

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)

PCP: Inheritance-Related Inversion

- When a low priority task is holding a resource and a high priority task is waiting for resource:
 - The priority of the low priority task is raised.
 - An intermediate priority task not needing that resource:
 - Suffers inheritance-related inversion.

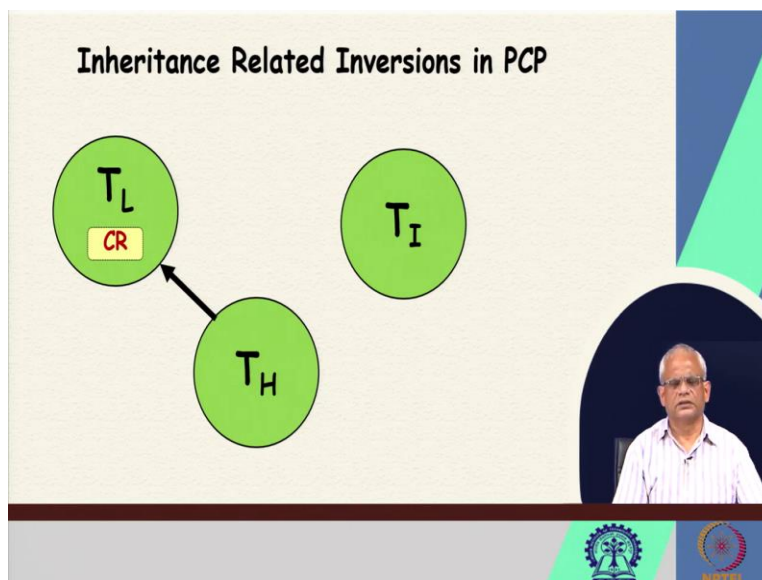
Let us, proceed from there. So, in the priority ceiling protocol we know that that besides the direct inversion, where a task high priority task is waiting for a low priority task holding resource, we can have inheritance related inversion. In the inheritance related inversion, a low

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.

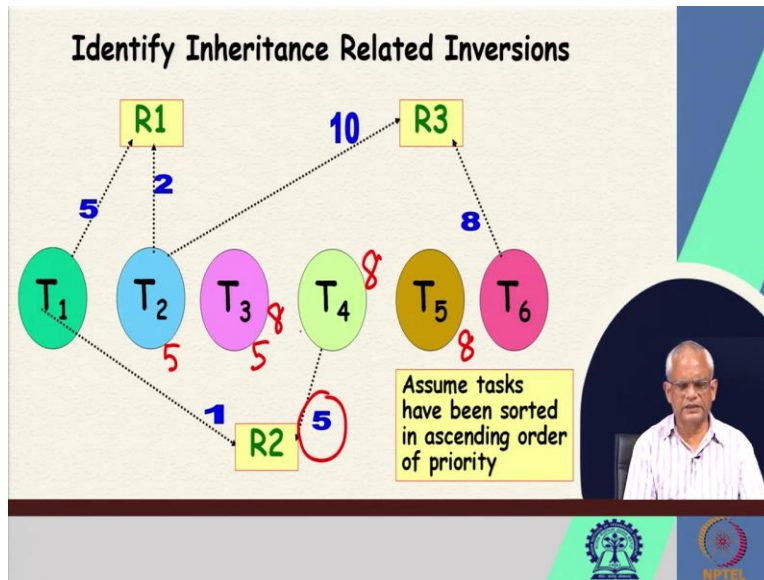
So, that we call is the inheritance related inversion suffered by T_I because T_L 's priority has got increased on the other hand T_I does not need any resource. So, when low priority task T_L holding a resource high priority task waiting for it, several other tasks which have higher priority than T_L 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 T₁ is the highest priority task and T₆ 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 T₁ suffer inheritance related inversion? No, T₁ cannot suffer inheritance related inversion, because for R₂ unless T₁ is waiting for it, then T₄'s priority will not increase and similar is T₂. So, if T₁ is already waiting for resource, so, that will be direct inversion, it is not inheritance related inversion. But what about T₂?

Now, T₂ can suffer inheritance related inversion, the situation is that T₄ holding R₂ and T₁ waiting for the resource and T₁ cannot proceed T₂ and T₃ cannot proceed with their execution because T₄ priority has become equal to T₁ and what is the maximum duration for which T₂ and T₃ will undergo inheritance inversion on account of R₂. So, that is 5 units.

So, both T₂ and T₃ will incur 5 units of inheritance related inversion on account of T₄ when T₁ waits for R₂. So, that is a worst case. Now, what about T₃ can it suffer any other inversion, can it suffer inheritance inversion other than due to T₄? Yes, it can suffer inheritance inversion due to T₆. When T₆ is holding the resource R₃ and T₂ is waiting for it then T₆ priority will become T₂ and T₃ cannot execute it will undergo 8 units of inversion on account of T₆.

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)

Avoidance-Related Inversions in PCP

- Consider a low priority task that is holding a resource:
 - CSC is made equal to the ceiling of the resource being held.
- A higher priority task, whose priority is lower than the CSC, needs a resource currently not in use:
 - Undergoes avoidance-related inversion
 - Due to the resource grant rule
 - Also called priority ceiling-related or deadlock-avoidance inversion.

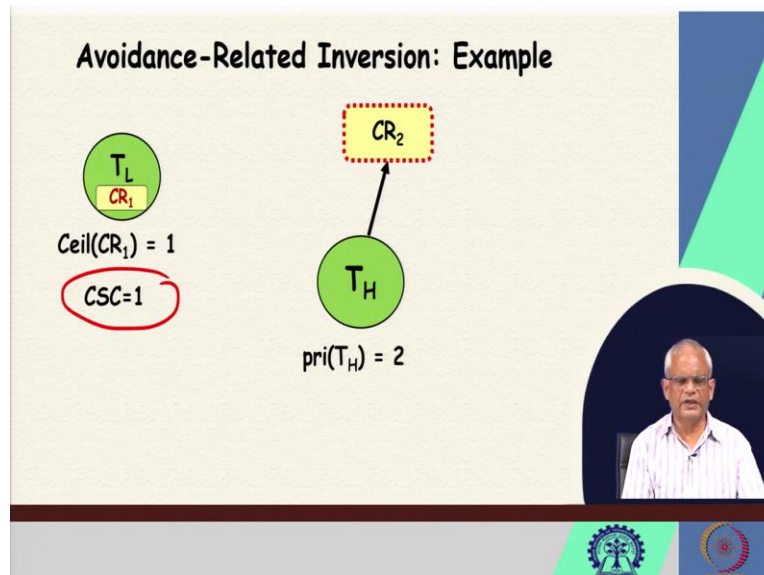
Handwritten annotations: T_L holding R , T_H with $CSC = T_H$, and T_I needing R_2 .

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.

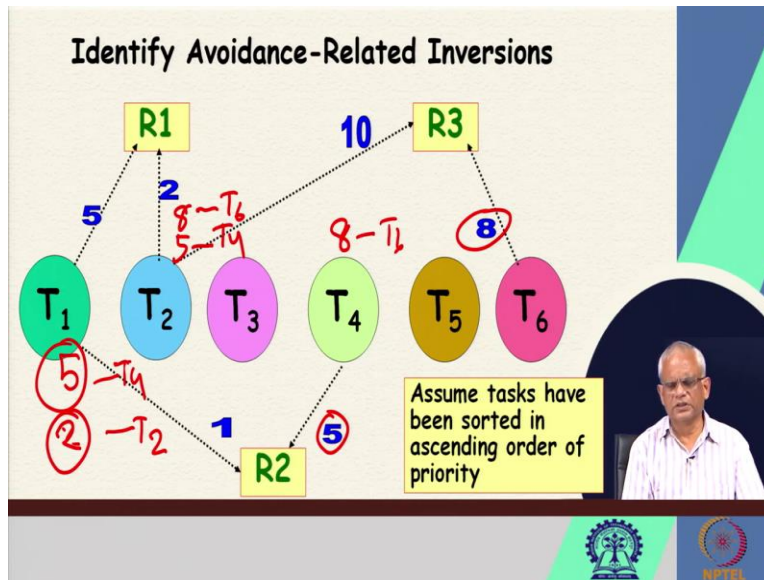
(Refer Slide Time: 10:31)



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 ceiling 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 T₁ can T₁ undergo avoid insulated inversion? Yes, when T₄ is holding R₂ the current system ceiling is set to 1 that is the priority of T₁ and when T₁ tries to acquire R₁ it will not be permitted it will be denied access to R₁ and what is the maximum avoidance related inversion suffered by T₁ that is 5 unit and that is on account of T₄.

Now, similarly, when T₂ is holding R₁ and T₁ accesses R₂ it will undergo 2 units of inversion avoidance related inversion on account of T₂. But what about T₂ can it undergo avoidance related inversion? Yes, when T₄ is holding R₂ T₂ while trying to lock either R₃ or R₁ 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 T₂ can suffer inversion avoidance related inversion due to T₄ it will be 5 units. So, T₂ will suffer 5 units of avoidance related inversion due to T₄, can T₂ suffer avoidance related inversion due to T₆? Yes, when T₆ is holding R₃ T₂ is trying to access R₁ it will undergo inheritance related inversion, sorry, avoidance related inversion for 8 units.

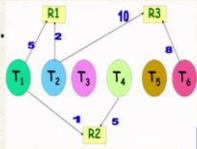
So, on account of T₆ it will undergo 8 units of avoidance related inversion. But what about T₃? T₃ will not undergo any avoidance related inversion because T₃ does not need any resource. What about T₄? Yes, T₄ will undergo avoidance related inversion due to T₆ when T₄ requests R₂ it will denied, because T₆ is holding R₃ and then it will undergo 8 units of inversion on account of T₆. What about T₅?

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)

Avoidance-Related Inversions in PCP


- **Theorem:** Tasks are single-blocking under PCP.
 - Once a task acquires a resource, it does not undergo any priority inversion.
- **Corollary 1:**
 - Under PCP a task can undergo at most one priority inversion during its execution.



The diagram illustrates a task-resource dependency graph. It shows six tasks, T_1 through T_6 , represented by colored circles. T_1 is green, T_2 is blue, T_3 is purple, T_4 is yellow, T_5 is orange, and T_6 is red. Three resources, R_1 , R_2 , and R_3 , are shown as yellow rectangles. R_1 has a priority ceiling of 5 and is held by T_1 . R_2 has a priority ceiling of 5 and is held by T_4 . R_3 has a priority ceiling of 10 and is held by T_5 . Arrows indicate dependencies: T_2 depends on R_1 , T_3 depends on R_1 , T_4 depends on R_2 , and T_6 depends on R_3 . The priority of each task is indicated by a number: T_1 (5), T_2 (2), T_3 (3), T_4 (4), T_5 (6), and T_6 (1).



A small video frame in the bottom right corner of the slide shows a man with glasses and a light-colored shirt, likely the presenter.



At the bottom of the slide, there are two logos: the logo of Anna University on the left and the logo of NITRR on the right.

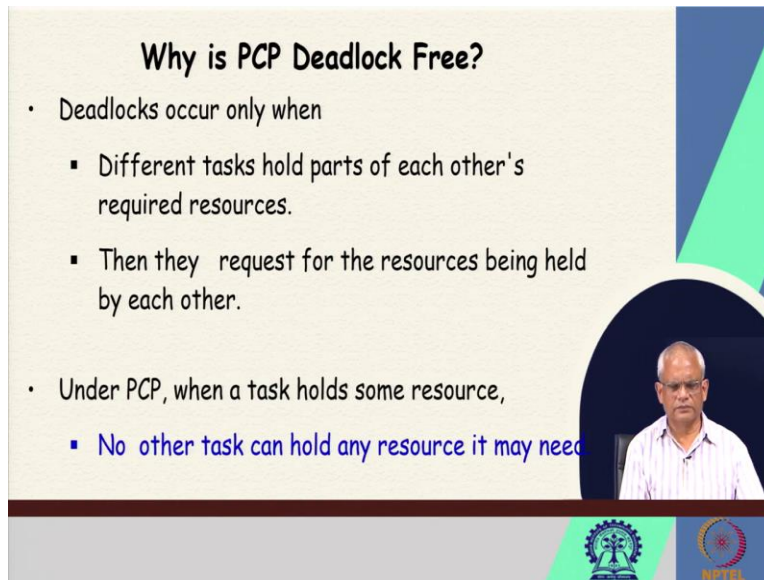
Now, let us look at one simple theorem related to avoidance related inversion in the ceiling protocol the tasks are single blocking under the ceiling protocol. So, here because of the avoidance clause here, the current system ceiling is set to the priority ceiling priority of the resource and therefore 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)

Why is PCP Deadlock Free?

- Deadlocks occur only when
 - Different tasks hold parts of each other's required resources.
 - Then they request for the resources being held by each other.
- Under PCP, when a task holds some resource,
 - No other task can hold any resource it may need.

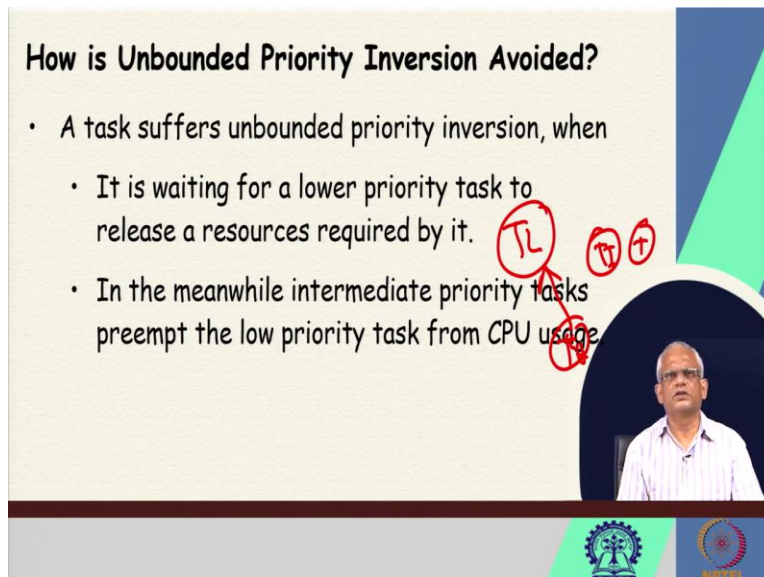


Now, why is the priority ceiling protocol deadlock free that is very simple to see here that any task once it gets one resource all other resources are free. It can get them anytime and there is no question of a deadlock all the requirements of a task can be met under the priority ceiling protocol once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

(Refer Slide Time: 17:48)

How is Unbounded Priority Inversion Avoided?

- A task suffers unbounded priority inversion, when
 - It is waiting for a lower priority task to release a resources required by it.
 - In the meanwhile intermediate priority tasks preempt the low priority task from CPU usage.



How is Unbounded Priority Inversion Avoided?

- **Inheritance clause:** Whenever a high priority task waits for a resource held by a low priority task,
 - The lower priority task inherits the priority of high priority task.
 - Intermediate priority tasks can not preempt the low priority task from CPU usage.



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)

How is Chain Blocking Avoided?

- Already we have seen:
 - Resource sharing among tasks under PCP are single blocking.
- This gives the clue as to how chain blocking is avoided.



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)

Analysis of Priority Ceiling Protocol (PCP)

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)

Priority
Highest ← → Lowest

	T1	T2	T3	T4	T5	T6
T1		2	8			
T2			8			
T3				1		
T4						8
T5						
T6						

Direct Inversion

Let us, consider this task graph. Here again, T₁ is the highest priority, the tasks have been sorted and T₆ 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.

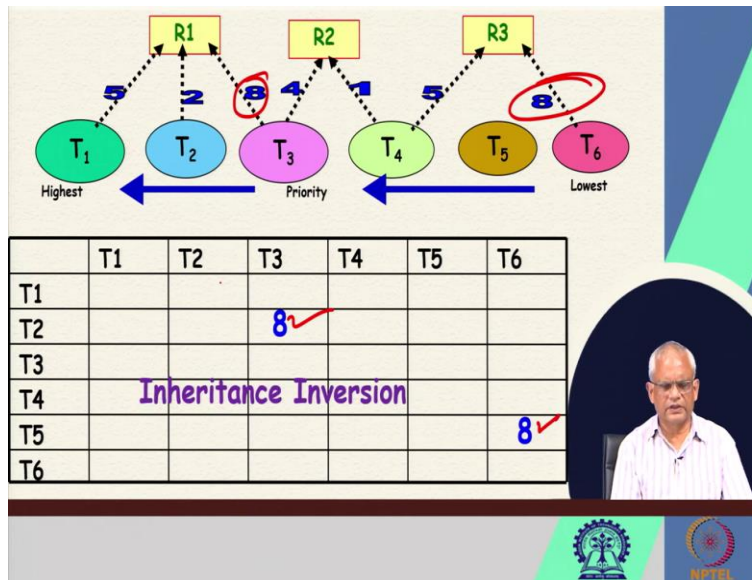
Now, let us do first direct inversion. Now, does T₁ suffer any direct inversion? Yes, it can suffer direct inversion due to T₂ and T₃ because T₂ might have locked R1 and when T₁ requires it will not get it. So, the direct inversion here, what is the maximum duration for on account of T₂ is 2 units here. On account of T₃ it is 8 units here.

What about T₂ does it incur any direct inversion? Yes, it can incur direct inversion on account T₃ and that is for 8 units can T₃ incur any direct inversion it cannot incur direct inversion, it cannot suffer direct inversion on account of T₁ T₂. Because they are higher priority tasks, task can suffer inversion only on account of a lower priority task.

So, T₃ can incur inversion on T₄ for 1 unit, what about T₄? T₄ can incur inversion on account of T₆ for 8 units, T₅ does not need any resource it cannot incur any direct inversion and T₆ 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 T₃ cannot suffer inversion on account of T₁ because T₁ 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.

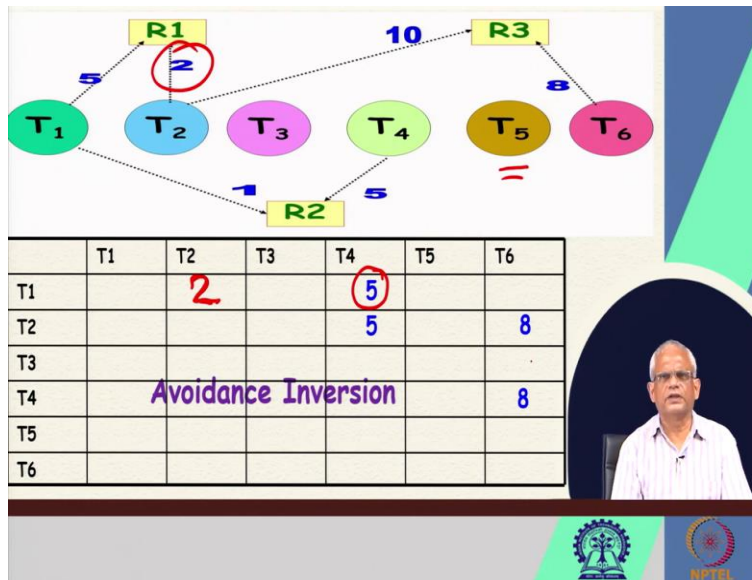
(Refer Slide Time: 22:34)



Now let us try to analyse the inheritance related inversion. So, here can T₁ have any inheritance related inversion? No, because it is the highest priority task and the other tasks will block, will inherit priority only T₁ blocks further. What about T₂? T₂ can suffer inheritance related inversion an account of T₃ when T₃ is holding R₁ and T₁ is waiting for R₁ the priority of T₃ becomes 1 and then T₂ cannot execute.

So, what is the maximum duration for which T₂ can suffer inheritance inversion is 8 units. Now, what about T₃? T₃ does not suffer any inheritance related inversion. What about T₄? T₄ also does not incur any inheritance related inversion. What about T₅? T₅ can suffer inheritance related inversion when T₆ is holding R₃ and T₄ is waiting for it and it can suffer 8 units of inversion on account of T₆. So, only 2 tasks can suffer inheritance related inversion.

(Refer Slide Time: 24:16)

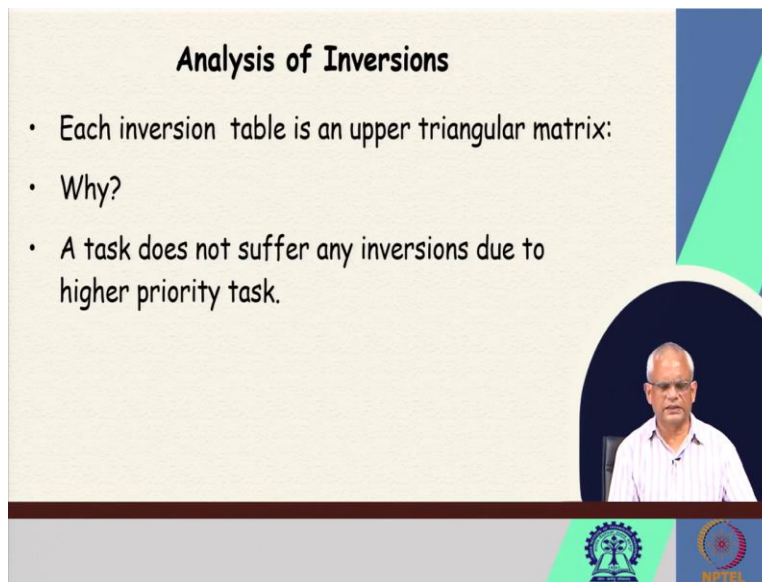


What about avoidance related inversion? This is the task graph here. Now, can T_1 suffer avoidance related inversion? Yes, T_1 can suffer avoidance related inversion on account of T_4 , when T_4 is well holding R_2 T_1 cannot get R_1 . And what is the maximum duration? 5, can T_1 undergo avoidance related inversion on account of T_2 ?

Yes, when T_2 is holding R_1 T_1 cannot lock R_2 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 R_2 T_2 cannot lock R_1 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 R_3 T_2 cannot lock R_1 .

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)



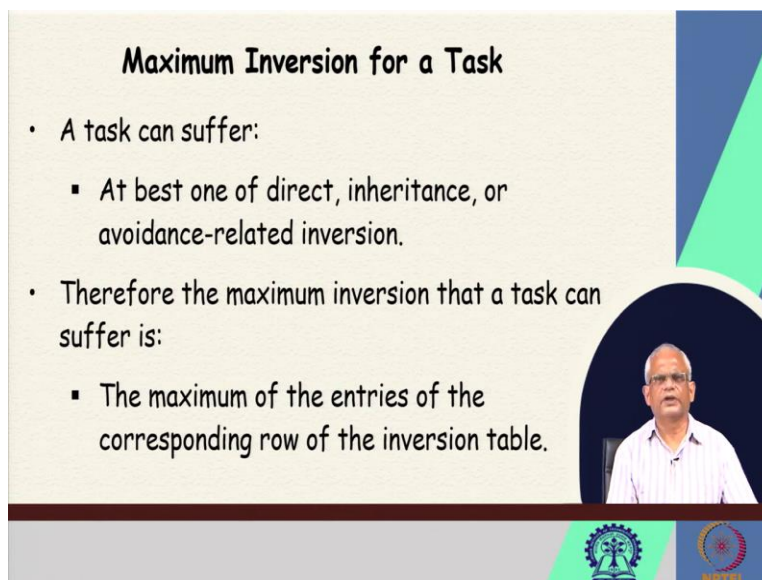
Analysis of Inversions

- Each inversion table is an upper triangular matrix:
- Why?
- A task does not suffer any inversions due to higher priority task.

The slide features a video inset of a speaker in the bottom right corner. The background is light beige with a blue and green geometric design on the right side. Logos for IIT Bombay and NPTEL are visible at the bottom.

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

(Refer Slide Time: 26:42)



Maximum Inversion for a Task

- A task can suffer:
 - At best one of direct, inheritance, or avoidance-related inversion.
- Therefore the maximum inversion that a task can suffer is:
 - The maximum of the entries of the corresponding row of the inversion table.

The slide features a video inset of a speaker in the bottom right corner. The background is light beige with a blue and green geometric design on the right side. Logos for IIT Bombay and NPTEL are visible at the bottom.

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)

REFERENCES

- R. Mall, "Real-Time Systems," Pearson
- Liu, "Real-Time Systems," Pearson
- Krishna and Shin, "Real-Time Systems," McGraw-Hill.

The slide features a video inset of a man in a white shirt speaking. To the right of the references is the cover of the book "Real-Time Systems: Theory and Practice" by R. Mall, published by Pearson. The cover includes the text "ALWAYS LEARNING" and "PEARSON". At the bottom of the slide, there are logos for IIT Delhi and NPTEL.

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.