

Real Time Operating System
Prof. Rajib Mall
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 05
Cyclic Scheduler (Contd.)

Welcome back. If you remember over the last few lectures, we are looking at some real time tasks schedulers. We had looked at very simple schedulers like table driven schedulers, but then we saw that it has its disadvantage requiring to set a timer again and again after each task's execution. We had remarked that time that the cyclic scheduler is extremely popular, because it does not require setting a periodic timer again and again; it just needs to set it once during the initialization. It requires to design tasks to be done, one is to set the major cycle for a set of periodic tasks and then also to set the minor cycle which is also called as a frame. Major cycle is rather simple to compute if you remember it is the LCM of the task periods. But then setting the frame size is not that straightforward it is non-trivial, we need to consider several design alternatives and may have to iterate between our frame sizes maybe we have to split the tasks and so on.

So, setting the frame size is an important design task while using a cyclic scheduler. Let us see how the frame size is to be set.

(Refer Slide Time: 02:17)

Selecting an Appropriate Frame Size (F)

- **Minimum scheduling overhead and chances of inconsistency:**
 - F should be larger than each task size.
 - **Sets a lower bound.**
- **Minimization of table size:**
 - F should squarely divide major cycle.
 - **Allows only a few discrete frame size.**
- **Satisfaction of task deadline:**
 - Between the arrival of a task and its deadline:
 - At least one full frame must exist.
 - **Sets an upper bound**

65

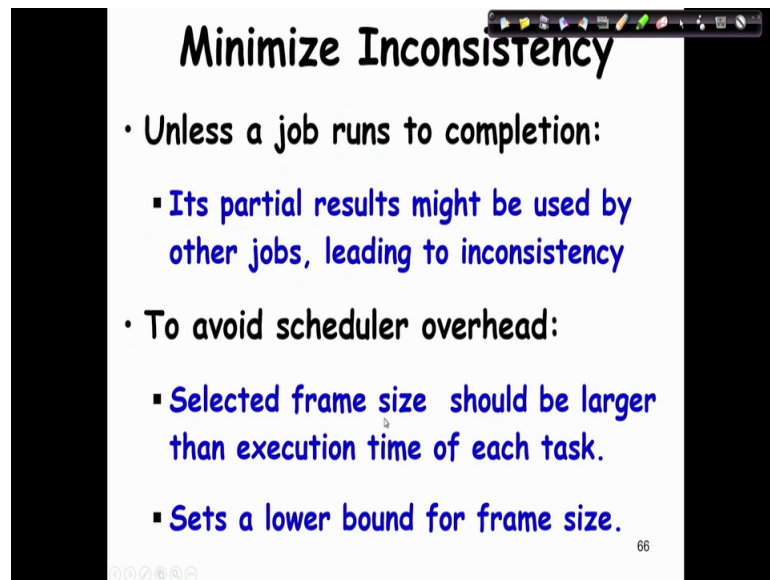
The first consideration is that each frame size must be larger than the execution time of every task. If you remember why this is so, it is because every frame size must hold the task or in other words if a task is taken up for execution in a frame, it must complete within the frame. It should not spill over to the subsequent farms and that is the reason a frame size should be larger than the largest task execution time.

So, based on this consideration, we get a lower bound for the frame. The frame if it become lower than the largest tasks size, then that is not a good schedule it will have problems as we will see. The second consideration is minimisation of the schedule table that we have to store. From this consideration, we require that the frame size that is chosen should squarely divide the major cycle or it must be a factor of the major cycle. Based on this consideration we can select only few discrete frame sizes. If the frame size does not squarely divide the major cycle, then we have to store the schedule for several major cycles. If the frame squarely divides the major cycle then after the major cycle the same tasks sequence will repeat. So, we need to store only the tasks schedule for one major cycle.

The third consideration is every task must meet it is deadline, and for this to happen there must be one full frame between the arrival of a task and it is deadline. To see why it is so, we need to consider that if a task arrives just after a frame has started, then the task cannot be taken up in that frame and if it is deadline also is within the frame, then the task execution cannot be started before it is deadline.

So, you must have at least one clear frame between a tasks starting and it is deadline. A corollary of this third condition is that the frame size has to be greater than every task period. So, this sets an upper bound for the frame size. So, a frame size must be less than every task period.

(Refer Slide Time: 05:54)



Minimize Inconsistency

- Unless a job runs to completion:
 - Its partial results might be used by other jobs, leading to inconsistency
- To avoid scheduler overhead:
 - Selected frame size should be larger than execution time of each task.
 - Sets a lower bound for frame size.

66

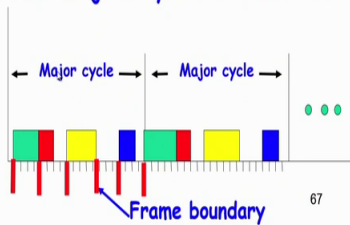
So, let us see we will get more explanation on this as we proceed. So, the first condition is that the frame size must be chosen such that every task must be able to execute in the frame size because if that does not happen then we have to run the task across multiple frames. So, in that case other tasks may run in between and then you use the partial results obtained by executing a task halfway or may be a partially and then that may lead to inconsistency.

So, normally we would need that frame size should be larger than the execution time of every task the second condition. So, the first condition we just saw that, that the selected frame size should be larger than execution time of each task.

(Refer Slide Time: 07:07)

Minimization of Table Size

- Unless the minor cycle squarely divides the major cycle:
 - Storing schedule for one major cycle would not be sufficient.
 - Schedules in the major cycle would not repeat:
 - This would make the size of the table large.



67

The second condition is minimisation of the table size. So, this is the schedule table which we need to store the schedule table if you remember is for every frame, which task is to be run. So, as the task as the frame interrupts come the scheduler just takes up the task by consulting the table.

So, a first frame we took up the green task, the second frame the red task, third frame yellow task and so on. And after the major cycle the same sequence repeats, that is because the frame last frame boundary coincided with the major cycle and by this time all the tasks are run. So, this is the LCM of all the task period and therefore, every task period also has the ending of that has coincided here, but if the frame boundary does not coincide with the major cycle boundary it spills over then we have to run for we have to store the schedule table for many major cycles. And from this consideration we get the condition that the frame size should divide the major cycle, it should be a factor of the major cycle and that gives us only few discrete choices to select the frame size.

(Refer Slide Time: 08:51)

Satisfaction of Task Deadline

- Between the arrival of a task and its deadline:
 - At least one full frame must exist.
- If there is not even a single frame:
 - The task would miss its deadline,
 - By the time it could be taken up for scheduling, the deadline could be imminent.

68

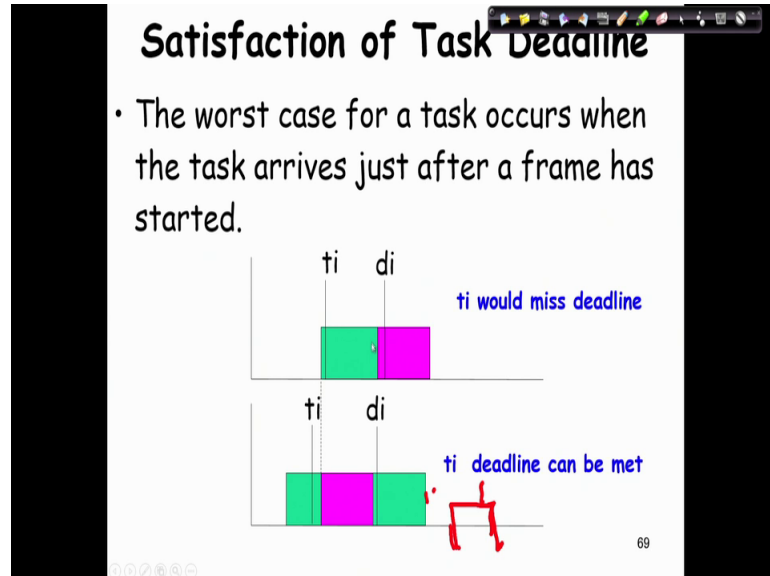
Now, let us look at the third condition, which says that the frame size should be chosen such that every task deadline must be met. So, let us see; what is the implication of this condition, let us say we have this frame boundary starting at this point. So, we have the frame boundary starting at this point. Now let us assume that after a delta time after some time the task instance t_i arrived at this point.

Obviously, the task cannot be taken up during this frame because the if you remember the scheduler code that it just executes the task and then the scheduler ends the scheduler comes up just before every frame, starts execution of the task consults the schedule table finds out which task to run it runs and then it stops. So, it can only run a start at the starting of a frame boundary, but here the task has arrived just after the frame boundary; obviously, it cannot run on this frame, and then it can only be taken up in the next frame next frame boundary.

But then when the task is started to execute at the next frame boundary, the deadline is very near it cannot meet it is deadline. And this condition requires that we must have at least one full frame between the arrival of the task and the deadline, because if there is full frame existing here let us say we have frame here then if the task arrives here then it can at least be taken up execution in the next frame, and we know that the frame size is larger than every task execution time. And therefore, it will complete within the frame and its deadline is after that.

So, the third condition requires that we must have one clear frame existing between the arrival of a task and its deadline.

(Refer Slide Time: 11:49)



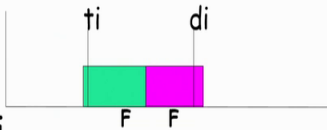
So, let us look at the satisfaction of the task deadline, we just saw that the worst case for a task occurs when the task arrives just after the frame has started. Because if the task arrives towards the end of a frame, then the next frame it can be taken up. So, the worst case occurs if a frame has just started and the task arrives. Now let us compute how much is this, what is the worst case scenario because every task if it arrives anywhere towards the end of a frame that is not a worst case, because it can immediately be taken up in the next frame.

So, the maximum time a task would have wait if it arrives just after a frame boundary.

(Refer Slide Time: 13:05)

Satisfaction of Task Deadline

- The minimum separation of arrival time of t_i from a frame start:
 - $GCD(F, p_i)$
- Thus, for all t_i
 - $2F - GCD(F, p_i) \leq d_i$ must be satisfied



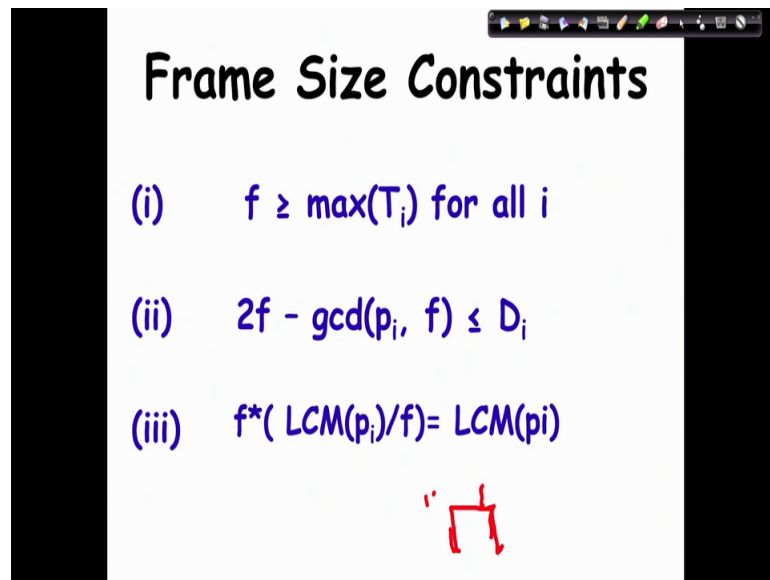
70

So, we need to do a little bit of mathematical derivation and we will not really go into derivation it is there in the book, here we just look at the result, the result is GCD the frame size and the period of the task. So, the greatest common divisor between the frame size and task period that gives the worst case situation of how much after the frame starts can a task arrive.

So, if the GCD is one then at most after one time unit the task can occur. So, from this consideration considering the worst case situation, then we have $F - GCD$, $F - p_i$ is the waiting time here. Because it cannot be taken up for scheduling during this duration and therefore, it must wait for $F - GCD - GCD - F, p_i$ and then we must have another clear frame. So, as long as $2F - GCD - F, p_i$ is less than equal to d_i , then we have a clear frame available between the tasks arrival and the deadline even in the worst case situation.

So, the third condition we can write in this expression and we will see given a tasks set whether it satisfies these and then if it satisfies, then we can take the frame size as feasible otherwise we have to discard that frame size.

(Refer Slide Time: 15:04)



Frame Size Constraints

- (i) $f \geq \max(T_i)$ for all i
- (ii) $2f - \gcd(p_i, f) \leq D_i$
- (iii) $f \cdot (\text{LCM}(p_i)/f) = \text{LCM}(p_i)$

"

So, written down the 3 conditions here, the first condition is that the frame size f must be greater than the execution time of every task. So, the maximum execution time of the tasks, even that that f should be greater than even the maximum execution time of a task.

The condition that there must be a clear frame between the arrival and the deadline gives us this condition, the worst case $2f$ minus $\text{GCD } p_i \text{ comma } f$ the task period and f it should be equal to the deadline of the task, that must be satisfied for every task i . The third condition is that the frame size must squarely divide the major cycle. The major we have written the $\text{LCM } p_i$ and this is the integer division f and when multiplied by f if we get back the $\text{LCM } p_i$, then you can say that the frame size squarely divides the major cycle.

(Refer Slide Time: 16:27)

Selection of a Suitable Frame Size

- Several frame sizes may satisfy the constraints:
 - **Plausible frames.**
- **A plausible frame size has been found:**
 - **Does not mean that the task set is schedulable.**
- Also, the largest plausible frame size needs to be chosen:
 - **Scheduler overhead would be lower.**

Now, let us consider the design trade ups, we find that there may be several frame sizes which may become feasible we call that as plausible frames, but when just because the they satisfied the 3 conditions and have got a plausible frame, does not mean that we have found a schedule, we have to check other things. The first thing we need to check whether the number of frames that we require to schedule the task exists. Let us say the task would not runs 3 times in the major cycle. So, the period of the task is one third of the frames of the major cycle, and the second task it runs 2 times during the major cycle. So, it is period is half the major cycle. So, we need five frames for these 2 tasks.

So, the frame size must be chosen such that there must be five frames, within the major cycle. Let us look at as we look the example it will become clear. If I find that multiple frame sizes become possible, then we need to choose the largest of those frame sizes; let us say the frame size 5 and 8 both of these are possible they satisfy all the 3 conditions and also they give rise to enough frames where all the instances arising during the major cycle can be scheduled, then we need to choose the larger of the frame size that is 8 why is that. Why do we need to choose the larger of the frame size a little thinking will show that at every frame boundary, the scheduler needs to run?

So, if we choose 5 as the frame size then there will be many frame boundaries compared to when we choose 8 as the frame size. So that means, the scheduler runs more frequently if we choose a smaller frame size compared to a larger frame size or in other

words the overhead of the tasks scheduler becomes more when we choose a smaller frame size and therefore, we always choose a the largest frame size that is possible. So, the scheduler overhead is lower when we choose a large frame size.

(Refer Slide Time: 19:40)

Cyclic Scheduling Example 1

- Job=(computation time, period, relative deadline)
- A = (1, 6, 6)
- B = (2, 8, 8)

- Major cycle = 24
- Frame sizes: 2,3,4,6,12,24
- For F=2
 - $2 \cdot F - \gcd(f, P_A) = 2 \cdot 2 - 2 = 6$ Acceptable
 - $2 \cdot F - \gcd(f, P_B) = 2 \cdot 2 - 2 = 8$ Acceptable

Now, let us take a few examples the first example we have 2 tasks A, B and the first element is here is 1 is the computation time. So, a takes one unit of computation it is period and deadline are the same. So, that a deadline occurs exactly at the period. So, it repeats after every 6 time units and before it repeats before the next instance comes, the first instance must have been over that is the implication of the relative deadline. The second task b execution time is 2 the period and deadline are both 8.

Now, our task is to choose the major cycle and the minor cycle choosing the major cycle is rather straight forward you just take LCM of 6 and 8, and I find that the least common multiple of 6 and 8 is 24. 24 holds an integral number of a task A and task B. So, we can see that there are 4 instances of task A, 3 instances of task B within one major cycle. So, within one major cycle we will have 7 instances altogether considering both A and B.

Now, let us try to choose the frame size. If you remember that the frame size must squarely divide the major cycle, that we get from the schedule table consideration minimising the size of the schedule table and therefore, we can get 24 12 6 4 3 2 these are the frame sizes that are that we can consider for the other conditions. Why not one we have not written one here because the size of the task B is 2 and every frame size must

hold must be greater than or equal to the largest task size and therefore, one we have not considered consider 2, 3, 4, 6, 12, 24.

Now, let us see we start with a smallest frame size and see if that becomes possible to satisfy the other conditions. So, let us check F equal to 2 and we will check if there is a full frame existing between the arrival of the task and its completion in the worst case and I had expressed it in the formula $2 \times F - \text{GCD}(f, \text{period})$. So, $2 \times 2 - \text{GCD}(2, 6) = 4 - 2 = 2$ and therefore, $2 \times 2 - \text{GCD}(2, 6) = 2$ and the period is 6.

So, the GCD of 2 and 6 is 2 and therefore, we get $4 - 2 = 2$ and 2 is less than equal to 6. So, this is fine similarly we can check for the second task n. So, $2 \times 2 - \text{GCD}(2, 8) = 4 - 2 = 2$ which is equal to $4 - 2 = 2$, 2 is less than equal to 8. So, frame size 2 is and also the number of frames that it can give us is $24 \div 2 = 12$. So, 12 frames to 7 of those frames we can assign an instance of A or B as required and 5 frames will remain unutilised.

(Refer Slide Time: 24:57)

Example 1

- For F=3
 - $2 \times F - \text{gcd}(F, P_A) = 6 - 3 \leq 6$ Acceptable
 - $2 \times F - \text{gcd}(F, P_B) = 6 - 1 \leq 8$ Acceptable
- For F=4
 - $2 \times F - \text{gcd}(F, P_A) = 8 - 2 \leq 6$ Acceptable
 - $2 \times F - \text{gcd}(F, P_B) = 8 - 4 \leq 8$ Acceptable

But not enough frames available!

- We can choose F=3

$A = (1, 6, 6)$
 $B = (2, 8, 8)$

Now, let us look at frame size 3. So, again we apply the similar conditions here $2 \times F - \text{GCD}(f, p_i)$ we find that it is check for the second task also we find. So, even frame size 3 is and we have $24 \div 3 = 8$ frames available in one major cycle and we have 7 task instances in a major cycle and therefore, even this is acceptable now let us look at F equal to 4. In F equal to 4 again we use these 2 expressions and we find

that both of these again become acceptable, but then the number of frames become in available is 6, 24 by 4 is 6 frames and we cannot schedule 7 task instances in 6 frames.

So, not enough frames are available and therefore, F equal to 4 we cannot choose and no point in looking further ahead 6 8 etcetera, because the number of frames becomes still smaller and therefore, 3 is the largest frame size for which we can have a possible schedule and based on this our design choice will be the frame size equal to 3.

(Refer Slide Time: 26:38)

Example 2

- **A** = (1, 10, 10)
- **B** = (3, 10, 10)
- **C** = (2, 20, 20)
- **D** = (8, 20, 20)
- **Major Cycle**
 - Lcm (10, 10, 20, 20) = 20
- **Frame Size**
 - 1, 2, 4, 5, 10, 20 (must divide Major Cycle)
 - $f \geq \max\{1, 2, 3, 8\}$ (geq longest computation time)
 - f can be 8, 10 or 20
- Not enough frames in a major cycle to run all the jobs

75

Now, let us look at another example we have 4 tasks sets A, B, C, D with execution time of 1, 3, 2, 8 and their periods are 10, 10, 20, 20. Now how do we find the major cycle? The major cycle is the LCM of the task periods and here there are only 2 periods 10 and 20. So, the LCM of 10 and 20 is 20. Now the next thing is to choose the frame size. So, the frame size must be larger than the execution time of all tasks greater than or equal to all tasks and also it has to satisfy the other conditions.

So, you find that it must divide the major cycle and based on that condition we get 1 2 4 5 10 and 20, but then it must be larger than 8 and therefore, we have only 10 and 20 is the possible frame size. So, in none of these cases, the frame size can be either 10 or 20. So, if it is 20 we have just one frame and if it 10 we have just 2 frames. So, in none of the cases we can have a schedule worked out, because we have 2 instances of these 2 instances of these and one of these and one of these in one major cycle. So, that gives us

6 instances and we need at least 6 frames, but then we are getting only either 1 or 2 frames.

We cannot schedule these tasks set, if you can see the major culprit here is this 8 here. As we proceed we will see how to handle this situation this task set is becoming un-schedulable, how do we make it schedulable on a cyclic scheduler. We will stop here and we will continue in the next lecture at from this point.