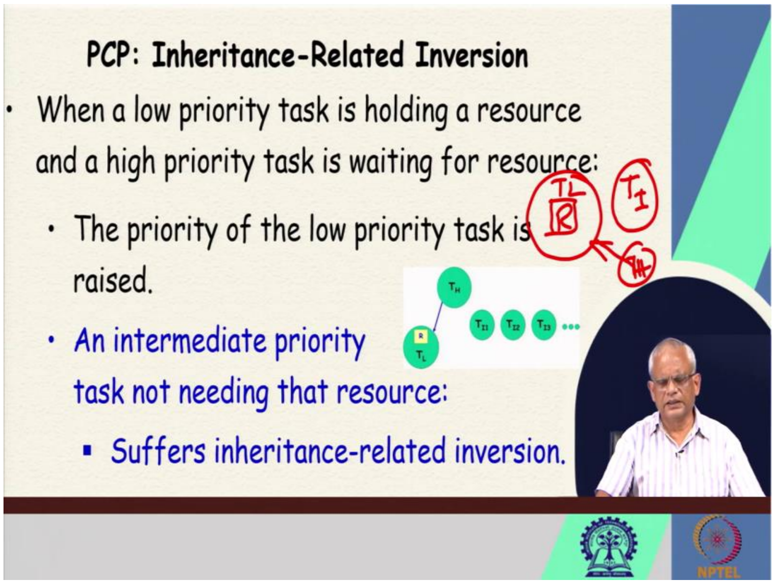
Real Time System
Professor Rajib Mall
Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur
Lecture 31
Analysis of Priority Ceiling Protocol

Welcome to this lecture. In the last lecture, we were discussing about the priority ceiling protocol. It is the protocol which is a bit sophisticated compared to the simple priority inheritance protocol and the highest locker protocol and we were trying to do some analysis of the various types of inversions that can occur.

Because if we are able to analyse what is the maximum inversion that a task can suffer under a PCP that is a priority ceiling protocol, we can use the completion time theorem to check whether the task set remains schedulable, the task set can meet their deadlines under the resource sharing with priority ceiling protocol. So, let us proceed from there, we will look at different types of inversions that can occur under priority ceiling protocol and we will try to get a upper bound on the priority inversion time for each of the tasks.

(Refer Slide Time: 01:35)

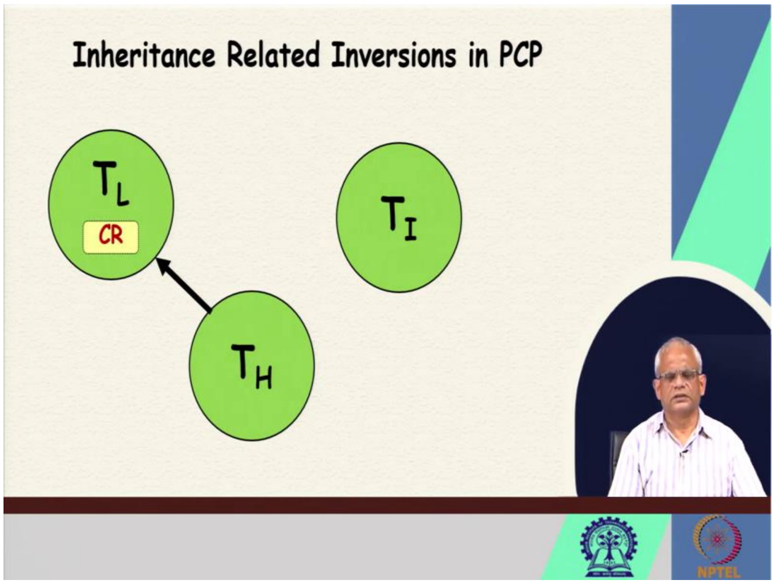


Let us, proceed from there. So, in the priority ceiling protocol we know that that besides the direct inversion, where a task high priority task is waiting for a low priority task holding resource, we can have inheritance related inversion. In the inheritance related inversion, a low priority task is holding the resource. So, there is a resource and a low priority task is holding the resource.

And a high priority task is waiting for the resource and in this situation, in the priority ceiling protocol the priority inheritance principle applies where the task T_L gets the priority of T_H, so T_L's priority becomes equal to the priority of T_H. Now, let us say T_L is having high priority starts executing, but what about a task T_I which is an intermediate priority task, it needs no resource. But simply because T_L is executing at a high priority T_I gets blocked it suffers inversion.

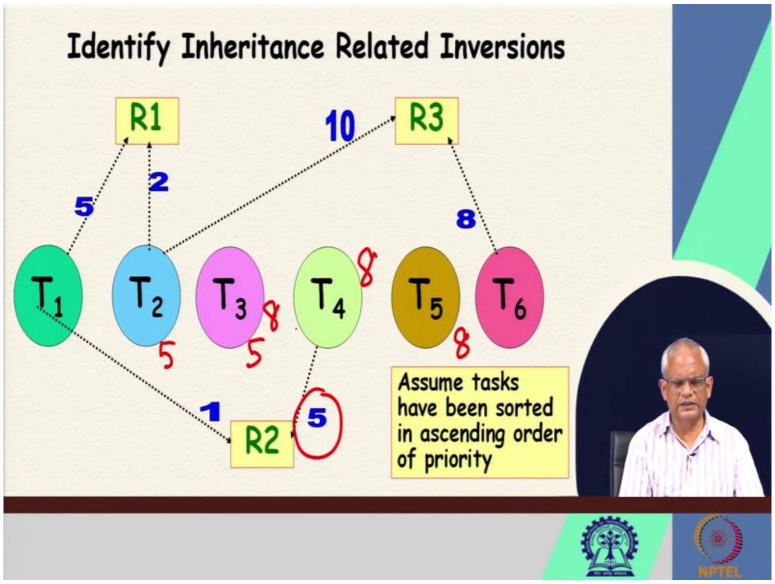
So, that we call is the inheritance related inversion suffered by T_I because T_L's priority has got increased on the other hand T_I does not need any resource. So, when low priority task T_L holding a resource high priority task waiting for it, several other tasks which have higher priority than T_L are all blocked for the duration that T_L holds the resource R and T_L's priority is equal to T_H and they suffer inversions.

(Refer Slide Time: 03:46)



So, this is an illustration of that low priority task holding the resource high priority task waiting for the resource low priority tasks priority is increased by the inheritance clause of the priority ceiling protocol and therefore the intermediate priority task T_I which do not need the resource here they suffer inversions and we call this as inheritance related inversion.

(Refer Slide Time: 04:20)



Now, let us do a small analysis here. Let us, assume that tasks have been arranged in decreasing order of their priority T_1 is the highest priority task and T_6 is the lowest priority task and the resource usage by various tasks is shown here with the dotted line and the time maximum time for which they need the resource is annotated here on the dotted line.

Now, let us find out which tasks are going to suffer from inheritance related inversion, can T_1 suffer inheritance related inversion? No, T_1 cannot suffer inheritance related inversion, because for R_2 unless T_1 is waiting for it, then T_4's priority will not increase and similar is T_2. So, if T_1 is already waiting for resource, so, that will be direct inversion, it is not inheritance related inversion. But what about T_2?

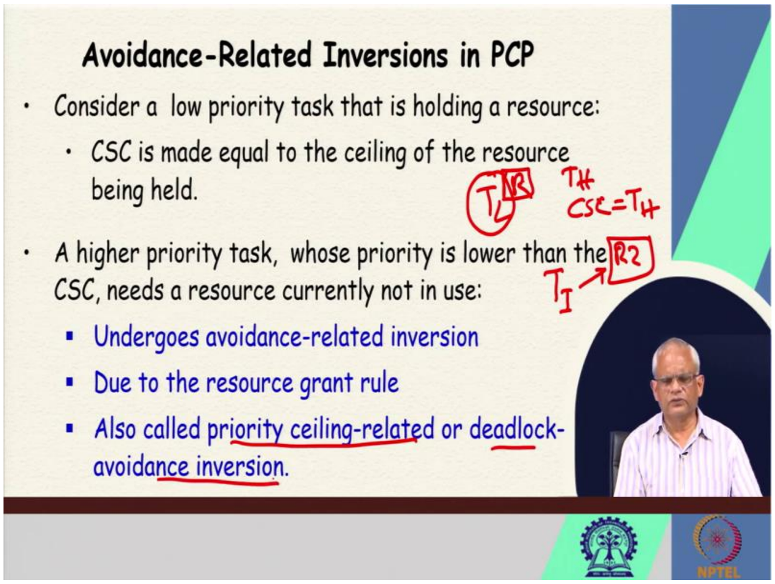
Now, T_2 can suffer inheritance related inversion, the situation is that T_4 holding R_2 and T_1 waiting for the resource and T_1 cannot proceed T_2 and T_3 cannot proceed with their execution because T_4 priority has become equal to T_1 and what is the maximum duration for which T_2 and T_3 will undergo inheritance inversion on account of R_2. So, that is 5 units.

So, both T_2 and T_3 will incur 5 units of inheritance related inversion on account of T_4 when T_1 waits for R_2. So, that is a worst case. Now, what about T_3 can it suffer any other inversion, can it suffer inheritance inversion other than due to T_4? Yes, it can suffer inheritance inversion due to T_6. When T_6 is holding the resource R_3 and T_2 is waiting for it then T_6 priority will become T_2 and T_3 cannot execute it will undergo 8 units of inversion on account of T_6.

What about T_4? Yes, T_4 can also undergo inversion account of T_6 for 8 units. What about T_5? Yes, it can also undergo inversion on account of T_6 per 8 units. But can T_5 undergo inversion on account of T_4? No, because T_4 is a higher priority task than T_5 and we do not call it a inversion. It is a normal execution that T_4 is executing T_5 is waiting that is not an inversion.

Inversion is suffered by from a lower priority task T_5 suffers inversion from T_6, T_4 suffers inversion from T_6, T_3 suffers inversion from T_4 and T_6 that is for 5 and 8 units and T_2 suffers from T_4. So, these are the inheritance related inversion here.

(Refer Slide Time: 08:20)

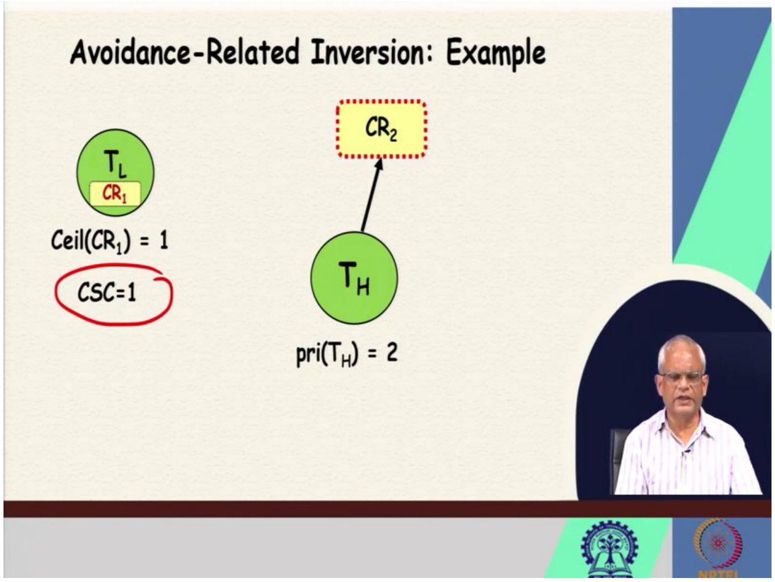


Now, let us try to understand the avoidance related inversion. In the avoidance related inversion when a low priority task holds the resource the current system ceiling is made equal to the ceiling priority of the resource. And now, the situation is that we have a low priority task which is holding a resource CR and therefore, let us say R is also used some time or other by T_H then T_L's priority becomes equal to T_H.

Now, when T_L locks R the ceiling priority becomes T_H as soon as T_L locks R the current system ceiling is set to T_H this is by the resource grant clause of the PCP and now when CSC becomes T_H then there intermediate task might want to access some resource R_2 which is not needed by T_L or T_H and then it cannot because T_I is less than CSC. So, in this case, we say that it undergoes avoidance related inversion in the clause the resource grant rule.

Even though the resource is free, it is denied. Because this is the clause that prevent deadlock avoidance. We call it as avoidance related inversion. This is also called as a priority ceiling related or deadlock avoidance related inversion.

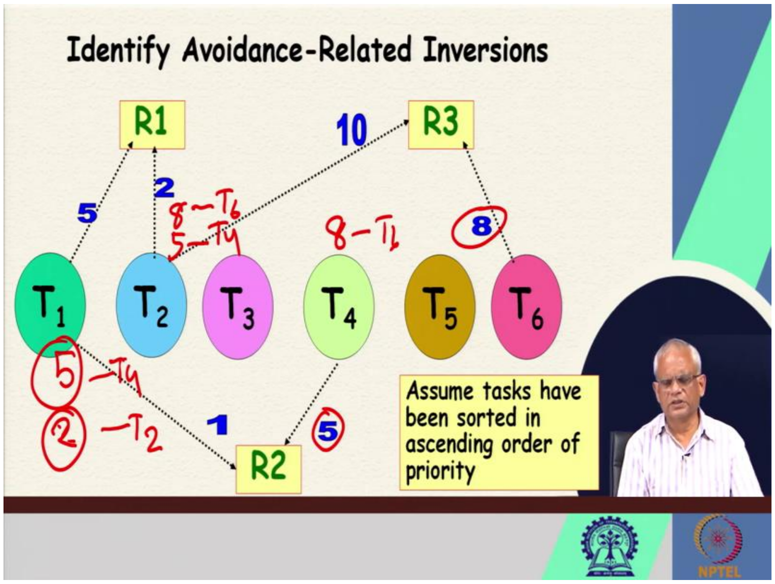
(Refer Slide Time: 10:31)



Now, let us try to do a small analysis. This example first and then we will do the analysis. So, a low priority task is holding the resource CR_1 and it so happens that the ceiling of CR_1 is very high priority is equal to 1 and as soon as T_L gets CR_1, the current system selling is set to 1 and the high priority task T_H whose priority is 2 started to execute, because T_L's priority has not yet changed, its executing a T_L only T_L maybe T_{10}.

And T_H whose priority is 2 can easily preempt T_L and starts executing, but then after some time CR_2 is needed by T_H and when T_H tries to lock CR_2 in the resource grant clause the priority of T_H is compared to the current system ceiling and it happens to be less than the current system ceiling and T_H is denied access to CR_2 and T_H blocks and this we call as the avoidance related inversion suffered by T_H.

(Refer Slide Time: 11:54)



Now, let us do a small bit of analysis and try to identify the avoidance related inversions. What about T_1 can T_1 undergo avoid insulated inversion? Yes, when T_4 is holding R_2 the current system ceiling is set to 1 that is the priority of T_1 and when T_1 tries to acquire R_1 it will not be permitted it will be denied access to R_1 and what is the maximum avoidance related inversion suffered by T_1 that is 5 unit and that is on account of T_4.

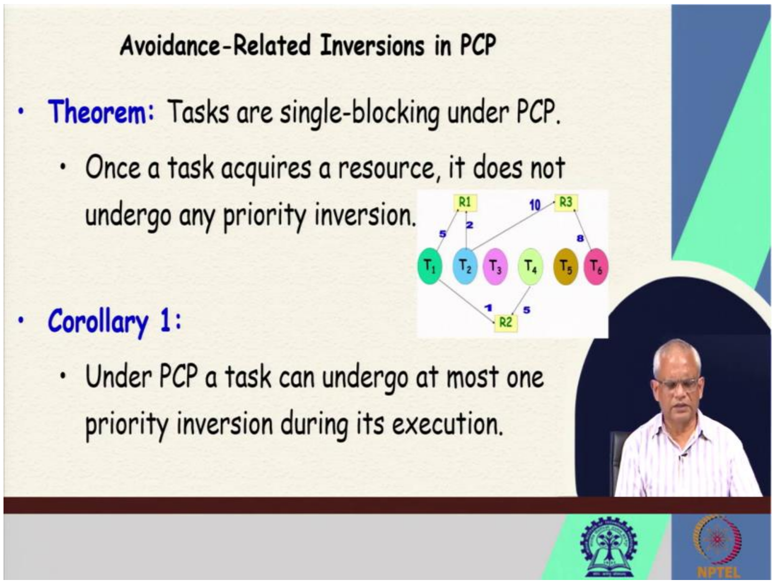
Now, similarly, when T_2 is holding R1 and T_1 accesses R_2 it will undergo 2 units of inversion avoidance related inversion on account of T_2. But what about T_2 can it undergo avoidance related inversion? Yes, when T_4 is holding R_2 T_2 while trying to lock either R_3 or R_1 will be prevented locking it. Because the ceiling the current system ceiling would have been set to 1.

So, what is the duration for which T_2 can suffer inversion avoidance related inversion due to T_4 it will be 5 units. So, T_2 will suffer 5 units of avoidance related inversion due to T_4, can T_2 suffer avoidance related inversion due to T_6? Yes, when T_6 is holding R_3 T_2 is trying to access R_1 it will undergo inheritance related inversion, sorry, avoidance related inversion for 8 units.

So, on account of T_6 it will undergo 8 units of avoidance related inversion. But what about T_3? T_3 will not undergo any avoidance related inversion because T_3 does not need any resource. What about T_4? Yes, T_4 will undergo avoidance related inversion due to T_6 when T_4 requests R_2 it will denied, because T_6 is holding R_3 and then it will undergo 8 units of inversion on account of T_6. What about T_5?

No it cannot undergo avoidance related inversion because it does not need any resource. What about T_6? T_6 is the lowest priority task; it does not suffer any inversion any type of inversion T_6 does not incur.

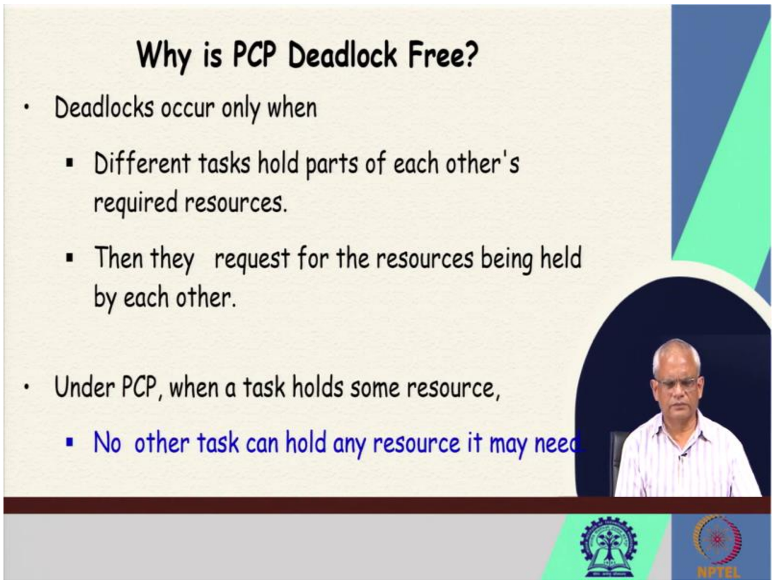
(Refer Slide Time: 15:35)



Now, let us look at one simple theorem related to avoidance related inversion in the ceiling protocol the tasks are single blocking under the ceiling protocol. So, here because of the avoidance clause here, the current system ceiling is set to the priority ceiling priority of the resource and therefore once a task acquires a resource all other resources that it needs must be free, because of the avoidance clause that is current system ceiling is compared with the priority of the task needing a resource.

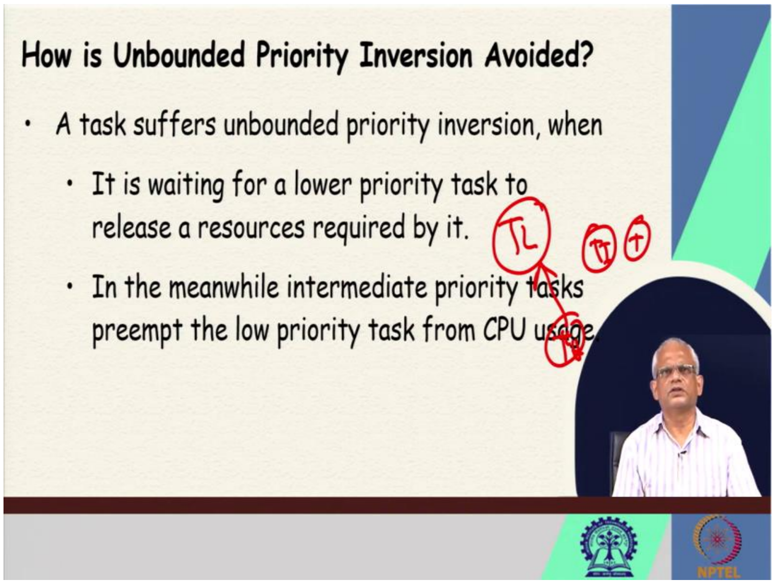
And the corollary is that under the priority ceiling protocol a task can undergo at most 1 priority inversion. So, it can block only once because once it blocks and gets one resource, it will be all other resources will be available and also the current system ceiling is set to high value and therefore, other tasks cannot acquire any resource and therefore, it suffers at most one priority inversion.

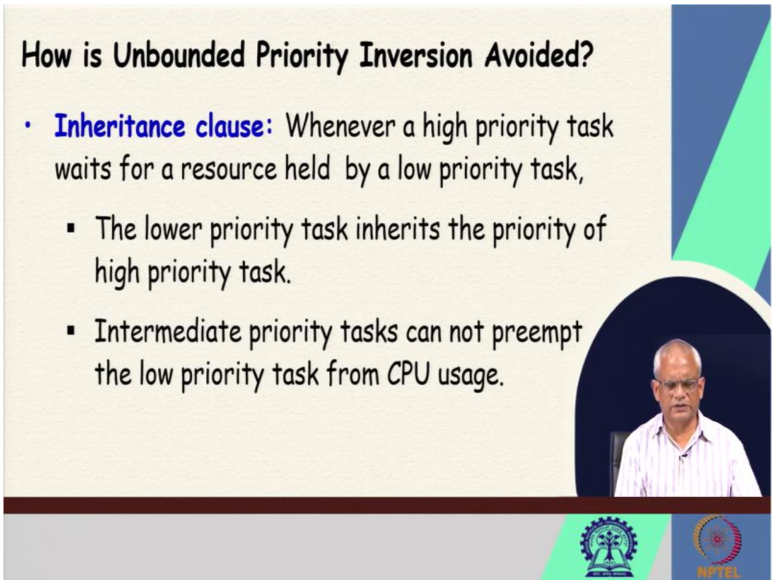
(Refer Slide Time: 17:12)



Now, why is the priority ceiling protocol deadlock free that is very simple to see here that any task once it gets one resource all other resources are free. It can get them anytime and there is no question of a deadlock all the requirements of a task can be met under the priority ceiling protocol once it gets one resource all other resources are free, it can lock them anytime.

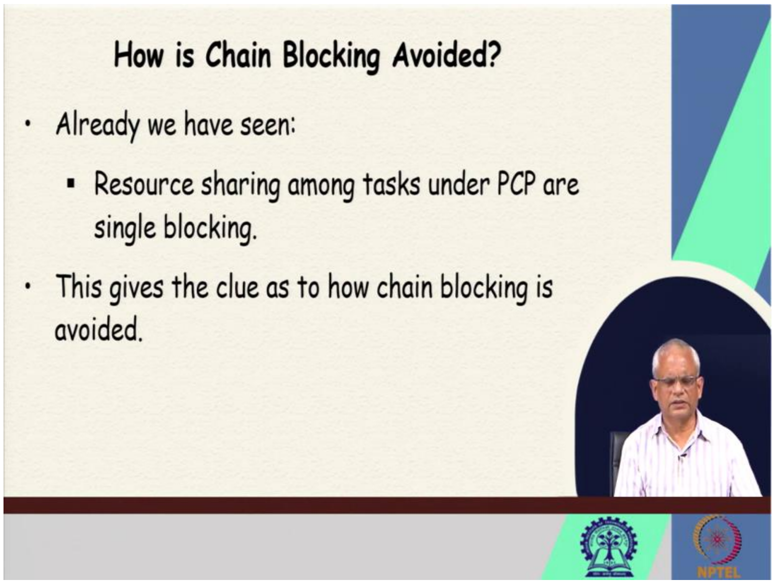
(Refer Slide Time: 17:48)





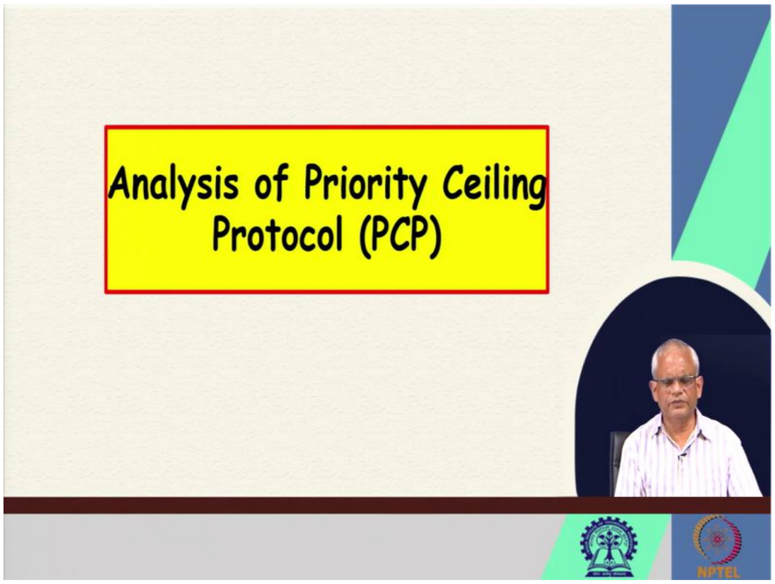
Now, how about the unbounded priority inversion? How is it avoided in the priority ceiling protocol? In the priority ceiling protocol and normally an unbounded priority inversion occurs when a task is holding a resource a high priority task is waiting for it and the intermediate priority tasks they keep on executing and preempting T_L. But here due to the inheritance clause unbounded priority inversion is avoided because T_L's priority is raised to that of the T_H. So, the inheritance clause prevents any unbounded priority inversion.

(Refer Slide Time: 18:45)



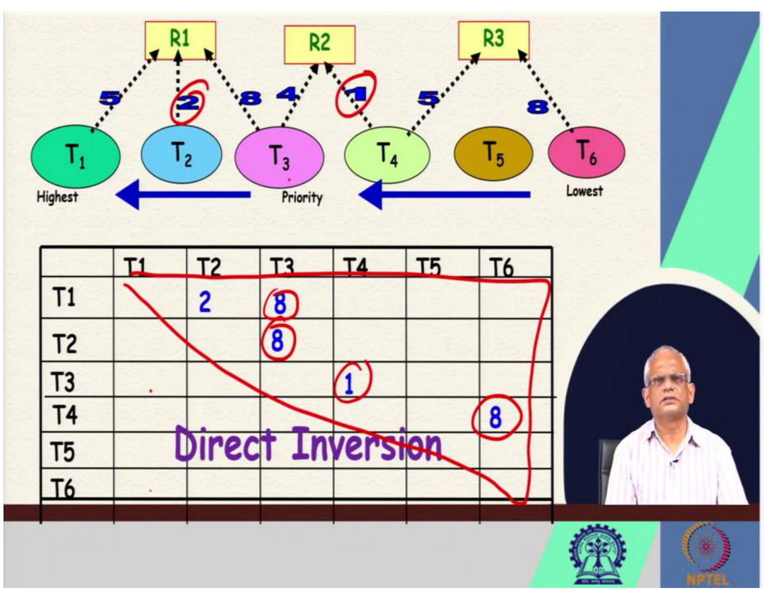
And how is the chain blocking avoided? Chain blocking is avoided because once a task gets any one resource all other resources must be free and therefore, a task is single blocking and there cannot be any chain blocking.

(Refer Slide Time: 19:07)



Now, let us do a small analysis of the priority ceiling protocol to determine. what is the maximum priority inversion time per task?

(Refer Slide Time: 19:22)



Let us, consider this task graph. Here again, T_1 is the highest priority, the tasks have been sorted and T_6 is the lowest priority and the resource requirement are represented here. The resources are rectangles, R1, R2, R3 and the dotted lines are the access to the resource and the maximum time for which a task access as a resource is annotated here.

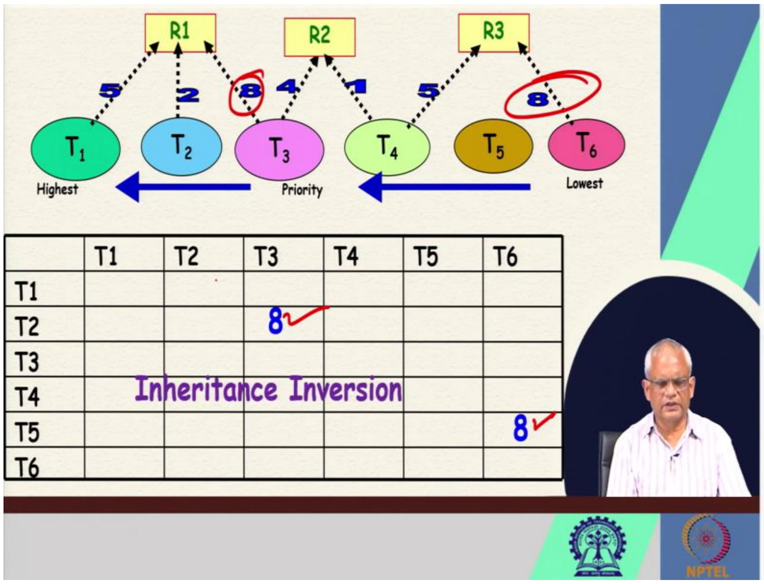
Now, let us do first direct inversion. Now, does T_1 suffer any direct inversion? Yes, it can suffer direct inversion due to T_2 and T_3 because T_2 might have locked R1 and when T_1 requires it will not get it. So, the direct inversion here, what is the maximum duration for on account of T_2 is 2 units here. On account of T_3 it is 8 units here.

What about T_2 does it incur any direct inversion? Yes, it can incur direct inversion on account T_3 and that is for 8 units can T_3 incur any direct inversion it cannot incur direct inversion, it cannot suffer direct inversion on account of T_1 T_2. Because they are higher priority tasks, task can suffer inversion only on account of a lower priority task.

So, T_3 can incur inversion on T_4 for 1 unit, what about T_4? T_4 can incur inversion on account of T_6 for 8 units, T_5 does not need any resource it cannot incur any direct inversion and T_6 does not suffer any inversion at all because it is the lowest priority task and see that the inversion is an upper triangular matrix, even for the other cases, the inheritance and the avoidance inversion we will see that it is an upper triangular matrix.

So, what is the reason behind this, that it is upper triangular matrix? The reason is that a task does not suffer inversion on account of a lower priority task T_3 cannot suffer inversion on account of T_1 because T_1 is a high priority task. A task suffers inversion only on account of lower priority task it does not suffer inversion from higher priority task and that is why the lower triangular part of the matrix is empty. It is the upper triangular matrix.

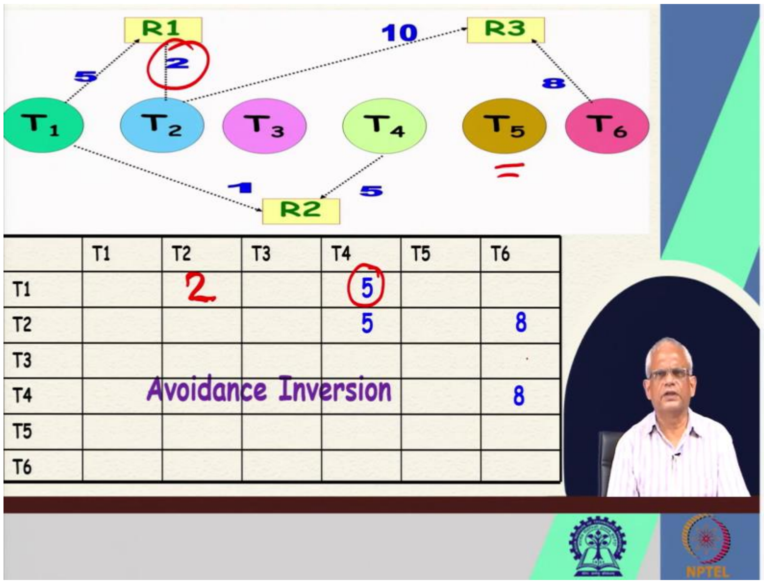
(Refer Slide Time: 22:34)



Now let us try to analyse the inheritance related inversion. So, here can T_1 have any inheritance related inversion? No, because it is the highest priority task and the other tasks will block, will inherit priority only T_1 blocks further. What about T_2? T_2 can suffer inheritance related inversion an account of T_3 when T_3 is holding R1 and T_1 is waiting for R1 the priority of T_3 becomes 1 and then T_2 cannot execute.

So, what is the maximum duration for which T_2 can suffer inheritance inversion is 8 units. Now, what about T_3? T_3 does not suffer any inheritance related inversion. What about T_4? T_4 also does not incur any inheritance related inversion. What about T_5? T_5 can suffer inheritance related inversion when T_6 is holding R3 and T_4 is waiting for it and it can suffer 8 units of inversion on account of T_6. So, only 2 tasks can suffer inheritance related inversion.

(Refer Slide Time: 24:16)

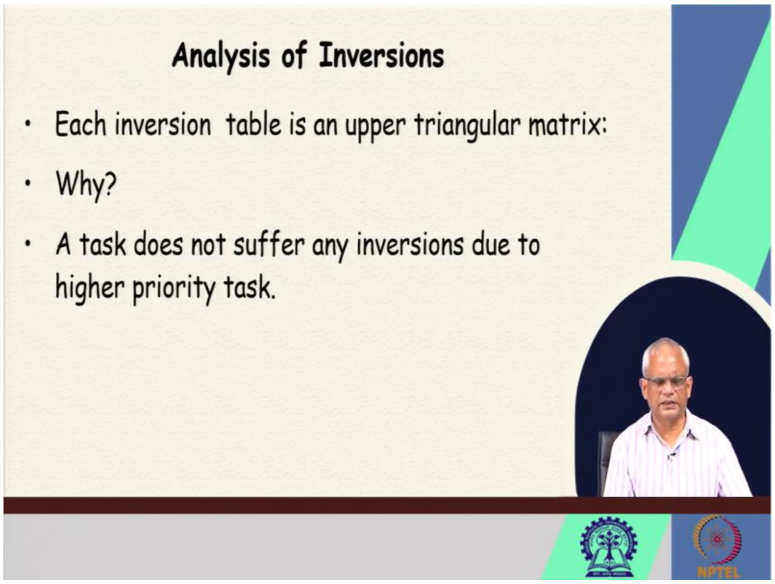


What about avoidance related inversion? This is the task graph here. Now, can T_1 suffer avoidance related inversion? Yes, T_1 can suffer avoidance related inversion on account of T_4, when T_4 is well holding R2 T_1 cannot get R1. And what is the maximum duration? 5, can T_1 undergo avoidance related inversion on account of T_2?

Yes, when T_2 is holding R1 T_1 cannot lock R2 and what is the maximum duration that it can undergo avoidance related inversion on account of T_2 is 2 units. Now, what about T_2? T_2 can undergo avoidance inversion on account of T_4. When T_4 is holding R2 T_2 cannot lock R1 So, T_2 can undergo inversion for 5 units and it can also undergo 8 units of inversion on account of T_6 because when T_6 is holding R3 T_2 cannot lock R1.

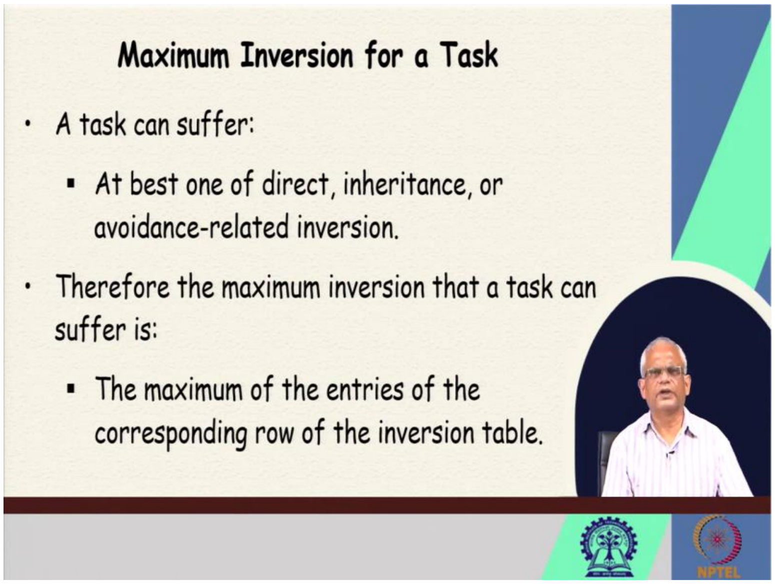
And what about T_3? T_3 does not undergo any avoidance related inversion because it does not need any resource. What about T_4? Yes, T_4 can undergo inversion on account of T_6 for 8 units and that is it, T_5 does not need any resource it will not undergo an inversion. Similarly, T_6 is the lowest priority task it does not undergo any inversion and again it is a upper triangular matrix.

(Refer Slide Time: 26:32)



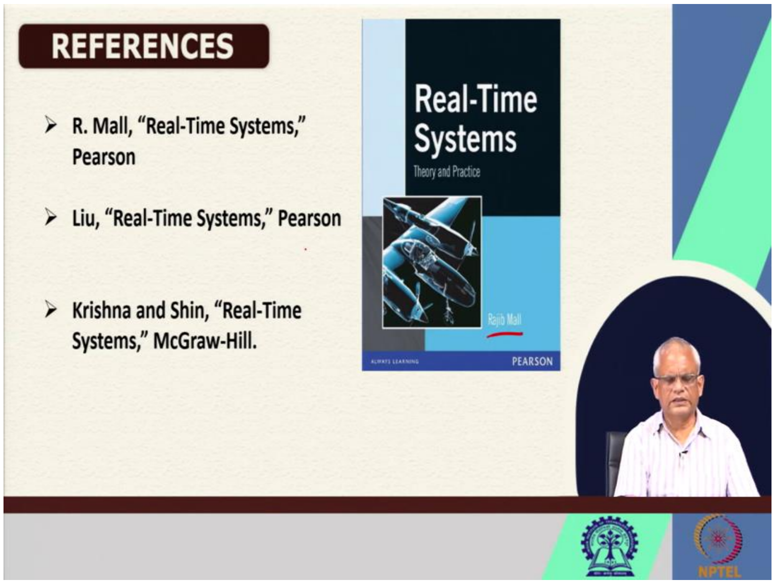
So, it is upper triangular matrix because the task does not suffer any inversion due to higher priority tasks.

(Refer Slide Time: 26:42)



What is the maximum inversion for a task? It can undergo at best one inversion either due to direct inheritance or avoidance and therefore, to compute the maximum inversion time we must check for a specific task the rows in all the 3 tables that is the direct inversion table inheritance inversion table and avoidance inversion table and pick the maximum entry there and that is the maximum time for which a task can undergo inversion.

(Refer Slide Time: 27:22)



So, we have completed our discussion on the worst case inversion being suffered by a task on account of the priority ceiling protocol we can even do analysis on similar lines using the priority inheritance protocol and the highest locker protocol. So, these discussions are there in the real time systems book authored by me and also on Liu and Krishna and Shin. So, we are at the end of the lecture. Thank you.