

Real Time Systems
Professor. Rajib Mall
Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur
Lecture 13
Frame size selection: Examples

Welcome to this lecture. In the last lecture, we were discussing about the cyclic scheduler. Just to recollect, it is one of the most popular clock driven scheduler used in very simple systems. The scheduler itself is used as the executive. Typically, the code for the scheduler is written, small code by the programmer, as well as the small application program and is run on small applications, small embedded systems.

And the advantages of the cyclic scheduler included simplicity, reliability and it is demonstratable fault freeness. So, that in safety critical applications, this is operating system of choice, small systems. But we were saying that here we have two things, one is the major cycle, which is basically the lowest common multiple of all the periods of the tasks in the system and that is called as a major cycle.

And the major cycle is divided into an integral number of minor cycles or frames and one task; here we had also said that the concept of task is a bit diluted in the sense that a task is actually a function. It is not a task in the real operating system sense, context switching and so on, context, tasks context and so on is not there, it is just a function or a set of functions which are run at a specific time and each frame contains a task.

And this schedule of which task to run on which frame is stored in something called a schedule table. But one of the design challenges here is selection of a suitable frame size, which we were discussing last time. And we had identified three constraints that a suitable frame size must satisfy and let us start from there.

(Refer Slide Time: 03:10)

Three Frame Size Constraints

- (i) $f \geq \max(e_i)$ for all i Frame size larger than the largest task execution time.
- (ii) $f * (\text{LCM}(p_i)/f) = \text{LCM}(p_i)$ Frame size divides major cycle.
- (iii) $2f - \text{gcd}(p_i, f) \leq D_i$ A full frame exists between task arrival and deadline.

The slide includes a diagram of a timeline with vertical lines representing task arrivals and deadlines. A red scribble is present over the diagram. A video inset in the bottom right shows a man in a white shirt speaking. Logos for IIT Bombay and IIT Madras are visible at the bottom.

So, there are three frame size constraints. The first constraint is that the frame size must be larger than the execution times of all tasks and this is intuitively obvious, because every task has to complete in one frame. Because here we do not have concept of a task being pre-empted, another task running and so on, this is just a set of function called very simple executed. So, here the frame size is larger than all execution times of tasks. So, this is intuitively obvious.

The second constraint is that the major cycle contains an integral number of frames, we write it like this, that LCM of the tasks which is the major cycle divided by f into f is the LCM, which really means that there is an integral number of frames in a major cycle and we were discussing the last class, that unless we have an integral number of frames in a major cycle, we need to store a schedule, schedule length in the schedule table will be much larger, we cannot just store the schedule up to the major cycle.

And this constraint comes from there that we need to store a small schedule, which is up to the major cycle. If the frame size does not divide the major cycle, so then we will have to store a much larger schedule. The third constraint is $2 * f - \text{gcd}(p_i, f) \leq D_i$, deadline for every task, for every task and as we had mentioned, that these are small systems and number of tasks are few, but still for every task, we need to check whether $2 * f - \text{gcd}(p_i, f) \leq D_i$, the deadline of the task.

And the reason behind this constraint is that if a task becomes ready just after a frame size, just after a frame interrupt, a clock interrupt marking a frame, it cannot be taken up in that frame, it needs to be taken up in the next frame, because after a task only becomes ready, it

can run on a frame and unless there is a full frame, the task would not meet its deadline. Just to recollect that point, if this is the timeline and we have this frame, interrupts coming from a cyclic, periodic clock, the frame 1, frame 2, frame 3, etc.

And let us say a task becomes ready here at this point, then it cannot run on this frame because the task to run on a frame is decided at the frame boundary. So, it cannot run on this current frame, it needs to run here and if the, as long as the deadline is after the next frame, if this is the deadline, this is the task arrival and this is the deadline and in between there is a frame, then the task can possibly be run here.

But if the deadline is before the next frame, even if the task is run here, it may not meet its deadline. So, that is the reason why we have this third constraint $2 * f - \gcd(p_i, f) \leq D_i$.

(Refer Slide Time: 08:52)

The slide contains the following text and handwritten annotations:

- Job=(computation time, period, relative deadline)
- T1 = (1, 6, 6)
- T2 = (2, 8, 8)
- Major cycle = 24
- Possible Frame sizes: 2, 3, 4, 6, 12, 24
- For F=2
 - $2 * F - \gcd(f, p1) = 2 * 2 - 2 = 6$ Acceptable ✓
 - $2 * F - \gcd(f, p2) = 2 * 2 - 2 = 8$ Acceptable ✓

Handwritten annotations include: a yellow box labeled 'Example 1', a list of numbers '1, 2, 3, 4, 6, 8, 12, 24' with '1' and '24' circled and 'x' marks, and '26' and '28' written above the constraint equations.

Now, let us take few examples and see whether we can select the frame size and here the jobs are given in terms of their computation time, their period and the relative deadline. There is a periodic task, 1 is the computation time and 6 is the period and 6 is also the relative deadline. So, the period and the relative deadline are the same. Second task 2 is the execution time or the computation time and 8 is the period and 8 is also the relative deadline.

Now, the first thing we need to do is to determine the major cycle and the major cycle is the LCM of the periods and the LCM of the periods 6,8 are the periods, if we take LCM of that, then the major cycle turns out to be 24, LCM 6,8 is 24. Now, the next step is to determine the feasible frames, because the first constraint in selection of the frame says that it must squarely

divide the major cycle and then we have only few options. For example, we have 24 itself, 12, 8, 6, 4, 2, and 1.

Now, so let me just write down 1, 2, 4, 6, 8, 12, and 24, these are the possible frame sizes which squarely divide the major cycle. But 1 cannot be a feasible frame size, because one of the task execution time is 2 and the frame size needs to be more than the largest execution time of the task and therefore, 1 is ruled out and similarly, 24 is ruled out. The reason is that in a major cycle, we need at least two frames, so that we can assign these two tasks into the two frames.

If we have one frame, how do we assign two tasks, we cannot. So, 24 is ruled out, so we need to check between 2, 4, 6, 8, 12. So, that is what written here, but again 24 is ruled out. So, 3 is also there. Because 3 is also squarely divides 24. So, 2, 3, 4, 6, 12, 6, 8, and 12, 6, 8 and 12. Now, let us check the frame sizes which are suitable, which satisfy the third constraint. All these 2, 3, 4, 6, 8, and 12, they satisfy the first two constraint, we need to check the third constraint, which is there is a frame between the arrival of a task and its deadline.

And that we have expressed in the form of a formula $2 * F - \gcd(f, P_i) \leq D_i$, so let us just check for $F = 2$. Here, $2 * F$ is 4 and $\gcd(f, P_1)$ is 2. So, $4 - 2 = 2 \leq (D_i = 6)$, and therefore this is okay. Now, let us check for the next task whether frame size 2 is suitable. For the next task, $2 * F = 4$. Now, $\gcd(2, P_2)$ is 2. So, again, $4 - 2 = 2 < 8$. So, frame size of 2 is okay, we can use frame size of 2, but let us check the other frame sizes.

(Refer Slide Time: 14:15)

Example 1
Cont..

- For F=3
 - $2 * F - \gcd(F, P1) = 6 - 3 \leq 6$ ✓ Acceptable
 - $2 * F - \gcd(F, P2) = 6 - 1 \leq 8$ ✓ Acceptable
- For F=4
 - $2 * F - \gcd(F, P1) = 8 - 2 \leq 6$ ✓ Acceptable
 - $2 * F - \gcd(F, P2) = 8 - 4 \leq 8$ ✓ Acceptable
- For F=6
 - $2 * F - \gcd(F, P1) = 12 - 6 \leq 6$ ✓ Acceptable
 - $2 * F - \gcd(F, P2) = 12 - 2 \leq 8$ ✗ Not Acceptable
 - What are Pros and Cons of choosing F=4?

T1 = (1, 6, 6)
T2 = (2, 8, 8)

Now, what about frame size of 3. So, again, we do $2 * F = 6$, $\gcd(3, P1) = 3$, $6 - 3 \leq 6$. So, this is also okay. Now, let us check for the other task $2 * F - \gcd(F, P2) = 5 \leq 8$. So, again, frame size 3 is okay, we can use frame size 3.

Now, let us check for frame size 4. Again, we check $2 * F = 8$, $\gcd(F, P1) = 2$. So, $8 - 2$ is just equal to 6 and it is acceptable. Now, what about for the second task, $2 * F$ is 8, $8 - \gcd(F, P2) = 4$. So, $4 \leq 8$. So, even frame size of 4 is okay, we can use frame size 4.

Now, let us check frame size of 6. So, here again, if we apply the formula for the first task, it becomes acceptable. What about for the second task, $2 * F - \gcd(F, P2) = 10$. So, $10 \geq 8$. So, here we have a, it does not satisfy, we cannot use 6 and if we find that it is 6 is not suitable, we need not look for higher frame sizes, that is unlikely to satisfy. So, we stop here.

But now the question is that whether we should use $F = 2, 3$, or 4. Now, let us check the advantages of each one of this. If $F = 2$ the frame size is 2 and one of the task size is 2, and if that task size takes a little bit longer. For some reason, maybe because there is some problem with hardware, or maybe it took a different path in the code or whatever, the system will fail. So, there will be overrun of the frame and system will fail.

So, if we take $F = 4$, that way, we are okay, if it takes little bit, any of the tasks take little bit longer frame overrun will not occur. But again, if we take $F = 4$, a major cycle will have 6 frames and out of that, we will be using 2 frames for our 2 tasks. But now let us say we want to update our system and we need to accommodate more tasks, then we will have fewer frames.

If we took $F = 2$, then we will have 12 frames. Out of that we use 2 and 10 empty frames are there, which you can do for, use for other purposes. For example, if we need more tasks in the future or maybe run some tasks, which are not critical tasks that we will just look at later. So, that leaves more frames idle, makes it more flexible, but difficult for prevents frame overruns.

So, that is an advantage for taking $F = 2$ and $F = 4$ and depending on the system, if our system it need not, we know that it would not be upgraded to an extent that we need so many frames, then we better go for $F = 4$, the frame size is 4. But if we know that we will need those extra frames, then we would go for $F = 2$. So, those are the pros and cons of choosing $F = 4$. Frame over run is unlikely to occur $F = 4$, but we have fewer empty frames available.

(Refer Slide Time: 19:51)

Example 2

- Compute a suitable frame size for the following task set:
- $T_1 = (1, 4, 4)$
- $T_2 = (1, 5, 5)$
- $T_3 = (1.5, 20, 20)$
- Major cycle = 20
- Possible Frame sizes = 2, 4, 5, 10

Now, let us take another example. Here we want to compute a frame size for a task set T_1 , the computation time or the execution time is 1, the period is 4 and deadline is 4, relative deadline is 4. T_2 , it is 1, 5, 5 and T_3 is 1.5, 20, 20 and the first task is to determine the major cycle and the major cycle is the LCM of the periods. So, what will be the major cycle, LCM of 4, 5, and 20?

So, that becomes 20. The LCM of 4, 5, and 20 is 20. So, the major cycle is 20. But then what are the possible frame sizes, the frame sizes need to be a multiple of 20, it needs to divide 20. So, that leaves us 1, 2, 4, 5, 10, and 20 itself. But since there are 3 tasks, we cannot really use 20. Because if we choose frame size 20, we will have only one frame, how do we run 3 tasks in one frame, because each task needs to be assigned to one frame.

What about 10, 10 also we cannot, because we need at least 3 frames and 10 will give us only 2 frames. So, this is also not suitable and what about 1, 1 also is not suitable from the first constraint that the frame size must be larger than the execution times of all tasks and we have a task here, which is 1.5. So, this also is ruled out.

The possible frame sizes are 2, 4 and 5. Now, let us check whether 2, 4, 5 satisfy the third constraint, which is existence of a full frame from the task arrival time to its deadline. So, the possible frames sizes are 2, 4, 5, even 10 also we cannot use, because that will leave us only with two frames and we have 3 tasks. So, 2, 4, 5 are really 3 frames, we need to check whether they satisfy the third constraint.

(Refer Slide Time: 22:43)

Example 2

Major cycle = 20
 Frame size = 2, 4, 5, 10

For F=2,

$2 * 2 - \text{gcd}(F, 4) = 4 - 2 \leq 4$ OK

$2 * 2 - \text{gcd}(F, 5) = 4 - 1 \leq 5$ OK

$2 * 2 - \text{gcd}(F, 20) = 4 - 2 \leq 20$ OK

•T1 = (1, 4, 4)
 •T2 = (1, 5, 5)
 •T3 = (1.5, 20, 20)

Now, let us check the third constraints. So, first, let us check for 2, 4, 5. So, first, let us check frame size is 2. Now, $2 * F - \text{gcd}(F, 4)$ because the period of the first task is 4. So, $2 * 2 - \text{gcd}(2, 4)$ is $4 - 2$, so this is okay. Now, what about for the second task, $2 * 2 - \text{gcd}(F, 5)$ and F is 2, so $2 * 2 - \text{gcd}(2, 5)$ is $4 - 1$, so $3 < 5$ this okay.

Now, what about the third task, $2 * 2 - \text{gcd}(F, 20)$, which is $4 - 2$, $\text{gcd}(2, 20) = 2$, which is $4 - 2 = 2 < 20$. So, frame size 2 can be used. Now, let us check for frame size 4.

(Refer Slide Time: 23:52)

Major cycle = 20

Frame size = 2, 4, 5, 10^{*}

For F=4,

$2*4 - \gcd(F,4) = 8 - 4 = 4$ OK



$2*4 - \gcd(F,5) = 8 - 1 = 7$ Not OK^X

$2*4 - \gcd(F,20) = 8 - 4 = 4$ OK

Frame size=2 can be chosen

Example 2

- T1 = (1, 4, 4)
- T2 = (1, 5, 5)
- T3 = (1.5, 20, 20)

So, frame size 4. So, $2*4 - \gcd(F,4)$, which is $8 - \gcd(4,4)$ is 4, so $8 - 4 \leq 8$, this is okay, is just equal to 4. Now, what about the second task, $2 * 4 = 8$. Now, $\gcd(F,5)$ is gcd of 4,5 for the second task the frame size is 4.

So, GCD, 4,5 is 1 and therefore $7 \geq 5$. So, we cannot use frame size 4. So, the only frame size 2 is a is feasible. We can use that. So, even though we have the other task still satisfies the third constraint, but since one task does not satisfy, we cannot use that. So, for this problem frame size of 2 can be chosen.

(Refer Slide Time: 25:11)

What If No Frame Size is Suitable?

- Possible culprit is a task with large execution time e.g. (20,100,100):
- Prevents a smaller frame size to be chosen.
- Try splitting task with large execution time into two or three sub-tasks.
- **Example:** $T_1=(20,100,100)$ can be split into $(10,100,100)$ and $(10,100,100)$.

The slide features a video inset of a man in a white shirt speaking, and logos for IIT Bombay and NPTEL at the bottom.

But it may be the situation that none of the frame sizes is suitable, given a task set, we compute the major cycle, list out the feasible frames and then check for the third and see that none of the frame sizes is suitable. What do we do then, can this task be run? Yes, it can be run, but we have to do a bit of jugglery design and programming jugglery, we need to split the largest task into two smaller tasks and still it will run, it will run in one, the required time, but we just split it and run it in two instances.

So, that requires a little bit of programming overhead. So, the task like requiring 20 execution time will really make the frame size to be more than 20 and therefore, the existence of a frame size of, frame of size 20 between the arrival of task and the completion of a task may not be feasible for many of the tasks.

And therefore, if we can somehow reduce the 20 into smaller number, then we can find a frame where the task can run. So, we would split the task and since, as I was mentioning that in this cyclic scheduler, the entire operating system and the program is written by the programmer, he can just do a bit of manipulation, split the largest task into two smaller tasks.

For example, 20, 100, 100 will be split into one task which is 10, 100, 100 and another task which is 10, 100, 100 or it can be 12, 100, 100 and 8, 100, 100 whatever is the best the programmer can do, just reduce the size of that task and then it will become feasible, we can find a feasible frame size on which the application can be run. We are almost at the end of this lecture. We will stop here and we will continue from this point in the next lecture. Thank you.