μs.

6. **Assuming strong priority scheduling of** $A > T > N > J > C$, what is the load of the machine? Taking the answer from above: 14μs for service time for N, then total load is 1.8825% + 5*0.000014% = 1.88257%

Now lets make the question slightly tougher by increasing the service time of each task by 50, and modifying some deadlines.

Task	Service Time	Deadline	Frequency	Period
T	50 μs	300 μs	1000/s	1ms
N	?	500 μs	500/s	2ms
C	1250 μs	–	333.333/s	3ms
A	500 μs	1000 μs	250/s	4ms
J	1750 μs	2000 μs	200/s	5ms

Table 2: Schedules 2

1. **Assuming strong priority scheduling of** $A > T > N > J > C$, what is the *maximum* service time of task N?

   - Similarly, we first consider about N meeting its own deadline. Within 0.5 ms, A and T will happen once. So, since A and T has higher priority than N, that means the maximum service time for N is 0.5 - 0.1 - 0.05 = 0.35 ms.

- Secondly, we consider about J meeting its own deadline. Within 2ms, T happens twice, A happen once, and N happen once. We don't need to care about C because it doesnt have a deadline. As long as load is lesser than 100%, C will eventually be completed. So, the in this case, the maximum service time for N is 2 - 0.5 - 0.05 - 1.75 = -0.3 ms. This means that there's no room for N to have any service time such that it doesnt cause J to miss its deadline
- Hence this scheduling **does not work** with these set of constraints (deadlines).

2. **What about weak priority scheduling?** Is there any weak priority scheduling that can meet all the constraints?
   - You need to check that if a particular task cannot be interrupted, does the service time of this particular task cause any other task to miss the deadline?
   - The answer is yes, for task J, if it has already started, everybody has to wait for 1.75ms. Which means T or N will miss their deadlines.
   - Hence no, this schedules 2 cannot be satisfied with weak priority scheduling.
3. **In fact, is this scheduling constraint solvable?**
   - You can answer this question by thinking whether or not the deadline for each task is too tight, meaning that is there any two pairs, i and j where task i cannot wait for task j to finish, AND vice versa.
   - In this case, task J cannot wait for task A to finish, and task A cannot wait for task J to finish.
   - Means we cannot put $A > J$ in priority ordering, neither can we do $J > A$ in the priority ordering. Hence this scheduling constraint cannot be solved with any measures, not even with strong measure.

Okay, now lets be realistic.

Task	Service Time	Deadline	Frequency	Period
T	50 μs	300 μs	1000/s	1ms
N	?	3500 μs	500/s	2ms
C	1250 μs	–	333.333/s	3ms
A	1500 μs	5000 μs	250/s	4ms
J	2800 μs	10000 μs	100/s	10ms

Table 3: Schedules 2

1. **Assuming strong priority scheduling of** $T > N > A > J > C$, what is the *maximum* service time of task N?

- Now, N can only be interrupted by T, and N still need to make sure that it fulfils its own deadline, as well as not to occupy the machine for too long such that A and J cannot complete.
- The idea is to find the maximum service time of task N from N's perspective, A's perspective, and J's perspective, and select one value that satisfies everybody
- Lets consider N's deadline at 3.5ms and its period at 2ms.
- If deadline is smaller than period, then of course we should use the deadline to compute this question, however it is not the case here
- Recall that **if completion time is larger than the period, then the scheduling problem is unsolvable with finite deadline**. Hence we should **not consider** the time window of 3.5ms to compute this (service time of N), but rather 2ms, which is the period of task N.
- Within 2ms, task T will be invoked for 2 times, and task N will be invoked for 1 time. From t=0, it will first wait for task T to finish (again at t=0, these periodic tasks T, N, C, A, and J are all invoked at the same time). Therefore the max service time for N so that at least the machine can run indefinitely without violating any deadlines is 2 - 2*0.05 = 1.9ms
- Now we have to check the maximum service time of N from task A's perspective
- We choose to use 4ms to compute this (period of A), and not deadline of A thats larger than its period with the same reasoning above.
- Within 4ms, T will happen 4 times, and N will happen 2 times, and A will happen 1 time. So the max service time for N is: (4 - 4×0.05 - 1.5)/2 = 1.15ms, otherwise A will "snowball" and will be eventually unable to meet its deadline (rate of production of task A is bigger than rate of completion of task A).
- Now we have to check the maximum service time of N from task J's perspective
- Again, we choose to use 10ms to compute this (period or deadline J, same value)
- Within 10ms, T will happen 10 times, N will happen 5 times, A will happen 3 times, and J will happen 1 time.
- So the max service time for N is: (10 - 10*0.05 - 3*1.5 - 2.8)/5 = 0.44ms, otherwise task J will not meet its deadline.
- Therefore to satisfy all constraints, **the maximum service time for N is 0.44ms**.
- Since task C does not have any deadlines then we do not have to care about its completion