Real Time Systems1

lecture 10

Priority Ceiling Protocol

PRIORITY CEILING PROTOCOL

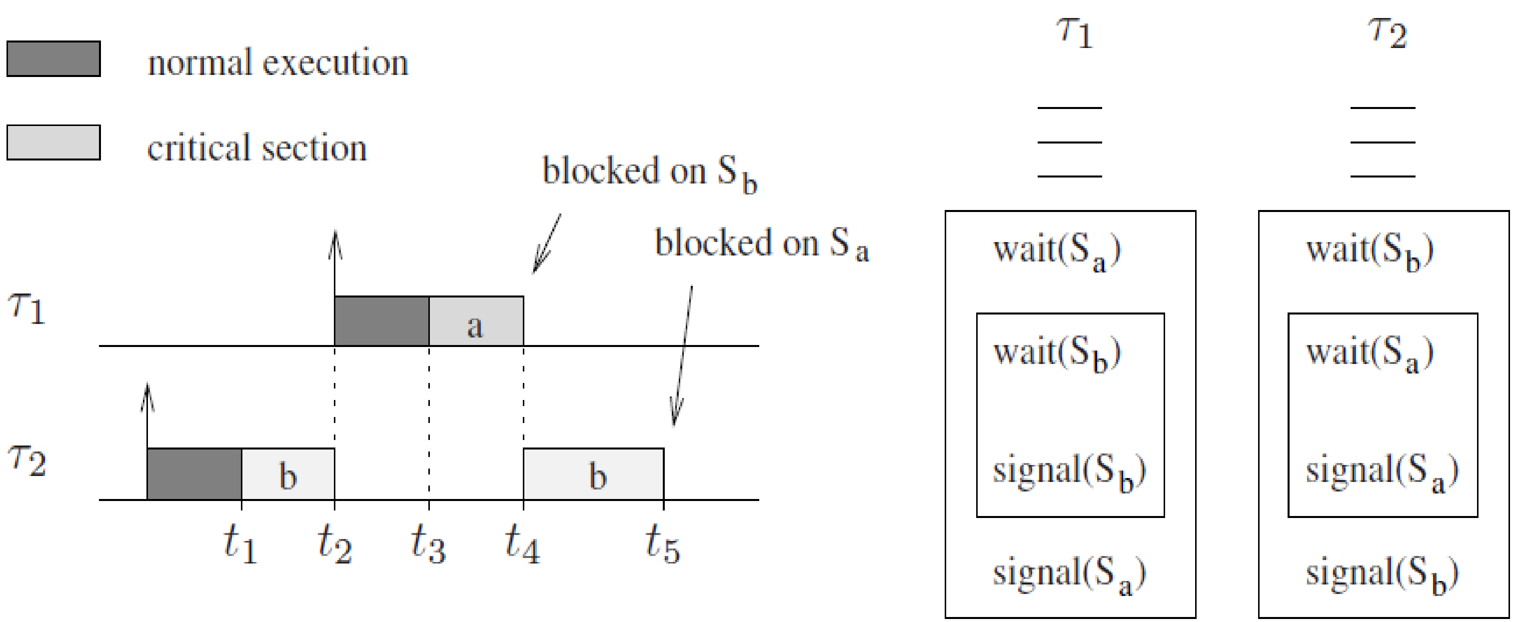

Figure 7.13 Example of deadlock.

PRIORITY CEILING PROTOCOL

- The Priority Ceiling Protocol (PCP) was introduced by Sha, Rajkumar, and Lehoczky [SRL90] to bound the priority inversion phenomenon and prevent the formation of deadlocks and chained blocking.

PRIORITY CEILING PROTOCOL

- The basic idea of this method is to extend the Priority Inheritance Protocol with a rule for granting a lock request on a free semaphore.
- To avoid multiple blocking, this rule does not allow a task to enter a critical section if there are locked semaphores that could block it.
- This means that once a task enters its first critical section, it can never be blocked by lower-priority tasks until its completion

PRIORITY CEILING PROTOCOL

- In order to realize this idea, each semaphore is assigned a *priority ceiling* equal to the highest priority of the tasks that can lock it.
- Then, a task $\tau\ i$ is allowed to enter a critical section only if its priority is higher than all priority ceilings of the semaphores currently locked by tasks other than τi.

PRIORITY CEILING PROTOCOL

- **The Priority Ceiling Protocol can be defined as follows**:
- 1. Each semaphore S_k is assigned a priority ceiling $C(S_k)$ equal to the highest priority of the tasks that can lock it. Note that $C(S_k)$ is a static value that can be computed off-line: $C(S_k) \stackrel{def}{=} \max_i \{P_i \mid S_k \in \sigma_i\}$.
- 2. Let τ_i be the task with the highest priority among all tasks ready to run; thus, τ_i is assigned the processor.

PRIORITY CEILING PROTOCOL

- 3. Let $S*$ be the semaphore with the highest ceiling among all the semaphores currently locked by tasks other than τi and let $C(S*)$ be its ceiling.
- 4. To enter a critical section guarded by a semaphore Sk, τi must have a priority higher than $C(S*)$.
- **If $Pi \leq C(S*)$, the lock on Sk is denied and τi is said to be blocked on semaphore $S*$ by the task that holds the lock on $S*$.**

PRIORITY CEILING PROTOCOL

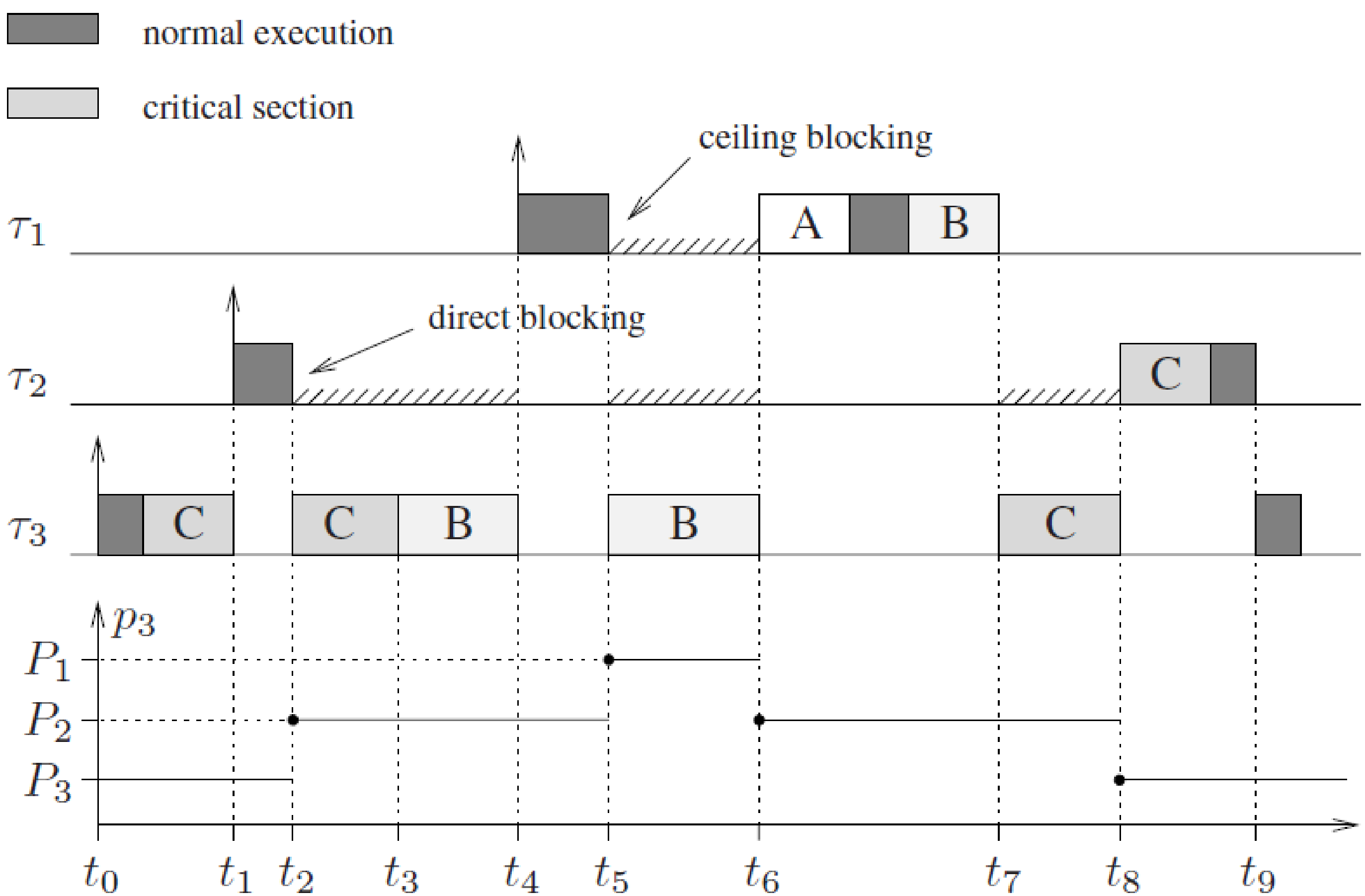

Figure 7.14 Example of Priority Ceiling Protocol.

PRIORITY CEILING PROTOCOL/ Example

- Figure 8-10 shows the schedule of the system of jobs when their accesses to resources are controlled by the priority-ceiling.

Job	r_i	e_i	π_i	Critical Sections
J_1	7	3	1	[*Shaded*; 1]
J_2	5	3	2	[*Black*; 1]
J_3	4	2	3	
J_4	2	6	4	[*Shaded*; 4 [*Black*; 1.5]]
J_5	0	6	5	[*Black*; 4]

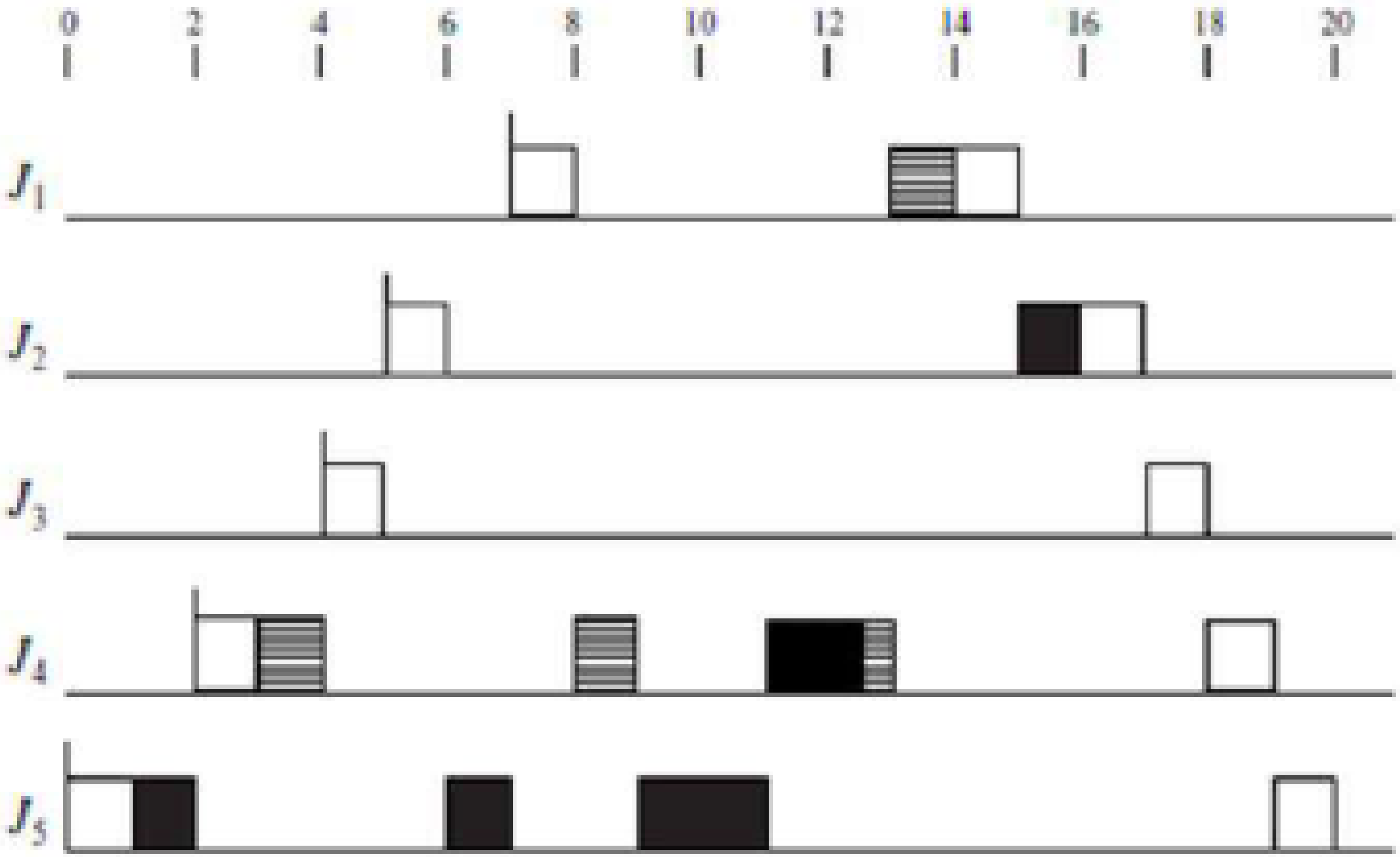

Figure 8-8 Example of Priority priority-inheritance Protocol

PRIORITY CEILING PROTOCOL

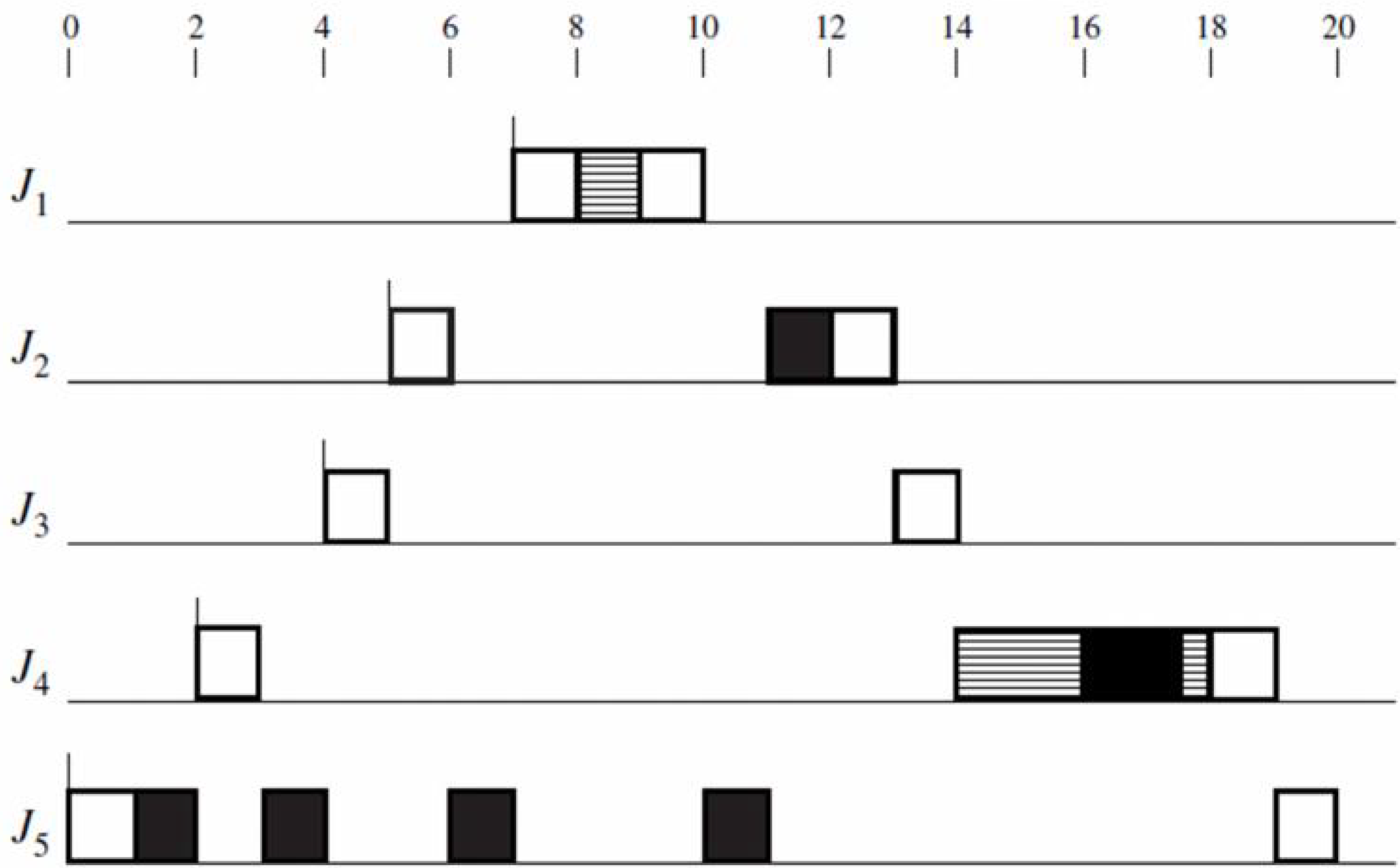

Figure 8-10 Example of Priority Ceiling Protocol

PRIORITY CEILING PROTOCOL/ Example

- As stated earlier, the priority ceilings of the resources *Black* and *Shaded* are 2 and 1, respectively.
- **1.** In the interval *(*0*,* 3], this schedule is the same as the schedule shown in Figure 8-8, which is produced under the basic priority-inheritance protocol.
- In particular, the ceiling of the system at time 1 is . When *J*5 requests *Black*, it is allocated the resource.

PRIORITY CEILING PROTOCOL/ Example

- After *Black* is allocated, the ceiling of the system is raised to 2, the priority ceiling of *Black*.

- **2.** At time 3, *J*4 requests *Shaded*. *Shaded* is free; however, because the ceiling (= 2) of the system and its higher than the priority of *J*4, *J*4's request is denied. *J*4 is blocked, and *J*5 inherits *J*4's priority and executes at priority 4.

PRIORITY CEILING PROTOCOL/ Example

- 3. At time 4, J_3 preempts J_5, and at time 5, J_2 preempts J_3. At time 6, J_2 requests *Black* and becomes directly blocked by J_5.
- Consequently, J_5 inherits the priority 2; it executes until J_1 becomes ready and preempts it.
- During all this time, the ceiling of the system remains at 2.
- **4.** When J_1 requests *Shaded* at time 8, its priority is higher than the ceiling of the system.
- Hence, its request is granted, allowing it to enter its critical section and complete by the time 10.

PRIORITY CEILING PROTOCOL / Example

- At time 10, *J*3 and *J*5 are ready. The latter has a higher priority (i.e., 2); it resumes

- 5. At 11, when *J*5 releases *Black*, its priority returns to 5, and the ceiling of the system drops to 5 . *J*2 becomes unblocked, is allocated *Black*, and starts to execute.

- **6.** At time 14, after *J*2 and *J*3 complete, *J*4 has the processor and is granted the resource *Shaded* because its priority is higher than the ceiling of the system at that time. It starts to execute.

PRIORITY CEILING PROTOCOL/ Example

- The ceiling of ***the system is raised to 1, the priority ceiling of Shaded.***

- **7.** At time 16, *J*4 requests *Black*, which is free. The priority of *J*4 is lower than 1, but *J*4 is the job holding the resource (i.e., *Shaded*) whose priority ceiling is equal to 1.

- Hence, *J*4 is granted *Black*. It continues to execute. The rest of the schedule is self-explanatory.

PRIORITY CEILING PROTOCOL

- Comparing the schedules in figures 8-8 and 8-10, we see that when priority-ceiling protocol is used, *J*4 is blocked at time 3.
- A consequence is that the higher priority jobs *J*1, *J*2, and *J*3 all complete earlier at the expense of the lower priority job *J*4.
- This is the desired effect of the protocol

PRIORITY CEILING PROTOCOL

- HW:

task	ri	ei	recourses
T1	10	4	A:1
T2	7	4	A:1 , B:1
T3	4	4	B:1 , C:1
T4	0	11	[A:5 [B:2]]