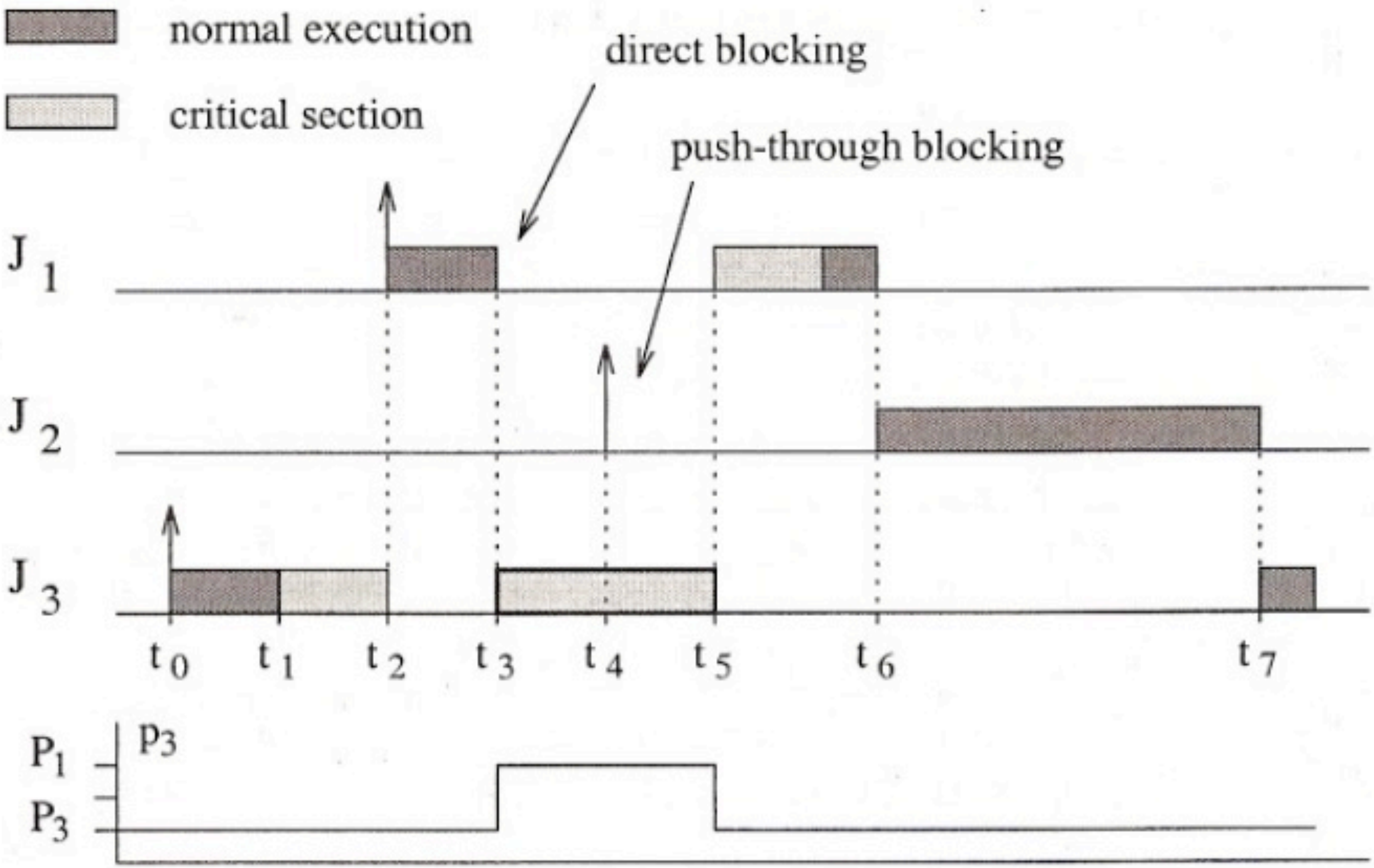
Two Kinds of Blocking



Two Kinds of Blocking

- **Direct blocking.** It occurs when a high-priority job tries to acquire a resource already held by a lower-priority job.
  - Direct blocking is necessary to ensure the consistency of the shared resources .
- **Push-through blocking.** It occurs when a medium-priority job is blocked by a lower-priority job that has inherited a higher priority from a job it directly blocks.
  - Push-through blocking is to avoid unbounded priority inversion.

Priority Inheritance Blocking Time

- Whenever a high priority job wants a resource locked by a lower priority job, the lower priority job gains the priority of the higher priority job as long as the higher priority job is waiting.
- The high priority job only needs to wait for one additional blocking time.
- So, we need to know the maximum time any lower priority job will hold a lock on the resource.

Blocking Properties

- **Lemma 7.1** A semaphore S_k can cause push-through blocking to job *Ji*, only if S_k is accessed both by a job with priority lower than *Pi* and by a job that has or can inherit a priority equal to or higher than *Pi*.
- **Lemma 7.3** If there are *n lower-priority jobs* that can block a job *Ji*, then *Ji* can be blocked for at most the duration of n critical sections (one for each of the n lower priority jobs), regardless of the number of semaphores used by *Ji* ·
  - A job can block another if it is inside a critical section. Once out, it can't enter another since it no longer has a high priority.
- **Lemma 7.4** If there are *m distinct semaphores* that can block a job *Ji*, then *Ji* can be blocked for at most the duration of m critical sections, one for each of the m semaphores.
  - A nested critical section uses the outer most duration since it is the longest.
- **Theorem 7.1 (Sha-Rajkumar-Lehoczky)** Under the Priority Inheritance Protocol, a job J can be blocked for at most the duration of *min*(n, m) critical sections, where n is the number of lower-priority jobs that could block J and m is the number of distinct semaphores that can be used to block J.

Finding the Blocking Time

- For each semaphore S_k we define a *ceiling* $C(S_k)$ to be the priority of the highest priority task that may use it.
- n is the number of lower-priority jobs that could block J, m is the number of distinct semaphores that can be used to block J.

Lemma 7.5 *In the absence of nested critical sections, a critical section $Z_{j,k}$ of J_j guarded by S_k can block J_i only if $P_j < P_i \leq C(S_k)$.*

$$B_i^l = \sum_{j=i+1}^{n} \max_k [D_{j,k} : C(S_k) \geq P_i]$$

$$B_i^s = \sum_{k=1}^{m} \max_{j>i} [D_{j,k} : C(S_k) \geq P_i].$$

Finding the Blocking Time

$$B_i^l = \sum_{j=i+1}^{n} \max_k [D_{j,k} : C(S_k) \geq P_i]$$

$$B_i^s = \sum_{k=1}^{m} \max_{j>i} [D_{j,k} : C(S_k) \geq P_i].$$

1. For each job J_j with priority lower than P_i, identify the set $\beta_{i,j}$ of all critical sections that can block J_i.

2. For each semaphore S_k, identify the set $\gamma_{i,k}$ of all critical sections guarded by S_k that can block J_i.

3. Sum the duration of the longest critical sections in each $\beta_{i,j}$, for any job J_j with priority lower than P_i; let B_i^l be this sum.

4. Sum the duration of the longest critical sections in each $\gamma_{i,k}$, for any semaphore S_k; let B_i^s be this sum.

5. Compute B_i as the minimum between B_i^l and B_i^s.

Calculating Blocking Time

Theorem 7.2 *A set of n periodic tasks using the Priority Inheritance Protocol can be scheduled by the Rate-Monotonic algorithm if*

$$\forall i,\ 1 \le i \le n,\quad \sum_{k=1}^{i} \frac{C_k}{T_k} + \frac{B_i}{T_i} \le i(2^{1/i} - 1). \tag{7.2}$$

	$S_1(P_1)$	$S_2(P_1)$	$S_3(P_2)$
J_1	1	2	0
J_2	0	9	3
J_3	8	7	0
J_4	6	5	4

$$B_i^l = \sum_{j=i+1}^{n} \max_k [D_{j,k} : C(S_k) \ge P_i]$$

$$B_i^s = \sum_{k=1}^{m} \max_{j>i} [D_{j,k} : C(S_k) \ge P_i].$$

$B_1^l = 9 + 8 + 6 = 23$
$B_1^s = 8 + 9 = 17$ ==> $B_1 = 17$

$B_2^l = 8 + 6 = 14$
$B_2^s = 8 + 7 + 4 = 19$ ==> $B_2 = 14$

$B_3^l = 6$
$B_3^s = 6 + 5 + 4 = 15$ ==> $B_3 = 6$

$B_4^l = B_4^s = 0$ ==> $B_4 = 0$

Nested Critical Sections

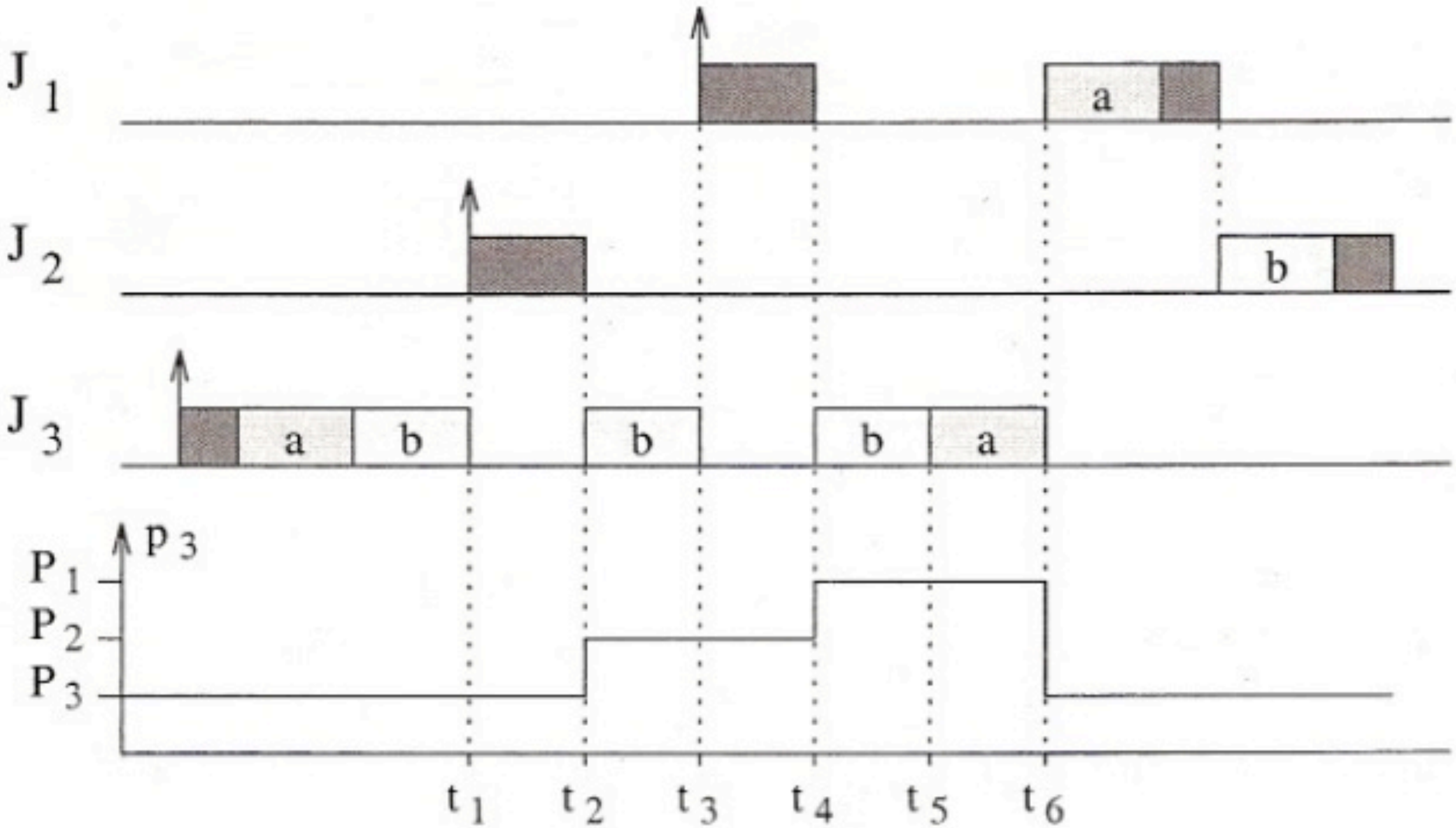


Figure 7.7 Priority inheritance with nested critical sections.

PIP Problems

- If we allow multiple blockings, we need the B_i for each resource.
- PIP does not prevent deadlock

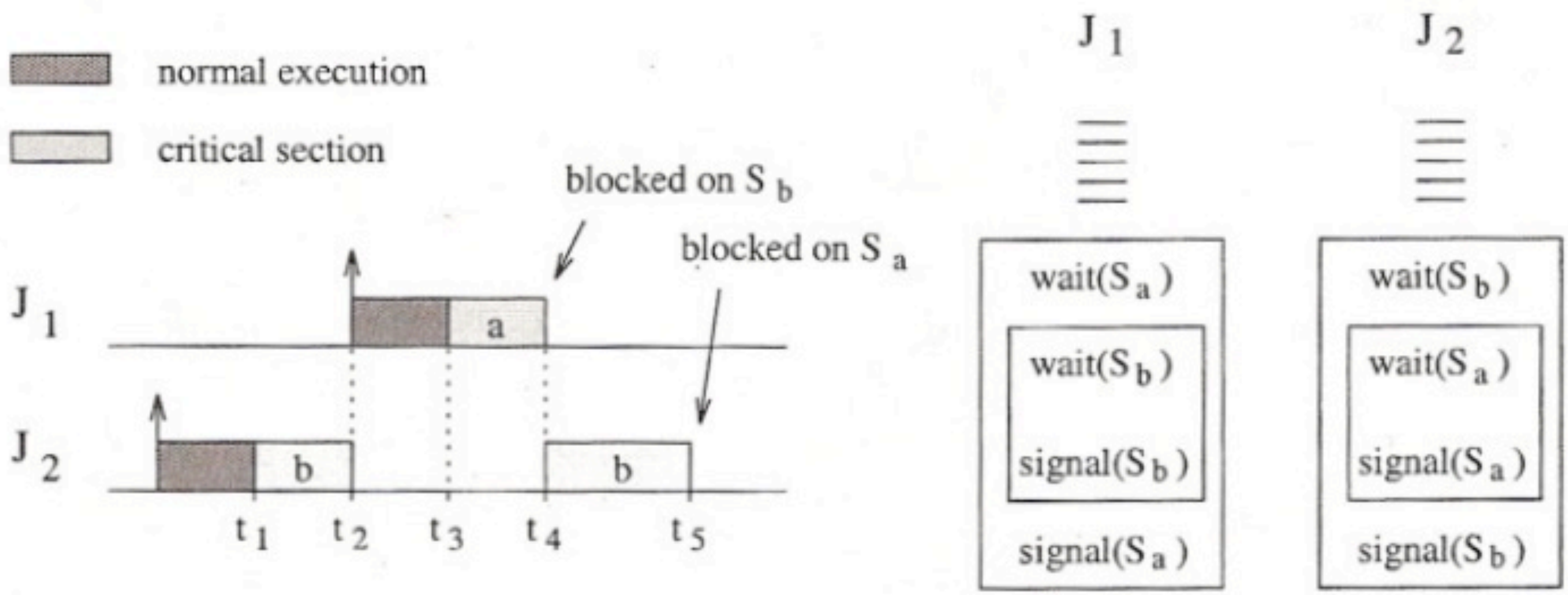


Figure 7.11 Example of deadlock.

Priority Ceiling Protocol

- Fixed priority scheduling
- Uses priority inheritance
- For each resource S_i define its resource priority ceiling $\pi(S_i)$ = max priority of all jobs which may lock S_i
- System Priority Ceiling at t, $\hat{\pi}(t)$ = max $\pi(S_i)$ of all locked resources

PCP Rules

- Suppose J wants to lock a resource R at time t
  - *If* R is held by some other job J_L
    - J is blocked *and* J_L should inherit the priority of J
- If the resource is free,
  - and if J's priority is higher than system ceiling $\hat{\pi}(t)$, then J can lock the resource
  - If J's priority is not higher than $\hat{\pi}(t)$, but if J is holding the resource R_m with $\hat{\pi}(t) = \pi(R_m)$, then J still can lock it
  - Else, block

PCP Example

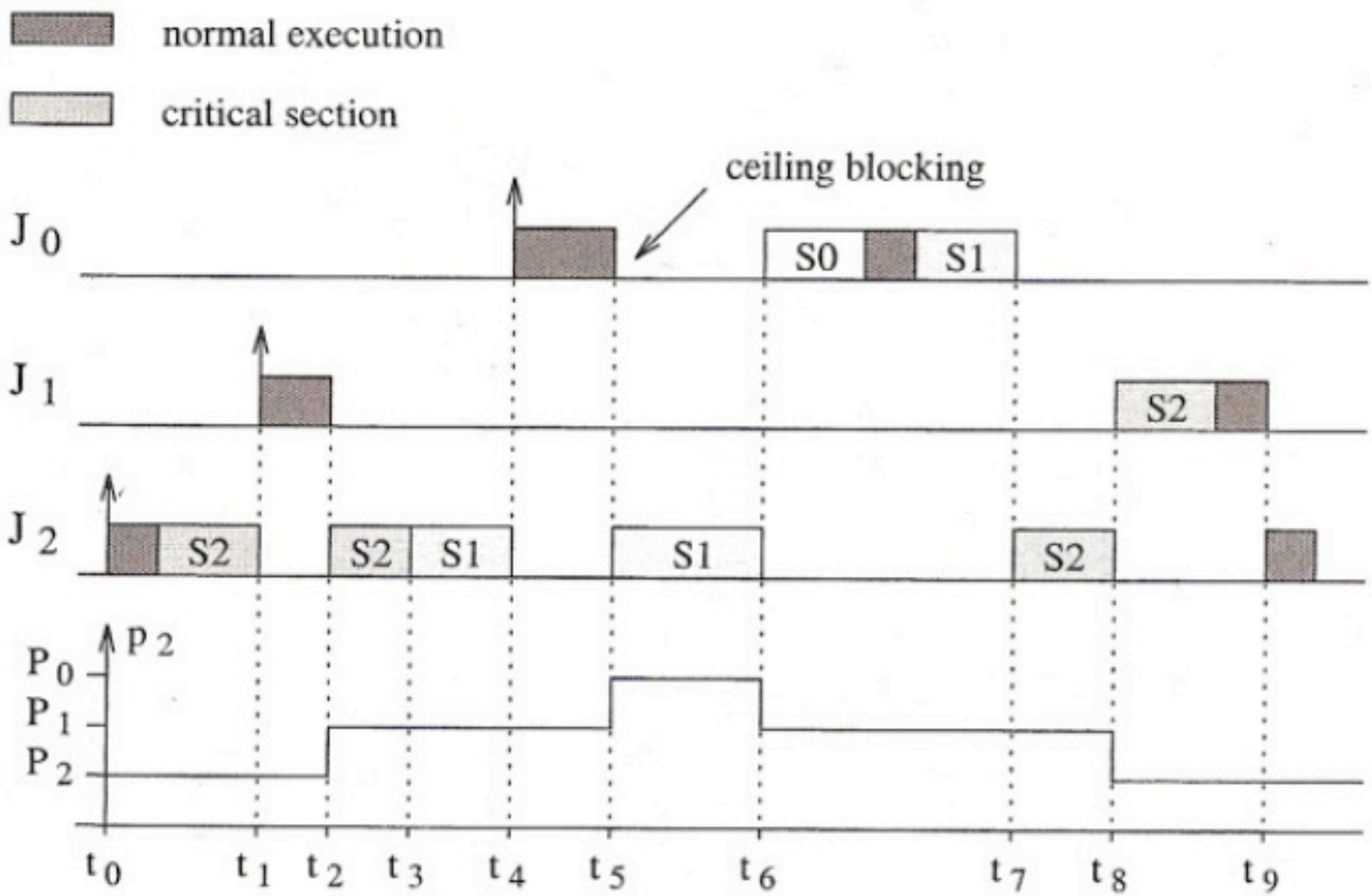
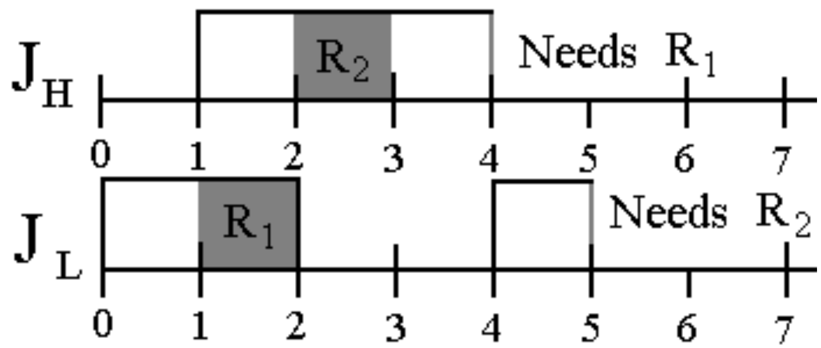


Figure 7.12 Example of Priority Ceiling Protocol.

PCP

- PCP avoids transitive blocking.
  - In the earlier deadlock example, J_H will be blocked since the system priority ceiling will be H when J_L locks R_1



  - Wait for any resource that has a higher priority ceiling than you

PCP Properties

- **Lemma 7.6** *If a job J_k is preempted within a critical section Z_a by a job J_i that enters a critical section Z_b, then, under the Priority Ceiling Protocol, J_k cannot inherit a priority higher than or equal to that of job J_i until J_i completes.*
- The maximum blocking time B_i of job J_i can be computed as the duration of the longest critical section among those belonging to tasks with priority lower than P_i and guarded by a semaphore with ceiling higher than or equal to P_i .
- If $D_{j,k}$ denotes the duration of the longest critical section of task T_i among those guarded by semaphore S_k,

$$B_i = \max_{j,k}\{D_{j,k} \mid P_j < P_i,\ C(S_k) \geq P_i\}. \tag{7.6}$$

PCP Schedulability

- Using the same example as PIP,

Theorem 7.2 *A set of n periodic tasks using the Priority Inheritance Protocol can be scheduled by the Rate-Monotonic algorithm if*

$$\forall i,\ 1 \le i \le n, \quad \sum_{k=1}^{i} \frac{C_k}{T_k} + \frac{B_i}{T_i} \le i(2^{1/i} - 1). \qquad (7.2)$$

	$S_1(P_1)$	$S_2(P_1)$	$S_3(P_2)$
J_1	1	2	0
J_2	0	9	3
J_3	8	7	0
J_4	6	5	4

$$\begin{cases} B_1 = \max(8,6,9,7,5) = 9 \\ B_2 = \max(8,6,7,5,4) = 8 \\ B_3 = \max(6,5,4) = 6 \\ B_4 = 0. \end{cases}$$

PIP vs PCP

- Priority Ceiling Protocol is more restricted than Priority Inheritance Protocol

- To lock a resource:
  - In PIP – Check if a resource is locked
  - In PCP – Check if job priority is higher than the current system resource priority to lock a resource.

PIP vs PCP Blocking Time

- PIP blocking time is 1 per resource or job if you do not allow nesting.
  - Nested: If you want a resource (A) with someone who has it locked, and they lock (B) as well, need to consider everyone who can lock (B)
- PCP blocking time is 1 (the longest critical section).