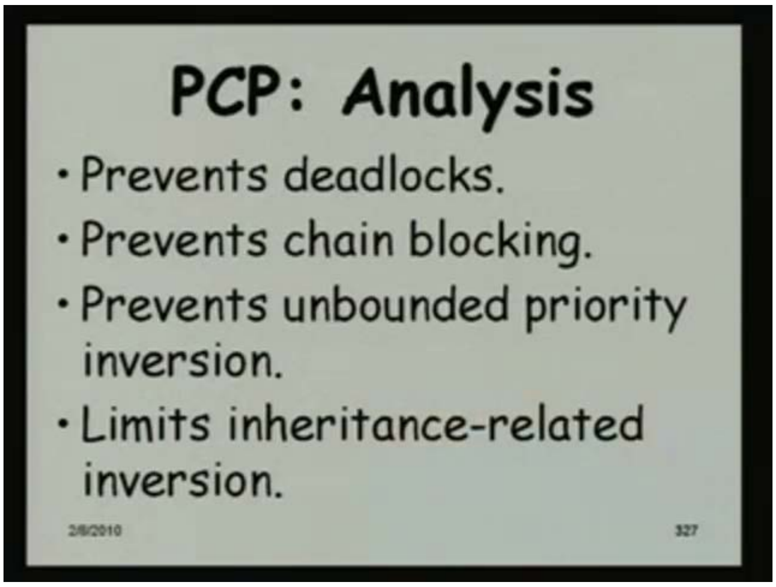
Real-Time Systems

Prof. Dr. Rajib Mall
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture No. # 15
An Analysis of Priority Ceiling Protocol

So, let us get started, we have looked at the highest locker protocol - the the priority ceiling protocol - the basic ideas of priority ceiling protocol, the resource request and the resource release clauses. Now, let us look at an analysis, let us try to analyze the priority ceiling protocol; its advantages, disadvantages, some examples, and before that, let us, we will look at example of how the priority ceiling protocol works.

(Refer Slide Time: 00:56)



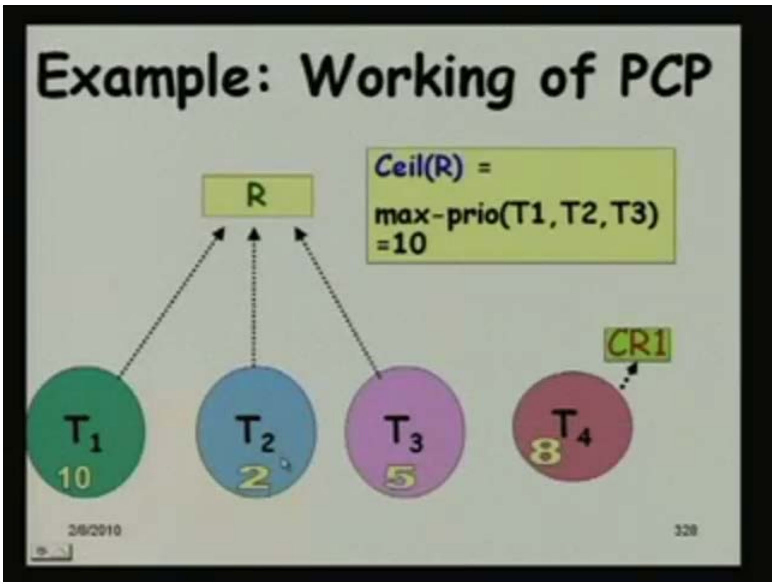
PCP: Analysis

- Prevents deadlocks.
- Prevents chain blocking.
- Prevents unbounded priority inversion.
- Limits inheritance-related inversion.

2/6/2010 327

So, we will also try to establish these results, how deadlock is prevented here, how chain blocking is prevented, how unbounded priority inversion is prevented and how inheritance-related inversion is limited.

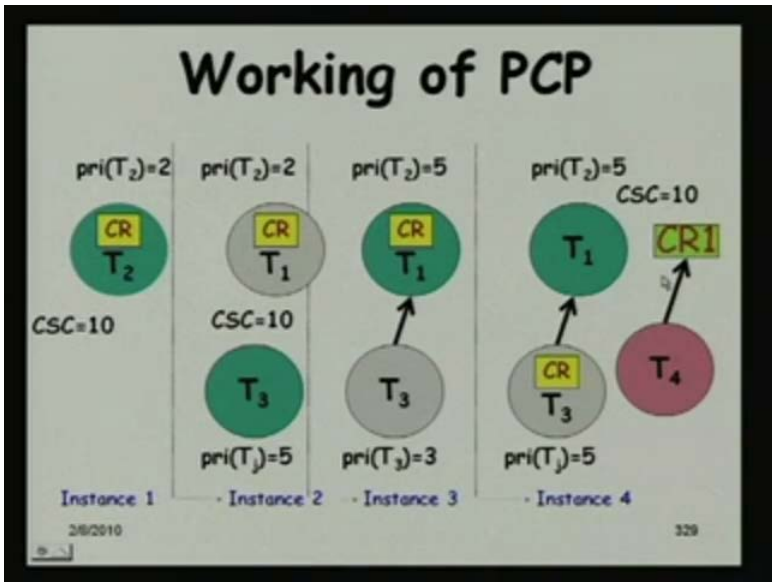
(Refer Slide Time: 01:11)



So, first let us look at an example of working of the priority ceiling protocol. Let us take this resource R and CR1, and let us say R is requested by T 1, T 2, T 3 - R is used by T 1, T 2, T3, and T 1's priority is 10, T 2 is 2 and T 3 is 5, and let us assume that the higher priority value indicates higher priorities, which is a window convention.

Now, the ceiling priority of R will become 10 in that case, is it not? So, that is higher priority - highest priority among these three resources. Now, let us assume that a task T 4 needs a resource CR1 and its priority is 8. Of course, you should consider only shared resources right and assume that CR1 is also shared by some other tasks. I have not shown that here, because of lack of space here. Let us assume that CR1 is a shared critical resource. See, if it is exclusively needed by T 4, then we do not have to consider that at all.

(Refer Slide Time: 02:35)

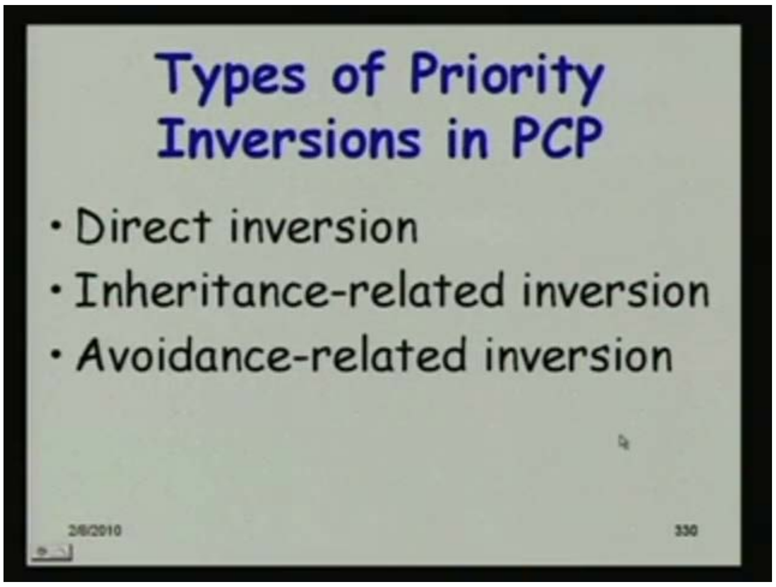


Working of PCP

Now, let us say T 2 acquires the CR, T 2's priority is 2, after acquiring the resource, a current system ceiling will be set to the ceiling of CR, which is 10, we have seen. And then, let us say, the task T j, see here, only the ceiling priority changed. T 1 continues to have its priority as 2. Now, let us say T 3 started executing, because it is has a higher priority 5 and after sometime it needed CR. Let us say priority of T 3 is 3; I think it should have been 5 or something. So 5, then by the inheritance clause T 2 will become priority of T 2 will sorry T 1 will become 5. I am sorry, if this is T 2 this would also have been T 2 right. So, there is a mistake here.

So, now let us say, the task T 4 starts executing, and then, requests the resource CR1, and then, it will be checked against CSC and because its priority is less than the CSC T 4 will block. And then T 1's priority will increase to that of the T 4's priority, because T 4 is blocking on another resource. So, these will not be granted, request to CR1 will be rejected.

(Refer Slide Time: 04:33)



Types of Priority Inversions in PCP

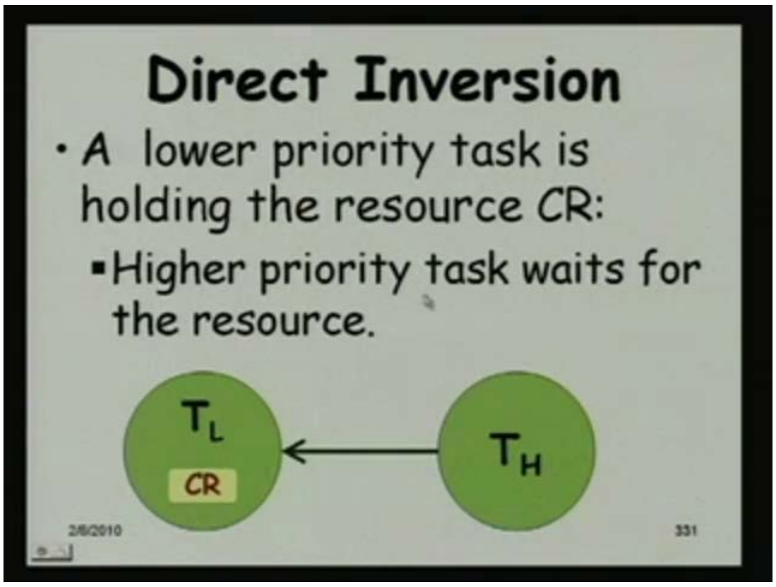
- Direct inversion
- Inheritance-related inversion
- Avoidance-related inversion

2/8/2010 330

Now, let us try to understand the different types of priority inversions that can occur under this priority ceiling protocol. Actually, there are three types of inversions that can occur; one is the direct inversion, then we also have inheritance-related inversion, this we had seen in case of highest locker protocol, and we had said that this minimizes inheritance-related inversion, but does not eliminate it all together. And then, we will also have an avoidance-related inversion.

So, first let us see, let us try to understand what are these three types of inversion, and then, we will do some analysis based on how much a task will undergo inversion due to all its resource requirements.

(Refer Slide Time: 05:27)

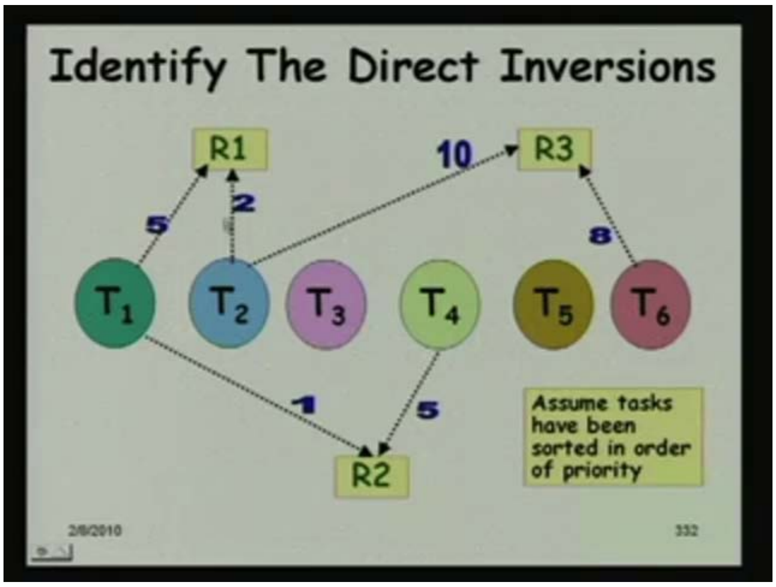


Direct Inversion

- A lower priority task is holding the resource CR:
  - Higher priority task waits for the resource.

First, let us look at the direct inversion, very simple, simplest of all all these three inversions. So, here, a lower priority task holding a shared resource and a higher priority task is waiting for the resource. So directly, the lower priority task is causing there a inversion - priority inversion - for the higher priority task. This is a direct inversion. I hope, I hope that there is no difficulty here.

(Refer Slide Time: 06:04)



Identify The Direct Inversions

Assume tasks have been sorted in order of priority

Just an example here, task scenarios; we have six tasks and there are three resources R1, R2, R3, and then, these are the requests for different resources that can occur. T 3 does not need any resource at all. T 5 does not need any resource. And assume that the tasks have been sorted in order of their priority. So, T 1 is the highest priority. We will not talk of priority 10 etcetera, because the convention, differing conventions; we will just assume that T 1 has higher priority than T 2, T 2 has higher priority than T 3, and so on. Of course, in the initial task set, the priorities were not ordered, we can just rename the tasks, so that finally, you have got these task set T 1 to T 6 ordered in terms of their priorities.

Now, these are the resource requirements. Now, what is the maximum or what is the direct inversions that T 1 will suffer from? What is the direct priority inversions that T 1 can suffer from?

The execution of…

What are the inversions due to which task it will suffer an inversion?

T 2 and R3.

Yes. T 1 will suffer an inversion - priority inversion - due to T 2 on account of sharing resource R1, because it might so happen that T 2 was holding the resource, and T 1 got blocked directly. Similar is the case with resource R2 required by it. It could have been held by R4 T 4. So, what is the maximum duration for which T 2 can block T 1 due to direct inversions? Yes, these are the millisecond values written here. 10 milliseconds - Do you think that T 1 will get for blocked for 5 milliseconds?

2 milliseconds.

2 milliseconds, because T 2 can hold at best for 2, is it not? After 2, it will release it. So, the maximum inversion to T 1 caused by T 2, maximum direct inversions is 2 milliseconds.

And what about…

And also its 1.

What about R4?

What is the sorry R2? What is the maximum inversion due to its use of R2 ?

5.

5.

T 4 can cause a direct priority inversion for up to 5 milliseconds.

What about T 2?

What are the direct inversions it will suffer?

Will it suffer a direct inversion due to T 1?

No.

No, because that is not a priority inversion. T 1 is already a higher priority than T 2. So, due to its resource uses R1, T 2 will not suffer any direct priority inversion.

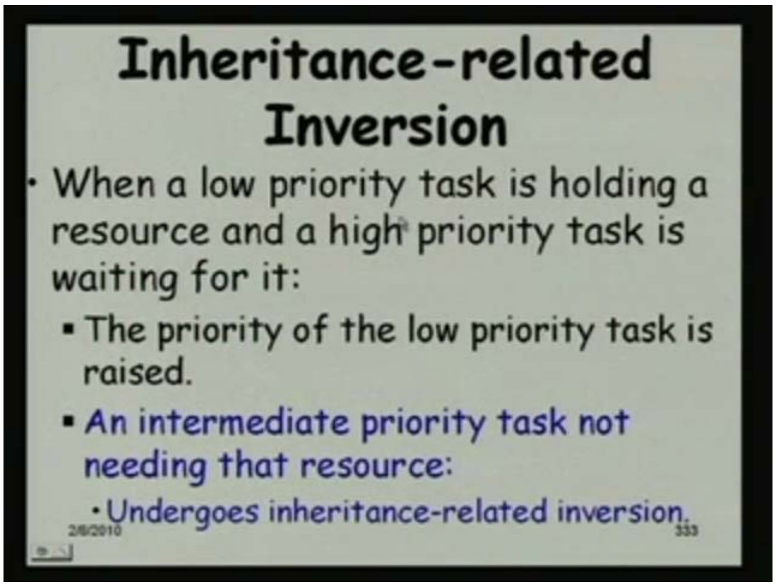
And, it will suffer priority inversion due to R3 and what is the duration?

8.

8.

So, it can suffer a priority inversion due to T 6, the lowest priority task here, for up to 8 milliseconds. What about T 3, will it suffer any direct priority inversion? No, it will not suffer any direct inversion.

(Refer Slide Time: 09:36)



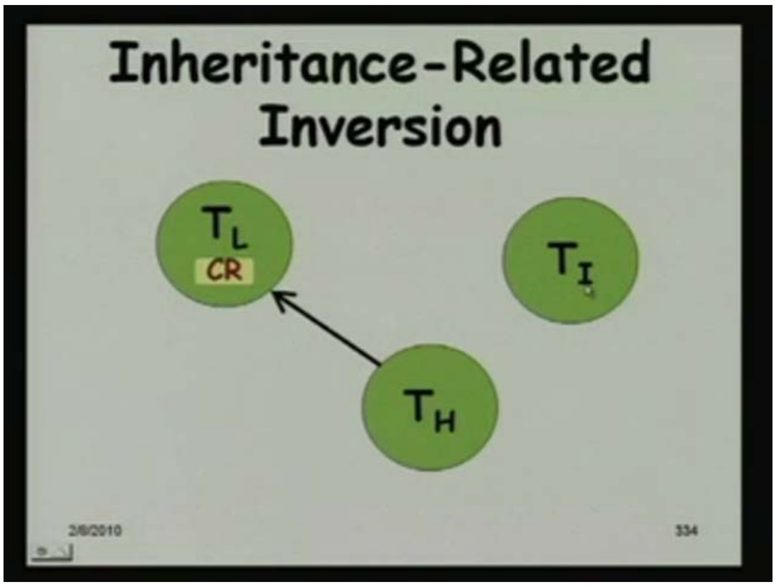
Inheritance-related Inversion

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

2/8/2010 333

Now, let us look at the inheritance-related inversion. When a low priority task is holding a resource, and a high priority task is waiting for some resource, then the priority of the low priority task is raised by the inheritance clause. Now, because the low priority task is executing at a higher priority, an intermediate priority task, not needing any resource will undergo inheritance-related inversion. Because see, it should have continued, because a low priority task is executing, and this would have got preference over that. It is not able to execute, just because that task, priority has raised.

(Refer Slide Time: 10:27)

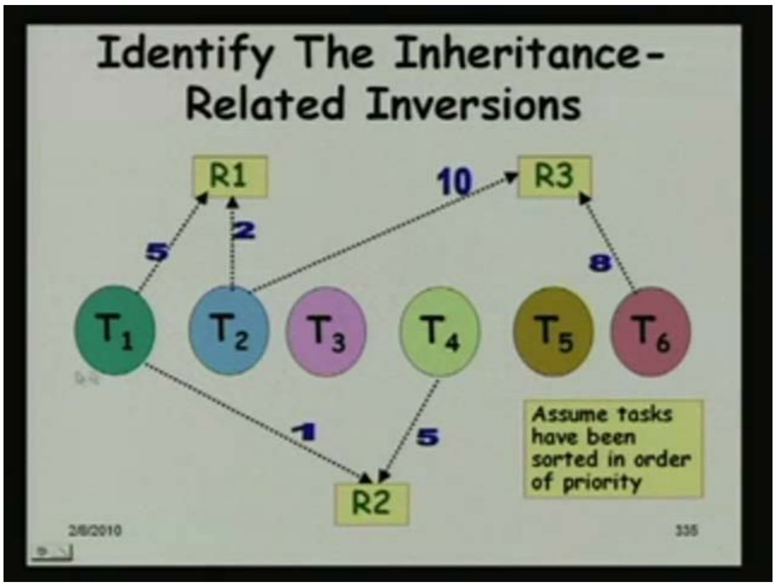


Inheritance-Related Inversion

2/8/2010 334

So, let us look at an example. A low priority task was holding a critical resource and a high priority task is blocked directly. And by the inheritance clause, this will acquire the priority of T H - the higher priority task - and therefore, T I would not be able to execute. It is not needing any resource, but it is prevented from being executed. So that we will call as the inheritance-related inversion.

(Refer Slide Time: 11:08)


Identify The Inheritance-Related Inversions
R1
R3
R2
5
2
10
8
1
5
T1
T2
T3
T4
T5
T6
Assume tasks have been sorted in order of priority
2/8/2010
335

Now, let us take the same example, tasks are sorted according to priority, these are the resource uses. So, what is the inheritance-related inversion for T 1? Will T 1 suffer any inheritance-related inversion? No, T 1 will not suffer any inversion is it not? Inheritance-related inversion? Because unless the inheritance clause is applied, and if the inheritance clause is applies to other tasks, then still it will be less than T 1, is it not? So, T 1 does not suffer an inheritance-related inversion. What about T 2? Will it suffer an inheritance-related inversion?

Yes.

Yes, due to what, due to which task?

T… sir T 4.

Yes.

It will undergo an inheritance-related inversion due to T 4, assume that T 4 is holding R2, and T 1 is waiting for resource R2, then T 4, the inheritance clause will be applied, and its priority will become that equal to T 1. And T 2 will undergo an inheritance inversion. What about T 3, will it undergo inheritance-related inversion?

Yes, yes.

Due to which task?

T 4.

T 4 and T 6, sir.

Yes. It will undergo inheritance-related inversion due to both T 4 and T 6, because T 4 priority might be raised to T 1 due to the resource, and similarly, T 6 priority could have been raised to that of T 2, because of R3. So, T 3 will undergo inheritance-related inversions due to both T 4 and T 6. Similarly, you can find for T 4, is it not? what is the…

T 6.

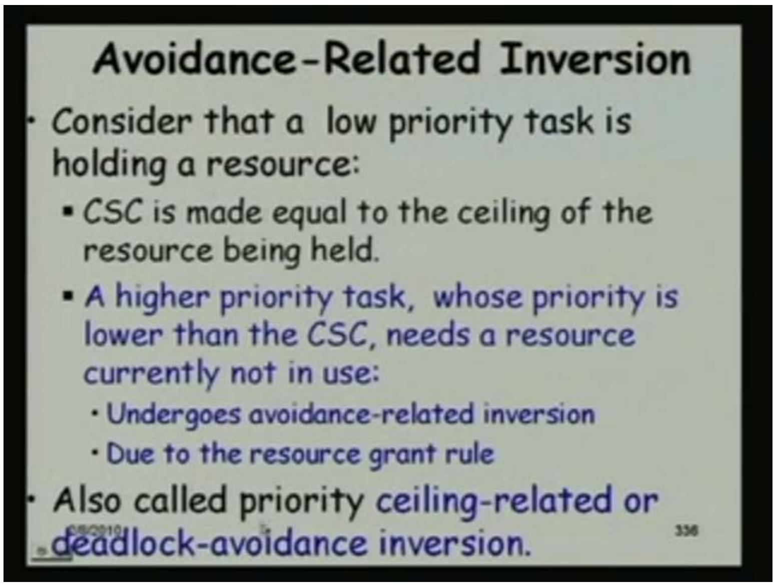
T 6, yes exactly, what about T 5 due to T 6?

Yes, what about T 6?

(Audio not clear. Refer Time: 13:23)

No, it is the lowest priority task, it cannot undergo any inversion.

(Refer Slide Time: 13:28)



Avoidance-Related Inversion

- Consider that a low priority task 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.

6/8/2010 336

Now, let us look at the avoidance-related inversion. Consider that a low priority task is holding a resource, and CSC is made equal to the ceiling of the resources being held, and now, a higher priority task, whose priority is lower than the CSC, of course, it is larger than the task which is holding the resource; it needs a resource currently not in use. It should have been granted, is it not? Because it is requesting a resource which is not being used by the low priority task. And it has to wait, because of the request, resource request clause, where the CSC value is checked against the task priority, and it will be denied the resource. And then, we will say that the task that was denied the use of resource, even though it was available. It is an avoidance-related inversion; occurs due to the resource grant rule. It is also called as priority ceiling-related or the deadlock-avoidance related inversion.

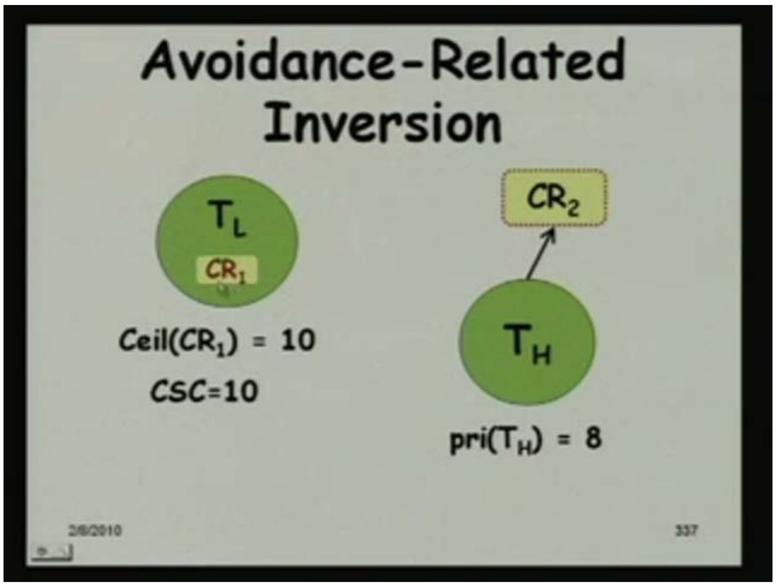
Different terms used, but why is it called avoidance-related inversion?

Deadlock.

Yes.

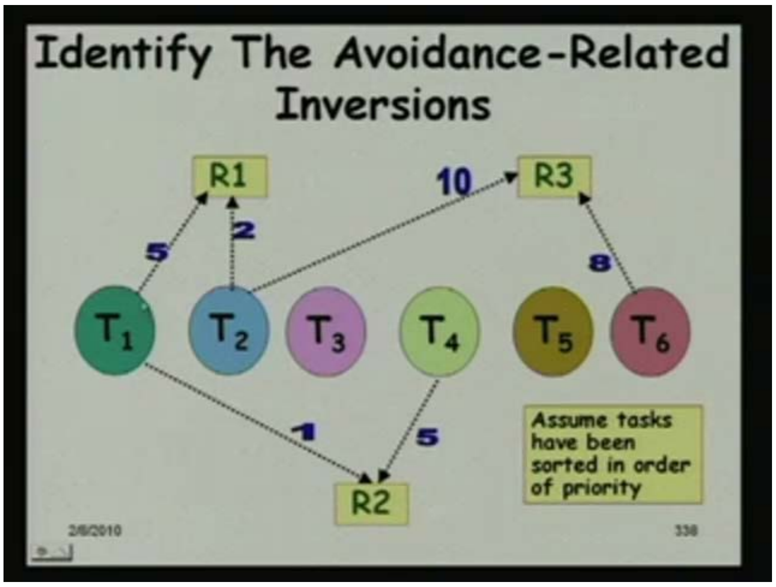
Because, this clause, sorry this situation occurred, because the task, which was requested a resource not being used, was not allocated, just to prevent deadlocks. So, this is called as the deadlock-avoidance inversion.

(Refer Slide Time: 15:14)


Avoidance-Related Inversion

This is an example of that. A low priority task holding resource CR 1, and let us say, the ceiling of CR 1 is 10, and immediately after the low priority task acquired its ceiling, the current system ceiling is set to the ceiling of the resource. Now, let us say another task, whose priority is 8 started executing, because it is priority was more than T L, started executing, but after sometime it requested CR 2. Now, when it requested CR 2, its priority 8 will be checked against 10. Now, assuming that 10 is a higher priority than 10, T H will be prevented from accessing CR 2. And T H will undergo an avoidance-related inversion.

(Refer Slide Time: 16:18)


Identify The Avoidance-Related Inversions

So, what about this example? Will T 1 undergo any avoidance-related inversion? So, due to which it will undergo avoidance?

T 1 T 2 T 3 T 2 is accessing R1 and if T 1 has requested R2 to prevent deadlock.

(Audio not clear. Refer Time: 16:52)

Right.

Yes. See, what he saying is the situation is that - T 2 is holding R1, and the current system ceiling will be set to that equal to T 1, and if T 1 requests R2 first, see if it requests R1, then this is a direct inversion. But if it requests R2 first, then it will not be granted, just because the current system ceiling is already equal to the priority of T 1. Right? So, T 1 will undergo an avoidance inversion due to T 2 holding R1. Is that ok? Now, what about T 2, will it undergo avoidance inversion?

Yes.

So, due to which task will it undergo an avoidance inversion?

So, it holds R3, and if T 2 equals R1 by 2.

Exactly.

So, R1 was available, and T 6 is holding R3. So, T 1 will be prevented from getting R1 even though it was available, and it will undergo avoidance inversion due to T 6, for what duration?

8 milli…

8 millisecond yes, because this can at best hold for 8 milliseconds, but is there any other avoidance inversion for T 2.

T 4 holds the R2

Yes. So, it will also suffer an avoidance inversion due to T 4, because T 4 gets R2, the current system ceiling is set equal to that of T 1. T 2 will be prevented, and it will follow an avoidance inversion. But what about T 3? will it follow any Will it have any avoidance-related inversion?

No.

Yes. It will not have any avoidance-related inversion, because it does not require any resource at all. See, avoidance relation, the inversion occurs when there is a resource request, and already the CSC value is high. What about T 4, will it undergo an avoidance inversion?

T 6, T 2 sir.

Yes, due to what?

T 6.

T 6.

It holds R3 and…

Yes. T 6 is holding R3, the CSC value has been set, and it cannot access the resource that it needs, even though it is available, and it will undergo an avoidance inversion, for what duration?

8.

8.

Maximum 8 millisecond it can suffer avoidance inversion.

Yes. Yes right.

(Conversation not audible. Refer Time: 19:56)

We will need to release, when T 3 if it is accessing T 6, and it will accessing R 3. So, the ceiling value R3 is 2.

T 6 and… T 6 is accessing R3 ceiling value is set to 2 that of T 2.

So like and ceiling value for R2, so, the R2 is 1.

Yes.

So, sir T 4, therefore, cannot undergo avoidance related inversion.

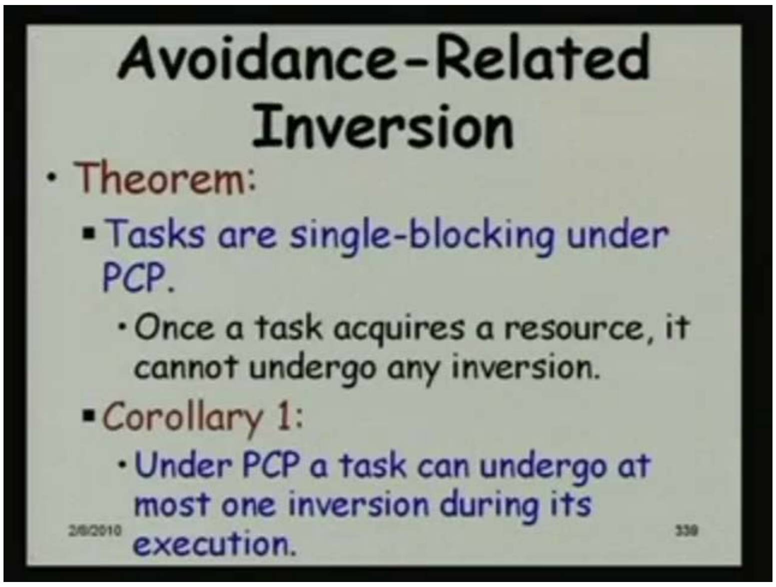
Because the ceiling of R2 is greater than R3

No, no, no.

See, the way it works is that, once T 6 gets R3, the current system ceiling is set to T 2. Now, if T 4 requests R2, T 4's priority is checked against the current system ceiling. So, current system ceiling is that T 2, and T 4's priority is less than T 2, and therefore, it will be prevented from accessing R2. Is that ok? Clear for everybody?

So, the in a resource grant clause, the priority of a task does not change after getting a resource immediately, and the only thing that changes is the current system ceiling, and when there is a request made, the priority of the task is checked against the current system ceiling. Now, let us look at a theorem.

(Refer Slide Time: 21:25)


Avoidance-Related Inversion

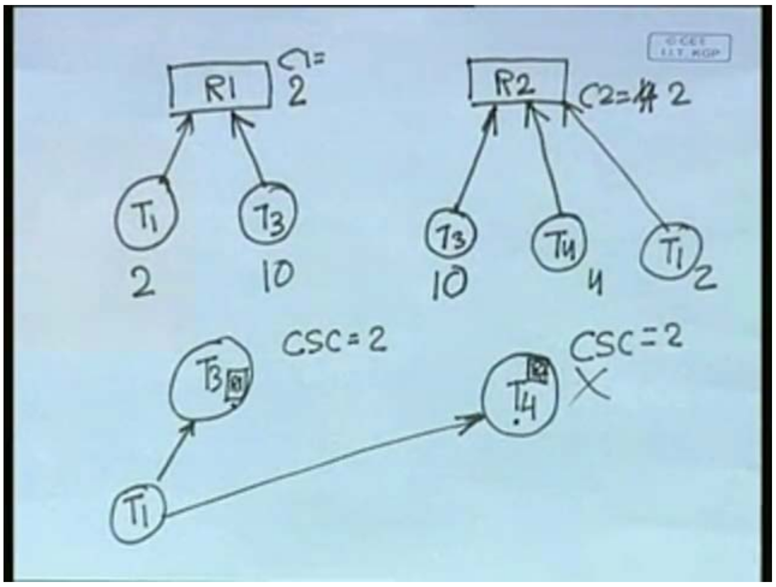
- Theorem:
  - Tasks are single-blocking under PCP.
    - Once a task acquires a resource, it cannot undergo any inversion.
  - Corollary 1:
    - Under PCP a task can undergo at most one inversion during its execution.

2/8/2010 339

The tasks are single-blocking under priority ceiling protocol. Once a task acquires a resource, it cannot undergo any inversion. Just similar to the highest locker protocol, we will just give a scheme of the proof here, simple ideas, and which can be formalized. And, the corollary will be that under priority ceiling protocol, a task can undergo at most one inversion, see even though we talked of so many inversions, inheritance-related inversion, direct inversion, avoidance inversion, but at best one task can undergo any one of those inversions. It converts at the same time; first suffer an inheritance inversion and then, an avoidance inversion, and then, a direct inversion not possible.

So this of course, this corollary will follow directly from here, because it is single blocking. So, once it has blocked under some inversion, it will not block further. So that is the intuitive argument here. But first, let us look at the theorem, that tasks are single blocking, even though it requires multiple resources it will block only once. Now, again like previous one. Let us take an example, and then, assume that in the more than one blocking will occur, and then, so that such a situation will not exist.

(Refer Slide Time: 23:08)



So, let us assume that task, we have a resource R1, required by a task T 1, and also required by a task T 3. Let us assume T 1's priority is 2 and T 3's priority is let us say 10. And let us assume that 2 is, priority 2 is more than 10, and let us assume R2, a resource again required by T 3, priority of 10, and let us say we have another task T 4 whose priority is 4. So, the ceiling value of R1 will be 2, is it not? Assuming that 2 is a higher priority than 10, and for R 2… So, C1 is equal to 2, and for R2, C2 is equal to 4 right.

Now, let us assume that T 3… I think T 3, let us show it for T 1 and then we need to show T 1 also here, T 1. So, the priority of the ceiling priority will become 2, T 1's priority is 2 right, the ceiling C2 is equal to 2, the highest of these 3 priorities right.

Now, let us assume that T 1 first blocked for R 1, which was being held by T 3, and then it blocked for R2, which was being held by T 4. Let us consider that scenario and we will show that scenario will not exist. So, T 3 was first holding R1 and T 3's priority does not change; only the CSC should have been set to 2 right.

Now, T1 and let us assume that T 3, sorry T what were saying, T 4, yes. So, T 4 started executing, and it acquired the resource R2, and CSC will be set to 2. And then, T 1 first blocked for T 3 - direct inversion - and then after some time, T 3 released the resource R1, and then it found that t R 2 is already with r T 4, and then it again blocked for T 4.

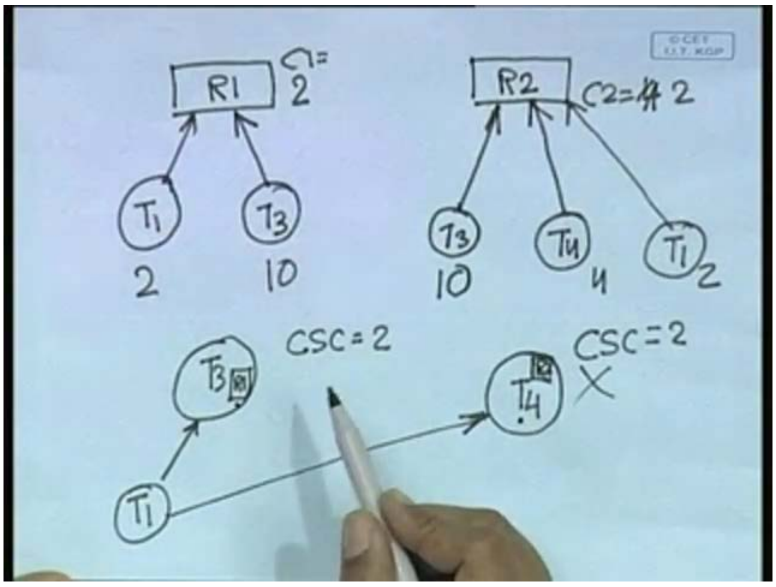
But such a situation will not occur, why is that? Anybody can argue, why such a situation cannot occur? T 3 is holding R1, T 4 is holding R2, T 1 first waited for T 3, T 3's priority increased due to inheritance clause and then T 1 waited for T 4 to release R2. Why such a situation is not possible? Anybody would like to give any answer? So, first T 3 has acquired R 1, then T 4 has acquired R 2.

It is not possible because if a… the system CSC will not… Before we not (Conversation not audible. Refer Time: 27:28)

Exactly, exactly very good.

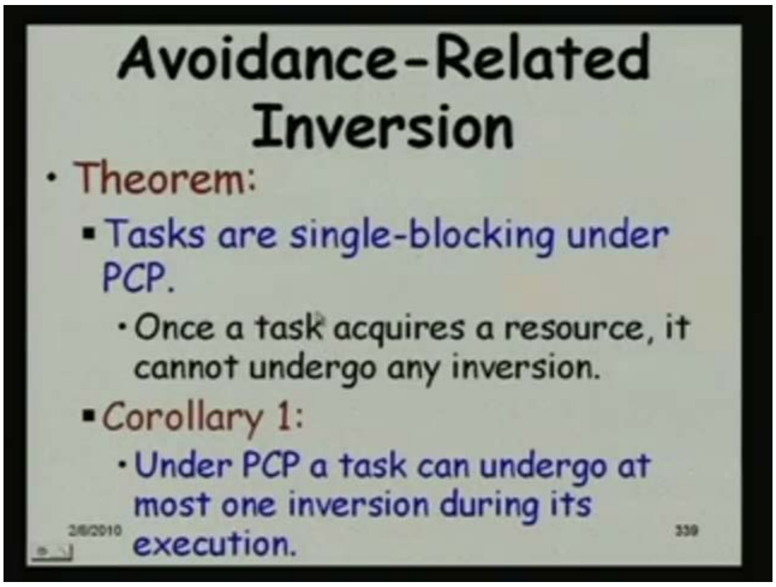
Just look at his argument, he says that see T 3 has set has got resource R1, CSC will be set to 2. Now, when T 4 requests R2, it is priority will be checked against CSC. So, it could not have been granted R2. Similar argument we can show, if T 4 first got R2 and T 3 was trying to get R1. So it is not possible.

(Refer Slide Time: 28:02)



We can take all cases, I mean whatever example you can give, that T 3 was holding both resources or any other example that you can think of, you can give, and we can show, that such a situation will not be possible. And that we can just write formally, saying that see, these are the situations one task can hold both resources or they are held in two different tasks, and for each case we will show that it is not possible. And therefore, under the priority ceiling protocol, the tasks are single blocking, and once it gets the resource, it will not block for any other resource. They will be free by that time. So, it has the property of the highest locker protocol, even without increasing the priority of the task to the ceiling value.

(Refer Slide Time: 28:54)



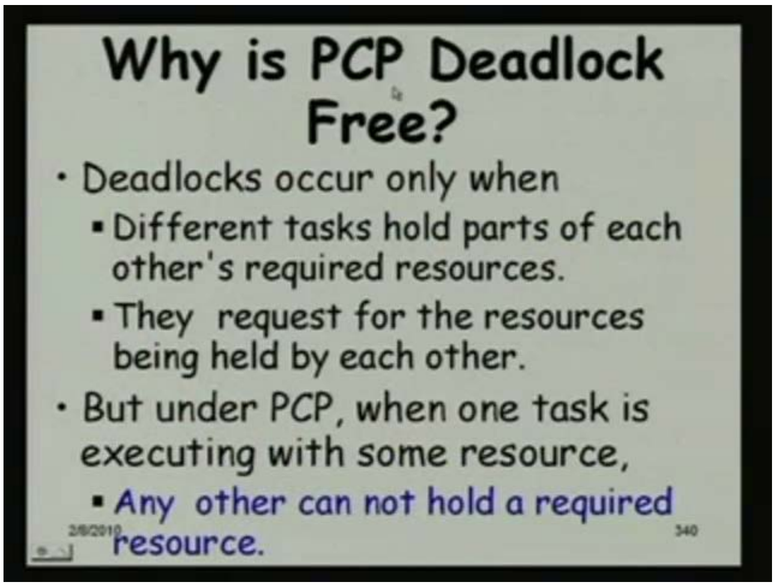
Avoidance-Related Inversion

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

2/8/2010 339

So, the theorem that tasks are single blocking under PCP, we can construct a formal proof based on the informal argument that we gave. We will just formal, we can just formalize that. That is what is done in the book. The same basic ideas are formalized, and the corollary of this is that - a task can undergo at most one inversion; it cannot undergo two direct inversions; it cannot undergo one direct inversion, and then, followed by one avoidance-related inversion; it is not possible. So, let us proceed with this.

(Refer Slide Time: 29:33)



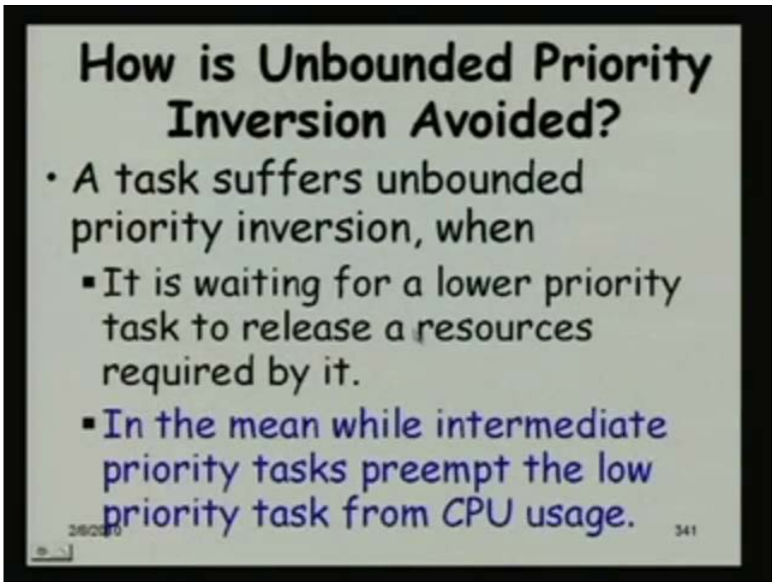
Why is PCP Deadlock Free?

- Deadlocks occur only when
  - Different tasks hold parts of each other's required resources.
  - They request for the resources being held by each other.
- But under PCP, when one task is executing with some resource,
  - Any other can not hold a required resource.

2/8/2010 340

Next let us argue, why the priority ceiling protocol is deadlock free. So, the idea is that, the deadlock can occur only when the different tasks hold parts of each other's required resource. So, one task is executing with some resource and part of its resources is being held by another resource - another task. But that we have already shown, that if it is holding one resource, then other resources must be free; so, undergoing at most one blocking. So by the theorem, it is easy to show that it will be dead lock free.

(Refer Slide Time: 30:21)



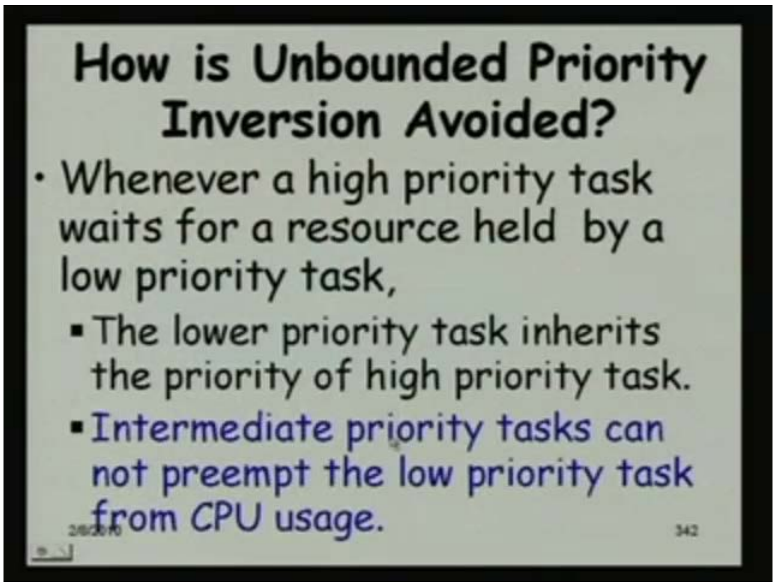
How is Unbounded Priority Inversion Avoided?

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

2/8/2010 341

Now, what about unbounded priority inversion? A task could suffer unbounded priority inversion, when it is waiting for a lower priority task to release the resource, and an intermediate priority task is preemting that task from the CPU uses, and the higher priority task keeps on waiting.

(Refer Slide Time: 30:46)



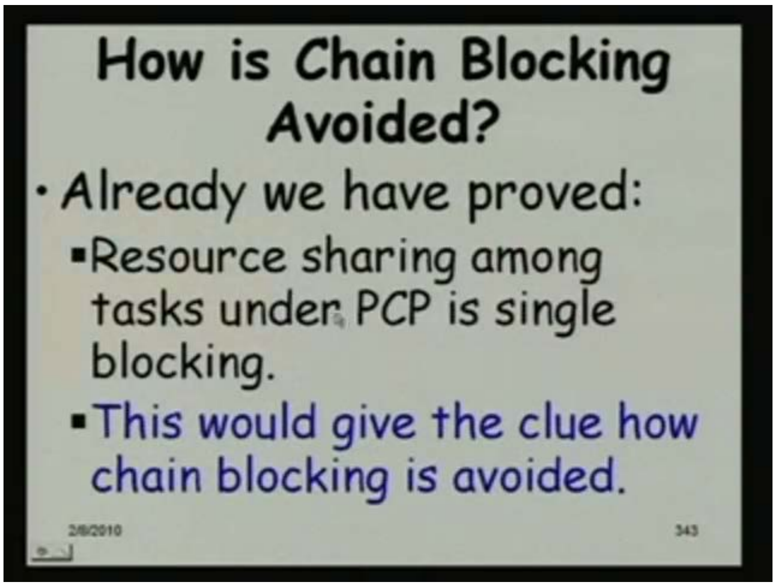
How is Unbounded Priority Inversion Avoided?

- 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.

342

But here, that is not possible, because it would inherit the priority of the highest waiting priority task by the inheritance clause. So, the inheritance clause will prevent unbounded priority inversion. So, not possible that intermediate priority tasks would preempt the lower priority task from CPU users, whereas the higher priority waiting task keeps on waiting.

(Refer Slide Time: 31:10)



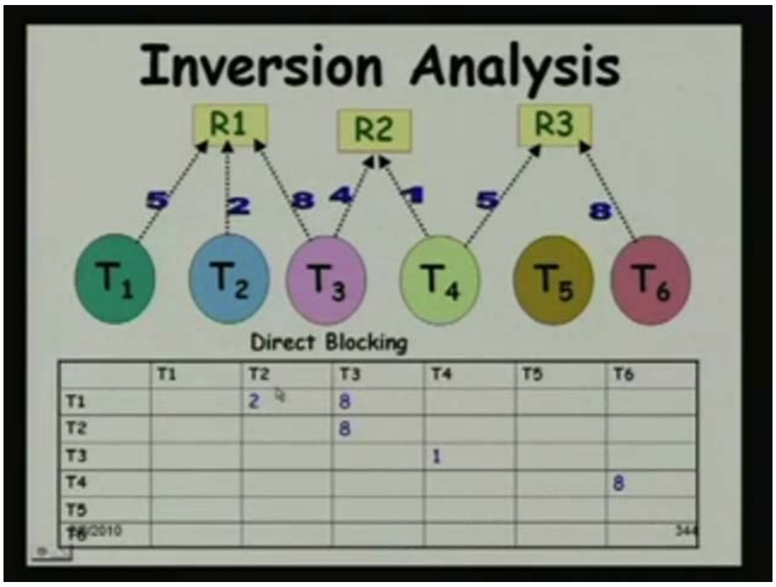
How is Chain Blocking Avoided?

- Already we have proved:
  - Resource sharing among tasks under PCP is single blocking.
  - This would give the clue how chain blocking is avoided.

2/8/2010 343

Now, what about chain blocking? I have already shown that it is single blocking. And, if you say how chain blocking is avoided, then we will have to show how we are good that the it is single blocking. So, to show that chain blocking does not exist, we will have to show that how the priority ceiling, the system ceiling was set, and how the priorities were compared of the task requesting resource. And that is why another task, two task cannot be holding two resources that are required by a task. The scenarios are not allowed under this protocol right.

(Refer Slide Time: 31:59)



Inversion Analysis

Direct Blocking

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

2/8/2010 344

Now, let us do some analysis. First, I will just do the analysis here on the slide, just observe it, and then, I will give some example for you to work out. So, the tasks are ordered according to their priority, and these are their resource requirements, the time duration for which they need the resource, milliseconds. Now, let us find the direct inversion. I think there is a mistake here. So, T 1 will undergo… I think it should have been T 1, I have just written it by mistake.

So, T 1 undergoes direct inversion due to T 2 for 2 milliseconds. Is it not? I know that is… correct correct correct.

So, I think, the see I think I just interpreted the rows and columns differently. See here, these are the task names and the T1 suffers direct inversion due to which task?

T 2.

T 2, it is correct. Actually, I have done it, but I just got confused. So, T1 undergoes inversion due to T2 for 2 milliseconds, direct inversion. Due to T 3, it undergoes direct inversion for 8, and it does not need any other resource, it does not undergo any direct inversion with any another task. Now, what about T 2? T 2 needs only R1. So, it will undergo direct inversion due to T 3 for 8 millisecond right maximum.

Now, what about T 3? T 3 needs two resources R1 and R2, but T 3 cannot undergo an inversion due to R1, because these are higher priority tasks. Right? So, T 3 can undergo inversion due to R T 4, maximum for 1 millisecond.

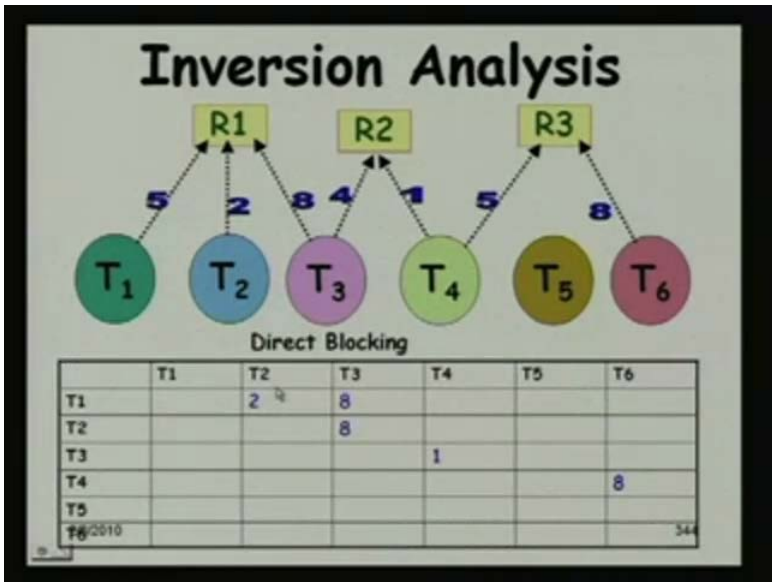
What about T 4? It cannot undergo due to T 3, but it can undergo inversion due to T 6 for 8 millisecond right, this 8 millisecond it can undergo.

What about T 5? T 5 does not need any resource. So, it cannot undergo any direct inversion.

And T 6 is the lowest priority tasks, it cannot go undergo any type of inversion, not only the direct inversion.

Now, what about the inheritance inversion? I do not have the diagram here.

(Refer Slide time: 35:02)



Inversion Analysis

Direct Blocking

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

So, you have to. So, just check here, what will be the inheritance inversion for T 1? Nothing, T 1 will not undergo inheritance inversion.

What about T 2? Will T 2 undergo any inheritance inversion?

No.

What about T 3?

It will not undergo any inversion.

It will not undergo any inversion. Is it? Is it?

Yes, please look at it; will T 3 undergo any inheritance inversion?

(No Audio till 36:07 min)

It cannot undergo inheritance inversion, it cannot undergo inheritance inversion, because…

Sir, T 2 can undergo inheritance inversion.

T 2 due to…

Sir, because like, the T 3 is T 3 is holding it.

T 3 is holding that.

Holding R1.

T 3 is holding R1, then T 2 will undergo a direct inversion.

Of course and T 1 is waiting.

T 1 has, (Audio not audible. Refer Time: 36:28) And now, T 2 as 1. So, that T 2 has to wait.

So, that is a… So T 2. So, T 3 is holding R1 and T 1 is waiting for T 3, then T 3's priority would have been increased.

It should be T 1.

So, that will be avoidance inversion. Is it not?

(No Audio till 37:00 min)

What about T 4? Will it undergo inheritance inversion?

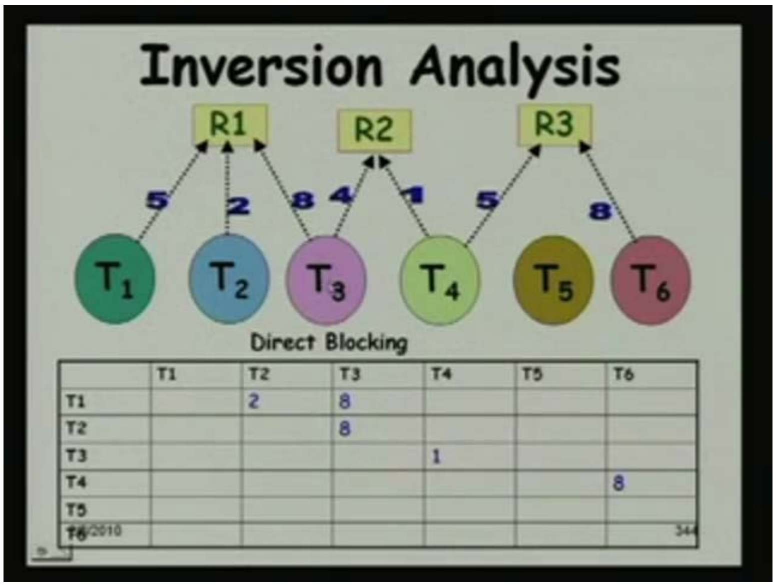
(No Audio till 37:16 min)

Yes.

Will T 4 undergo inheritance related inversion? See, just look at the basic definition of inheritance-related inversion. In an inheritance-related inversion, a task is not progressing, even though it does not need a resource, because the priority of the task that is holding the resource has increased. I think, according to that definition T 2 will follow inheritance inversion. Is it not?

See, what he says is that T 2, let us say, it is not needing R1 for some time, it is executing. And then, because T 3 is, T 1 is T 3 is holding R1 and T 1 is waiting for R1. So, T 2's ,T 3's priority would be raised to that of T 1 and T 2 even though when it is not needing needing R1 it cannot execute, is it not? So, T 2 will undergo inheritance inversion, that is correct. Do you see the difference here? Inheritance inversion occurs, when, even when a task is not needing a resource, it is not being able to progress, because the task that is holding the resource, its priority has increased due to another higher priority task waiting. But in avoidance inversion, a task needing a resource not required by any other task is blocked, because of the current system ceiling comparison rule. Is that ok?

(Refer Slide Time: 38:55)



Inversion Analysis

Direct Blocking

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

So, yes, T 2 will undergo inheritance inversion, due to T 3, for what duration?

8.

8.

Now, what about T 3, will it undergo an inheritance inversion? No, it cannot undergo any, because no other task can inherit another higher priority task.

What about T 5?

T 5 will undergo inheritance inversion.

Yes.

T 5 will undergo inheritance related inversion. Due to which task?

Due to T 6.

T 6, naturally.

Now, what about the avoidance-related inversion? Which task will undergo avoidance-related inversion? Will T 1 undergo avoidance-related inversion?

T 6.

Due to which task?

Due to holding R1, see, if we have R1. So, if you have R1 t equal to T1. The T1…

No, but T 1 does not require any (( )) saying that…

So, T 1 will directly blocked by T 2, that we have said. So, will there be any avoidance-related inversion?

T 2.

T 3.

T 3.

So, T 3, due to which task?

T 3 requires R2.

T 3 requires R2, yes.

But it is not granted as the T 1 of T 2 is using R1.

No, it would not suffer an inversion due to higher priority task. If a higher priority task is executing, it is not an inversion. Inversion is that when a lower priority task is executing and this is prevented.

T 4.

Let us see, T 4.

T 6 is holding R3, and CSC, if R3 is equal to T 4, and a T 4 requests R2, it cannot be granted, because the CSC is equal to T 4 it is not greater than that T 4.

So, T 4 will undergo an avoidance inversion.

Because CSC of R3 is equal to T 6.

Yes. Exactly. Exactly.

So, T 4 is holding sorry T 6 is holding R3 and the CSC is set to the ceiling of R3, which is T 4, and T 4 started executing, and then after sometime it requested R2.

Sir, it is not greater than CSC.

It is not greater than CSC. So, it will be prevented from using R2. So that will result in an avoidance inversion, but what duration?

8.

8.

Sir, will the same thing will hold for T 3?

Yes, so…

So, what about T 3?

Sir, T 4 is holding R2.

T 4

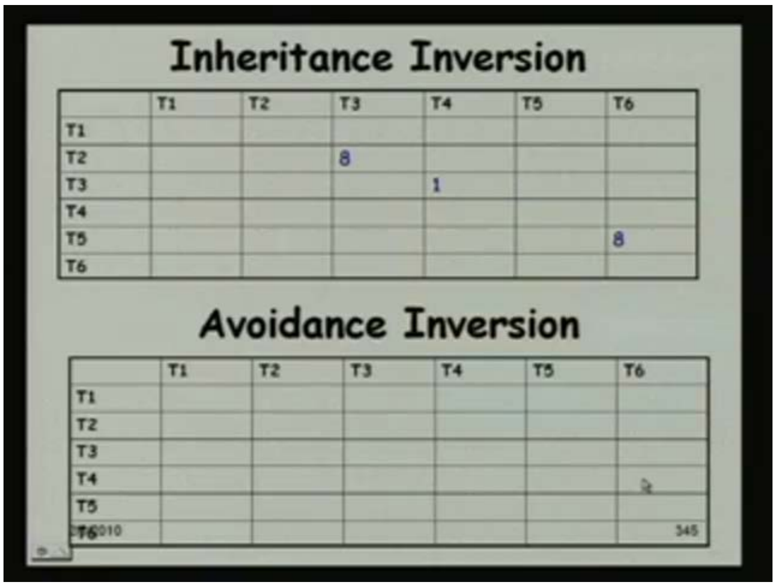
Yes.

And then, T 3 cannot access R1.

Yes.

T 3 cannot access R1. So, T 3 will also undergo avoidance-related inversion due to R2 for 1 millisecond. So, is the computation clear?

(Refer Slide Time: 42:08)



Inheritance Inversion

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

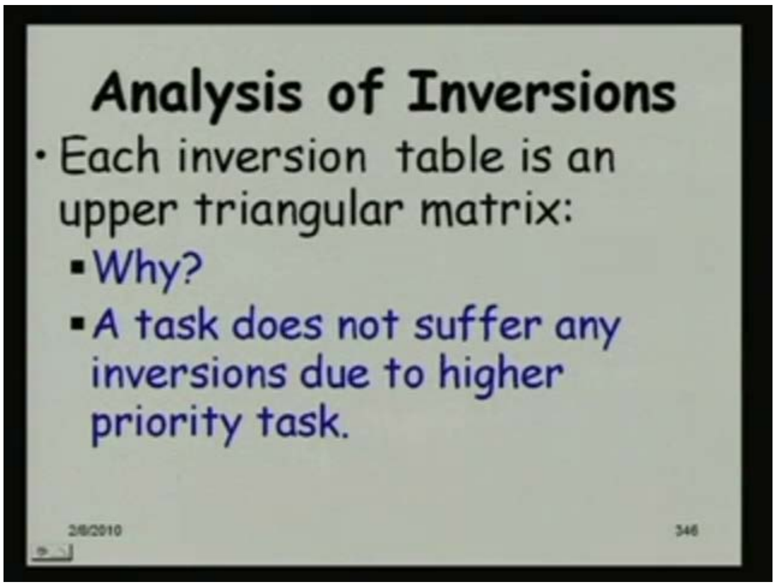
Avoidance Inversion

	T1	T2	T3	T4	T5	T6
T1						
T2						
T3						
T4						
T5						
T6						

2/6/2010 345

So, let us… So, I have written those values here.

(Refer Slide Time: 42:12)



Analysis of Inversions

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

2/6/2010 346

So now, one thing that we need to look at, is that each of this matrix, we just did 3 tables, one is the direct inversion, the other is the avoidance inversion, and we did the inheritance inversion - 3 different tables. Now, all the 3 tables will be upper triangular matrices. Why is that? You draw for any tasks set, you will find that the inversion tables, each of them, are upper triangular matrices

So, it is only on the priority issue.

So, then then why, then started does not… I mean, why should it make it upper triangle?

(Audio not clear. Refer Time: 43:01)

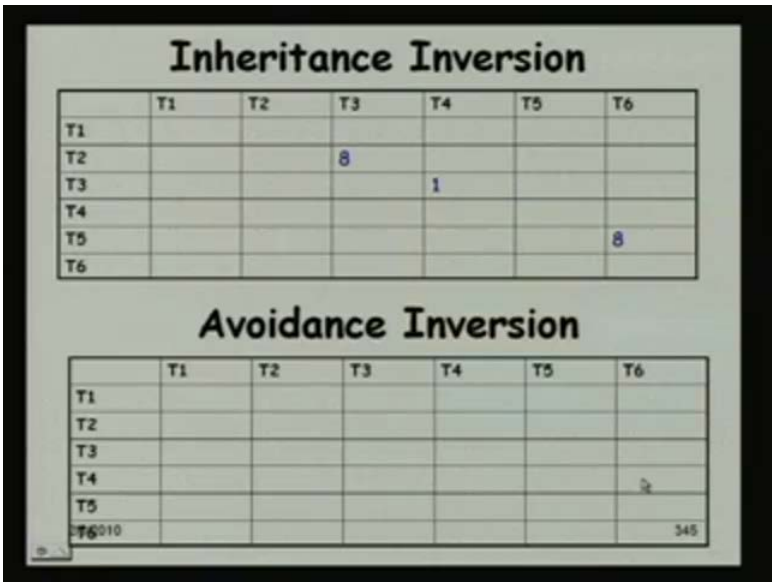
Exactly, so, the reason is that, a task does not suffer any inversion due to higher priority tasks. So, one part of the table will be empty, the lower priority with respect to a higher priority task will not undergo inversion. So, that part will be empty. So, only half of that some entries will be there.

Sir,

Yes.

Will it be (( )) to the table

(Refer Slide Time: 43:48)



Inheritance Inversion

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

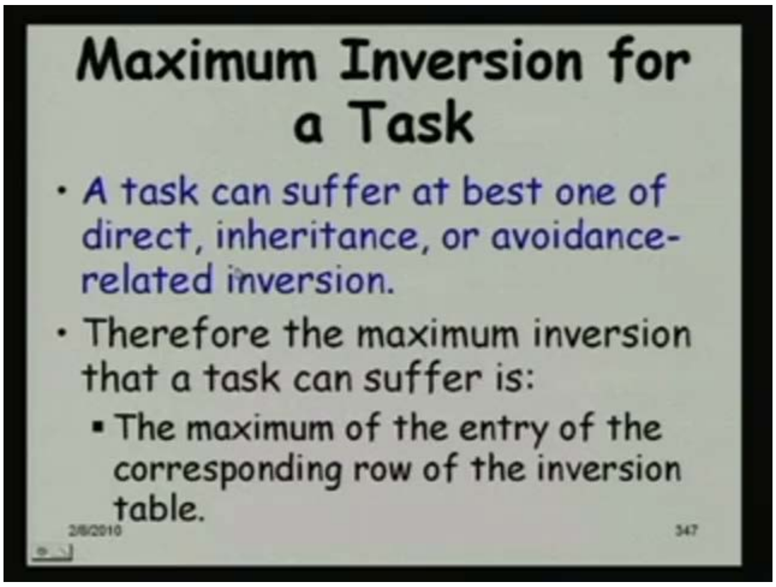
Avoidance Inversion

	T1	T2	T3	T4	T5	T6
T1						
T2						
T3						
T4						
T5						
T6						

This one. We identified T 3 and T 4.

I have not written that here, but what you identified - is that clear? Yes. So, we will take some example and then we will so. Here, I am not populated all this. We need to populate here. The avoidance related inversions here, we identified that T 3 will undergo an avoidance related inversion and T 4 will undergo avoidance. So, we need to populate that in the table.

(Refer Slide Time: 44:04)



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 entry of the corresponding row of the inversion table.

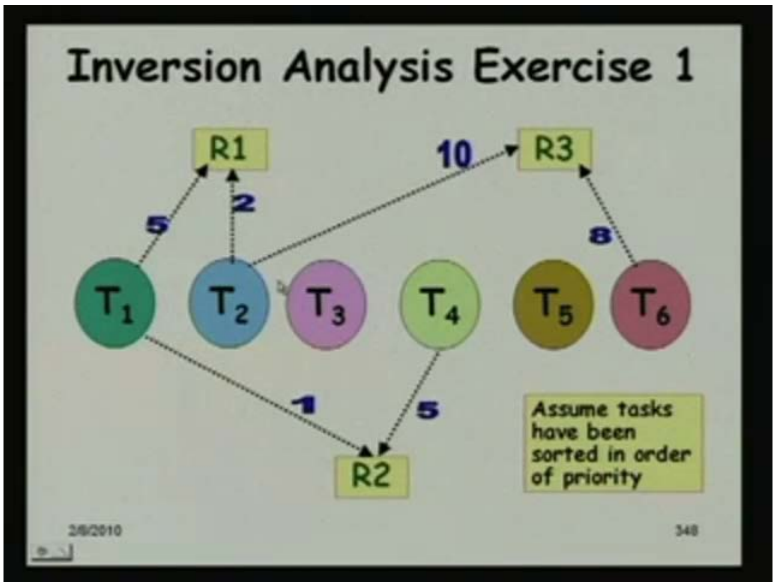
2/8/2010 347

But one thing is that, a task undergoes at most one of the inversions by our theorem. So, the maximum inversion that a task can suffer, we can check the entries under the all the three tables, for a row, and find what is the maximum value there. Any of the entries in the three rows and then pick that one.

It is a worst case.

Worst case, yes. We are trying to find the worst case, the maximum inversion that a task can occur. Because all the scheduling, all analysis with respect to the worst case.

(Refer Slide Time: 44:52)


Inversion Analysis Exercise 1
R1
R2
R3
T1
T2
T3
T4
T5
T6
5
2
10
8
1
5
Assume tasks have been sorted in order of priority
2/8/2010
348

Now, let us do an exercise. So, let us assume this is the situation, again six tasks arranged increasing order of their priority. So, let us find the three types of inversions, and then, find the maximum inversion that a task can suffer.

So, first let us do the direct inversion. So, you you tell me the values, I will write here.

The direct inversion table, so, we have the tasks T 1, T 2, T 3, T 4, T 5, T 6 and let us write T 1, T 2, T 3, T 4, T 5, T 6. First, I look at the diagram and then, tell me, what are the direct inversions, we will just populate on this table right.

So…

So, just look at the diagram. So, what are the direct inversions for T 1. T 1 will undergo direct inversion due to which task and for what duration?

T 2 and T 3.

It will undergo direct inversion with T 2, for what duration?

2

2

2 millisecond yes and…

T 4

And also, for T 4 for 5 millisecond.

Now, what about T 2?

T 2 undergoes the T 1 to T 2.

Yes, T 6 alone for 8 milliseconds

What about T 3?

It will not undergo.

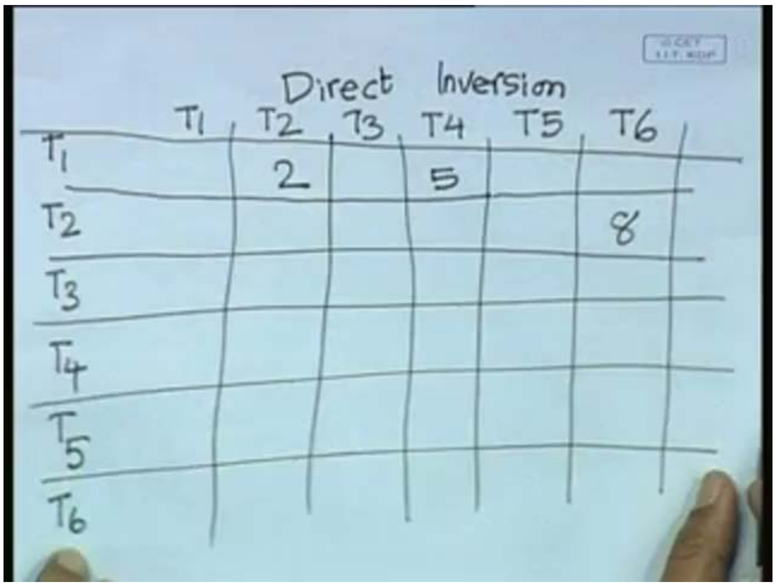
It will not undergo any direct inversion, because it does not need any resource. What about T 4?

No sir.

It will also not undergo any inversion, because if T 1 is holding R2, then it is not an inversion, because that is a higher priority. What about T 5? It will also not undergo any inversion.

What about T 6? It will also not undergo any inversion, because this is the lowest priority, all others are higher priority.

(Refer Slide Time: 47:27)



Direct Inversion

	T1	T2	T3	T4	T5	T6
T1		2		5		
T2						8
T3						
T4						
T5						
T6						

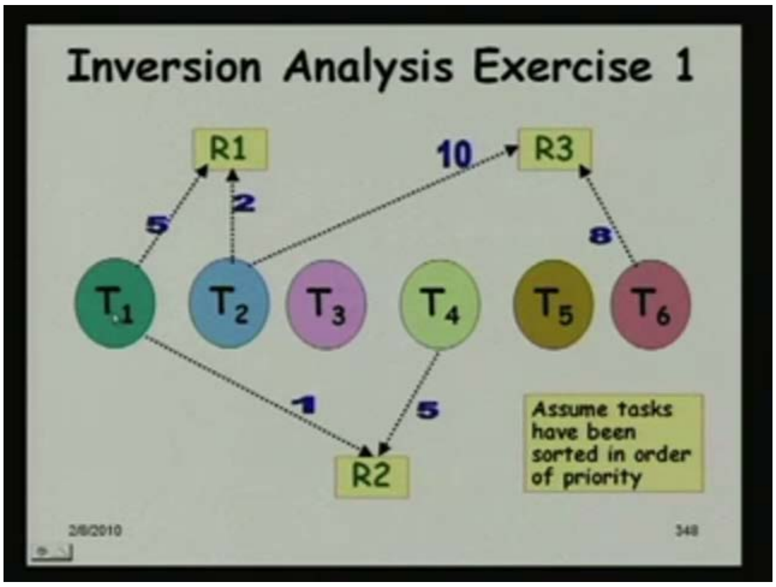
So, let me just write down what you told me. So, T 1 will undergo inversion due to T 2 for 2.

Sir, T 2.

And for T 4 for 5, and T 2 will undergo due to T 2 for 8, and T 3 will not undergo any inversion, T 4 will also not undergo any inversion, T 5, T 6 etcetera, none of them will not undergo inversion. So, only T 1 is undergoing inversion due to two tasks, and T 2 due to one task. So, let me just draw the line, so, that we do not get confused here. Now let me just do, let us do the inheritance inversion.

So, again look at the diagram, and please tell me about the inheritance related inversion. So, again we will write T 1, T 2, T 3, T 4, T 5, T 6. T 1, T 2, T 3, T 4, T 5, T 6. So now, let us look at the diagram, and then, identify what are the inheritance inversions; I will populate on this table.

(Refer Slide Time: 49:23)


Inversion Analysis Exercise 1

So, the tasks what about T 1? Will it undergo an inheritance inversion? So, inheritance inversion just remember that, the task that is holding the resource, and there is a task waiting for resource, and the task holding the resource has inherited the priority of the waiting task, which is greater than the task that is we are considering, and then it is preventing it from being executed.

So, what about T 1? Will there be an inheritance inversion?

No.

Due to which task?

(Audio not clear. Refer Time: 50:09)

No, no.

See, we have to have a situation where a task is waiting for a task holding the resource whose priority, the waiting task's priority should be more than T 1.

Sir, T 1 will not undergo.

T 1 will not undergo any inversion, because this is the highest priority task. Right? Any other task that is waiting will be lower than T 1. So, T 1 cannot undergo inheritance inversion.

What about T 2?

T 2 will undergo due to T 4.

T 2 will undergo due to T 4. T 4 is holding R2 and T 1 is waiting, its priority is raised, and T 2 cannot execute. Yes, T 2 will undergo inheritance inversion due to T 4 for 5 milliseconds right.

Now, what about T 3?

It will not undergo, due to T 4.

T 3 will undergo due to T 4 for 5, and T 6 for 8.

What about T 4?

It will undergo due to T 6.

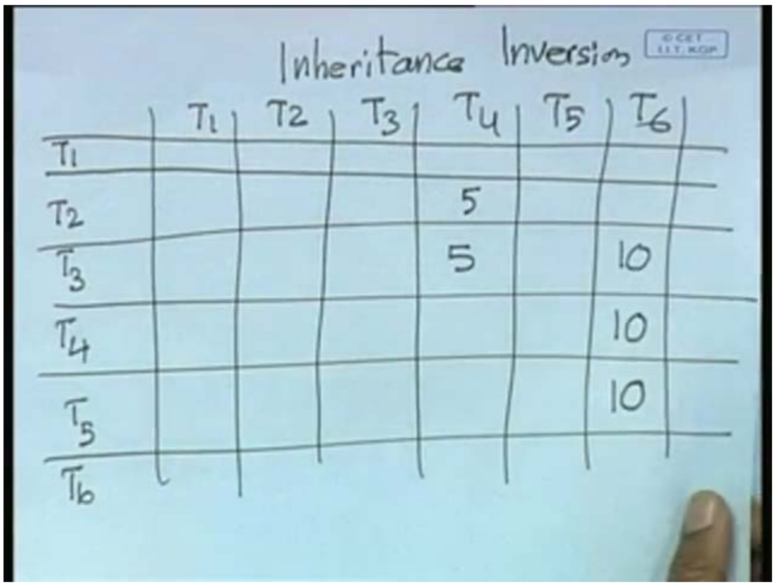
T 4 due to T 6.

What about T 5?

Due to T 6.

T 5 due to T 6.

(Refer Slide Time: 51:25)



Inheritance Inversion

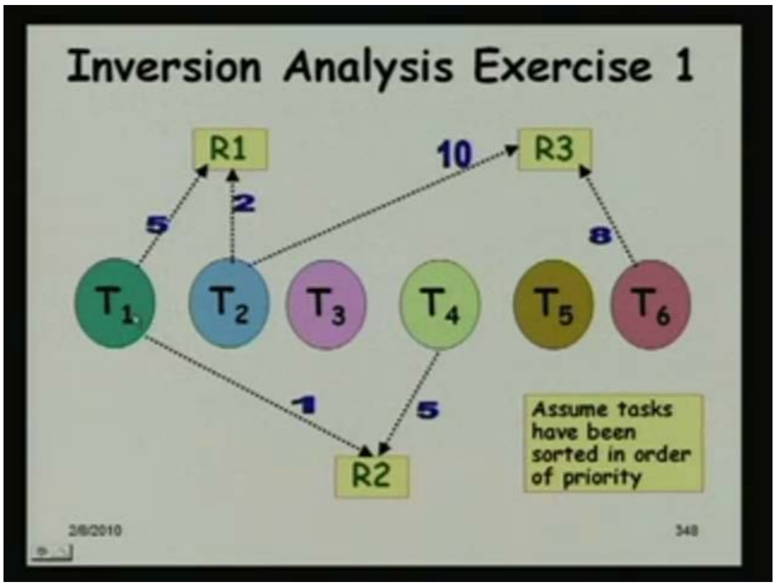
	T1	T2	T3	T4	T5	T6
T1						
T2				5		
T3				5		10
T4						10
T5						10
T6						

Let us just, whatever we said, let me just write here, that T 1 does not undergo any inversion, and then, T 2 will undergo on account of T 4 for 5 milliseconds, and T 3 will undergo on account of T 4 for 5 milliseconds and on account of T 6 for 10 milliseconds, and T 4 will undergo due to T 6 for 10 milliseconds and T 5…

Sir, T 5 will undergo due to T 6.

T 5 will undergo due to T 6 for 10 milliseconds. So, this is the inheritance inversion table. Now, let us work out the avoidance inversion table. So, again let me just draw the table avoidance inversion table. T 1, T 2, T 3, T 4, T 5, T 6, T 1, T 2, T 3, T 4, T 5, T 6. Now, let us look at the diagram and try to find the avoidance inversions

(Refer Slide Time: 53:07)


Inversion Analysis Exercise 1
R1
R2
R3
10
5
2
8
1
5
T_1
T_2
T_3
T_4
T_5
T_6
Assume tasks have been sorted in order of priority
2/8/2010
348

So, first let us look at T 1, will it undergo avoidance inversion? Yes, in avoidance inversion remember, that the CSC is set to a priority higher than sorry equal to or higher than the task and, it cannot execute.

Sir, T 1 can undergo due T 4 like if T 4 is holding R2. So, the CSC is set to 1.

Yes.

And then T 1 cannot activate R1.

R1, exactly. So, T 1 can undergo an avoidance inversion due to T 4 for 5.

Then this is equal to T 1 can also facing for T 2, because T 1 is not holding R1. So, the CSC of is set to 1 and then T 1cannot acquire R2.

No no… Sorry.

What is the scenario?

Sir, it can be both you know…

No no.

Let us…

T 4 is holding R2. T 4 is holding R2. CSC is not set to1. So that, task T 1cannot acquire R1. Then, suppose now T 2 is holding R1. So, the CSC value is set to 1.

Right right.

Then, T1 cannot acquire R2.

So, what are that T 1 undergoes in in avoidance inversion due to which two tasks?

T 2 and T 4.

T 2 and T 4.

So, that is for 2 and 5. What about T 2?

Sir, T 2…

No, you can (( )) cannot work.

What about due to T 4?

No, it will not.

Why not? See, T 4 is holding R2, priority set to 1.

Yes, and that time it will go.

And then, T 2 is needing R3 or R2, R1or R3

So, it will undergo 5 due to R T4. What about T 3? Will not undergo avoidance inversion, because it does not need any resource.

What about T 4?

T 4… we will again.

T 4 will undergo inversion due to T 6, because T 6 is holding R3 and T 4 is prevented from locking R2.

What about T 5?

No, sir, no.

T 5 it does not need any resource it will… So, we will we can write those values here on the table. So, T 1 does not undergo any avoidance inversion and…

Sir, before and then T 1 undergoes

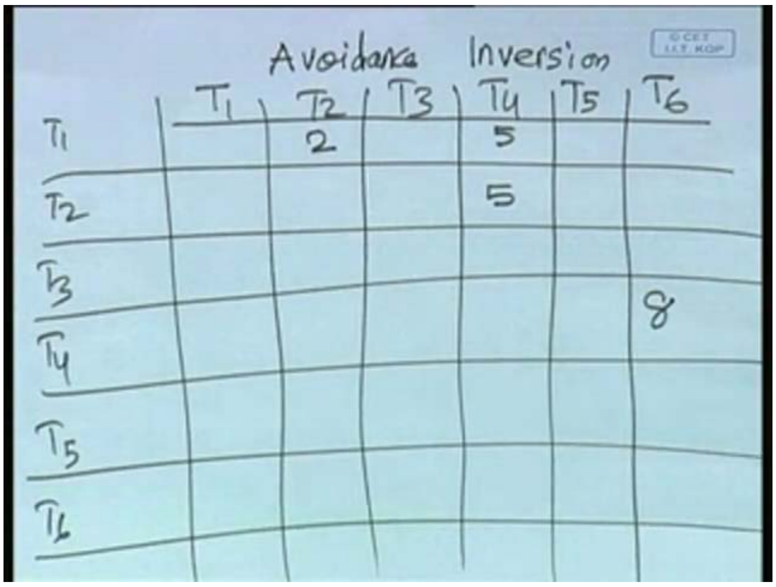
And…

Sir, due to T 2 and T 4.

T 1 undergoes due to T 2 and T 4

Yes, right.

(Refer Slide Time: 55:43)



Avoidance Inversion

	T_1	T_2	T_3	T_4	T_5	T_6
T_1		2		5		
T_2				5		
T_3						
T_4						8
T_5						
T_6						

T 1 undergoes due to T 2 and T 4. So, for 2 and 5, and T 2 undergoes on account of T 4. So, for 5, T 3 does not need any resource does not go any undergo any inversion, T 4 undergoes on account of T 6 for 8, and T 5 does not need any resource, does not undergo any avoidance inversion, T 6 lowest priority does not undergo any avoidance inversion. And then, we find out the highest maximum inversion for each task. So, here avoidance inversion, the maximum value for T 1 is 5, and if we look at the inheritance inversion it does not undergo, T 1 does not undergo any inheritance inversion. Look at the direct inversion it is again 5. So, the maximum inversion for T 1 is 5. T 2 avoidance inversion is 5, inheritance inversion is 5, direct inversion is 8. So, it will maximum inversion it can undergo is 8. Similarly, T 3 undergoes no inversion, no avoidance inversion, inheritance inversion 10. So, maximum inversion is 10.

So, we can work out like that, and based on the maximum inversion, in the next class, we will find out the total completion time, worst case completion time for a task, and see, whether it will meet the schedule, even when the tasks undergo a shared resources right. We will extend the Livosky’s clause - Livosky’s expression - and we will incorporate these values to check the schedulability criteria right. We will stop here.