Inovace tohoto kurzu byla v roce 2011/12 podpořena projektem CZ.2.17/3.1.00/33274 financovaným Evropským sociálním fondem a Magistrátem hl. m. Prahy.

OPP A PRAHA PRAGUE PRAGA PRAG

EVROPSKÁ UNIE

Evropský sociální fond

Praha & EU: Investujeme do vaší budoucnosti

Department of Distributed and Dependable Systems

Embedded and Real-time Systems

Resource Sharing

http://d3s.mff.cuni.cz

Department of Distributed and Dependable Systems

D3S

Tomáš Bureš

<bures@d3s.mff.cuni.cz>

CHARLES UNIVERSITY IN PRAGUE

Faculty of Mathematics and Physics

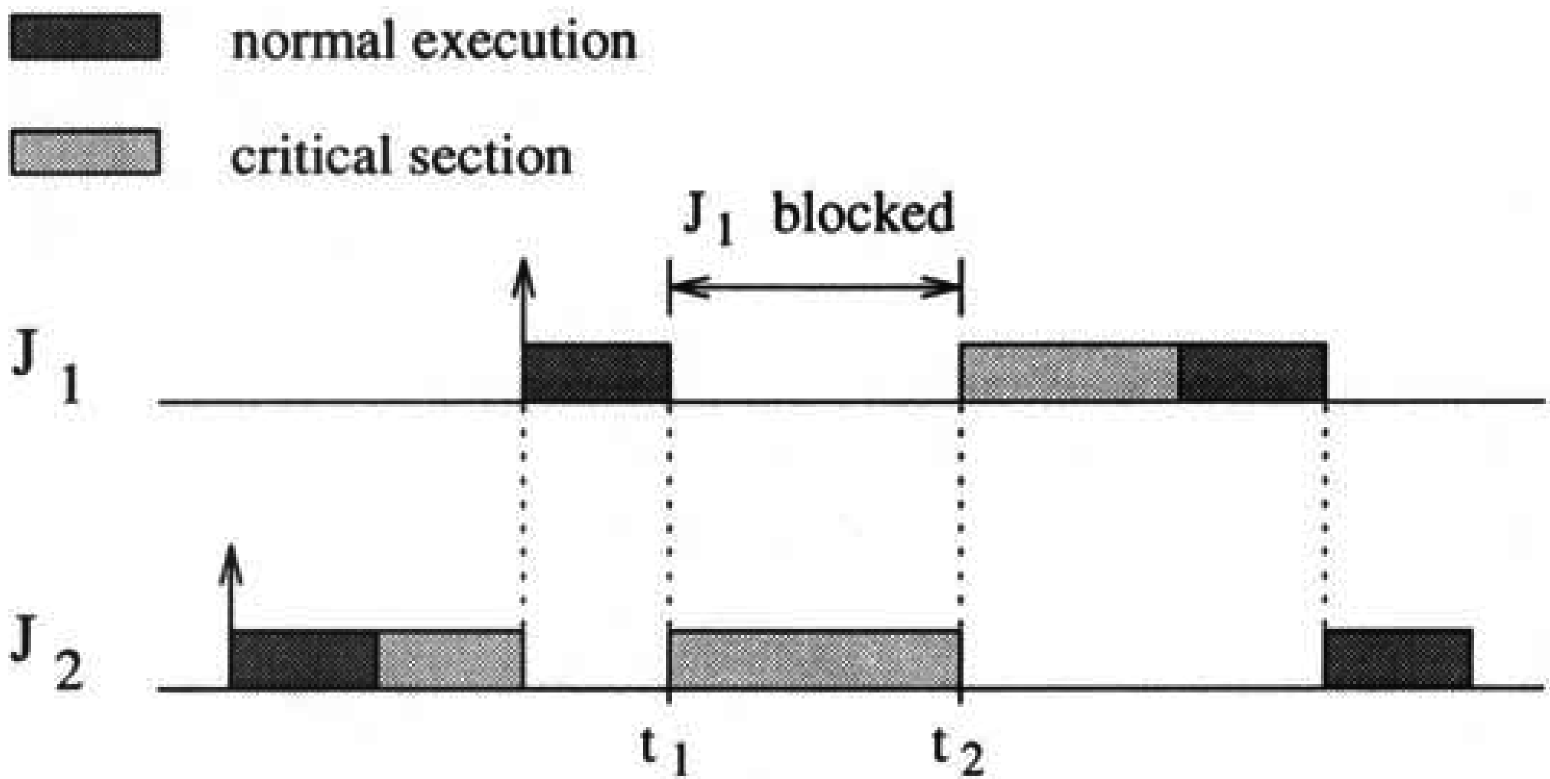
Resource Sharing

- Mutual exclusion
  - Access to a resource is limited to one task at a time
- Critical section
  - a code section that should be executed mutually exclusively by tasks
- Semaphore
  - a data structure used for protection of critical sections

- R - shared resource
- S - semaphore

```
while (1) {
   ...
   lock(S);
   ...
   access(R);
   ...
   unlock(S);
   ...
}
```

Blocking on an Exclusive Resource



Priority Inversion

- If a high priority task waits for a lock kept by a low priority task and an unrelated medium priority task interferes
  - The waiting time of the high priority task is unbounded

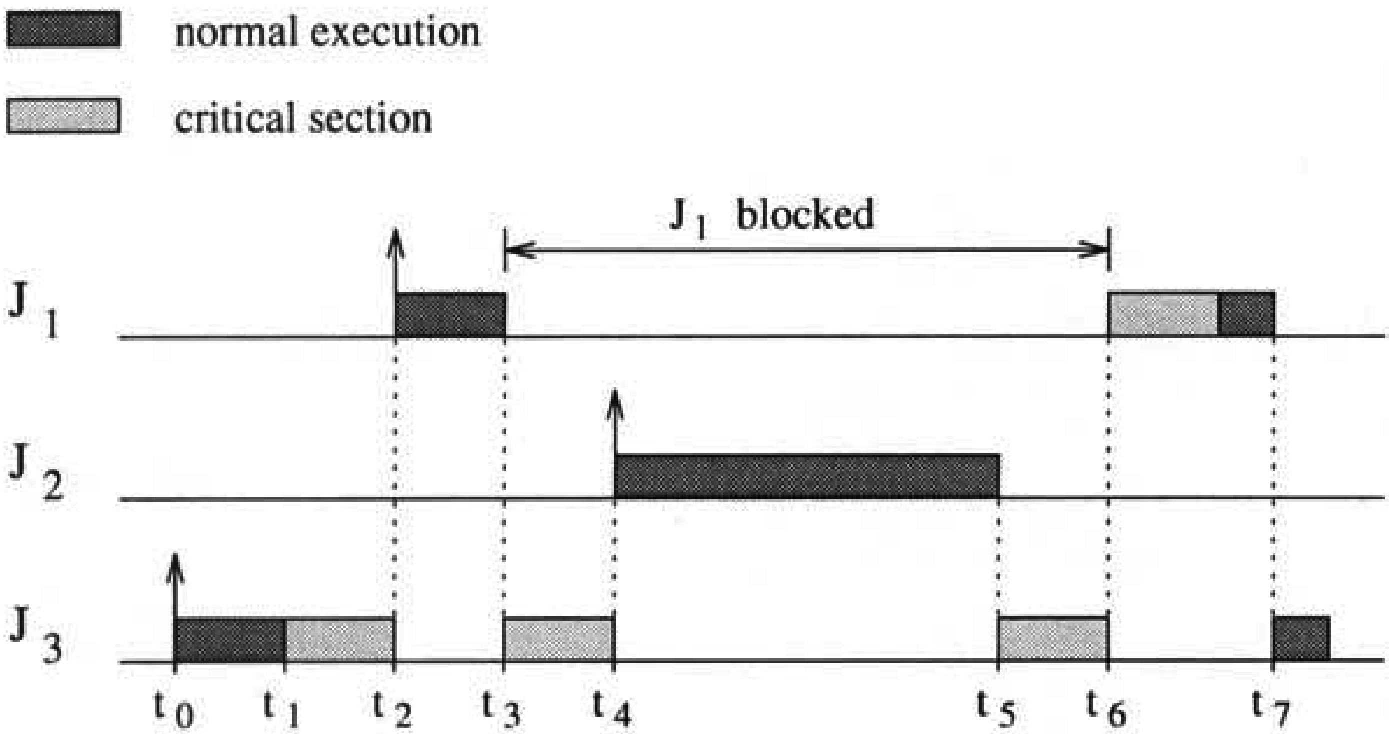
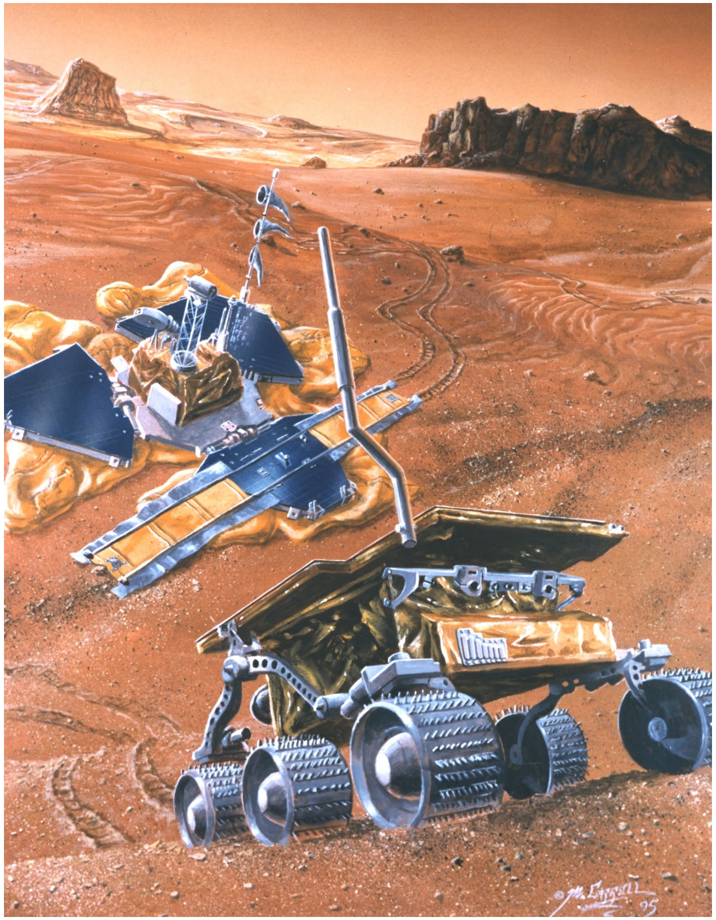


Figure from Buttazzo, G.:Hard Real-Time Computing Systems

Real Case

- Mars Pathfinder
  - experienced infrequent resets
  - high-priority information bus task
  - low-priority meteorological data gathering task
    - shared date with the high-priority information bus task
  - medium-priority communication task
  - after some time a watchdog noticed that information bus task is not running – initiated a reset



Department of Distributed and Dependable Systems

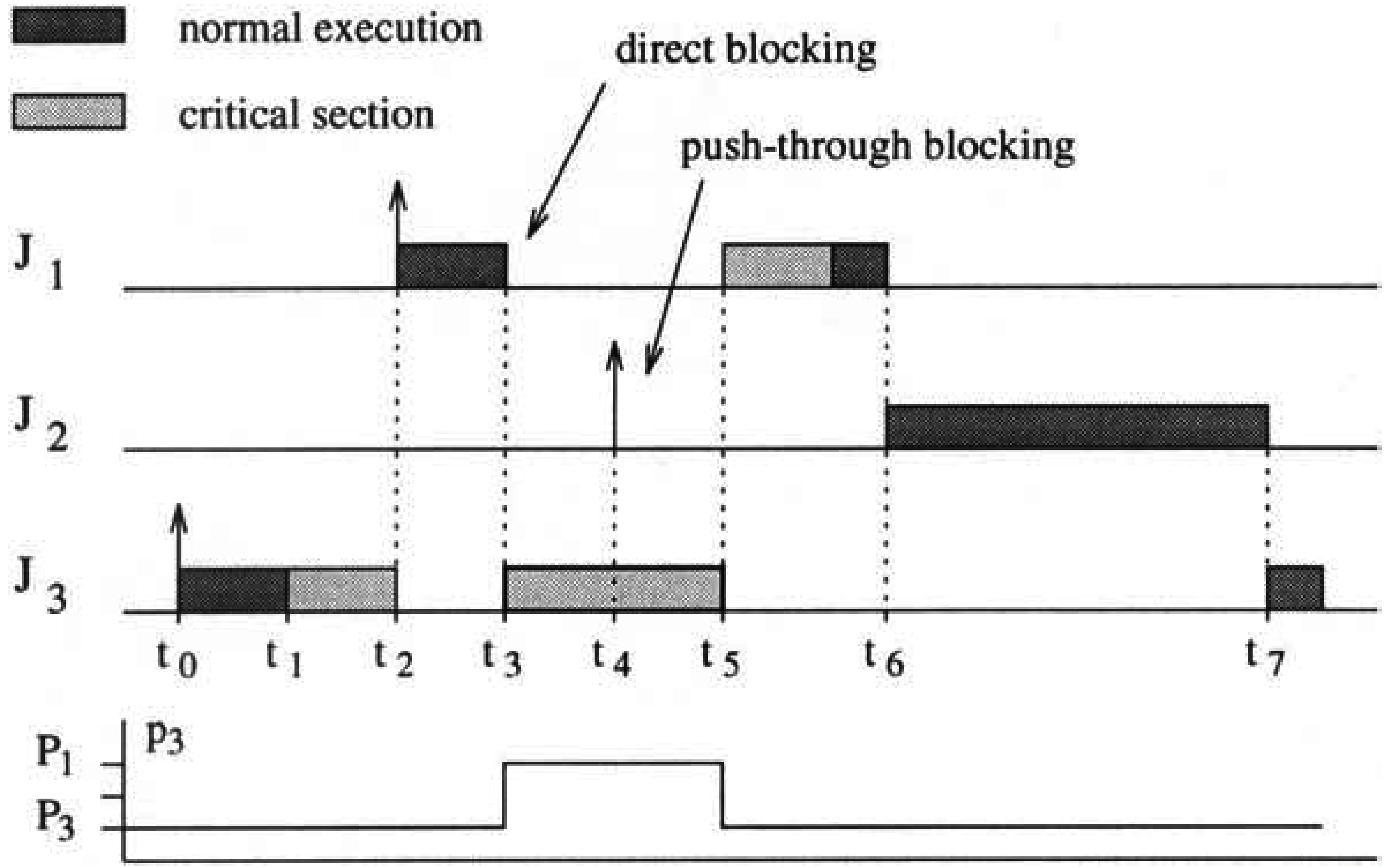
Priority Inheritance Protocol

- Assumptions
  - Jobs have a fixed nominal priority and an active priority
  - Critical sections are properly nested
  - Critical sections are guarded by binary semaphores
  - Jobs are scheduled based on their active priorities. Jobs with the same priority are executed on First Come First Served basis.

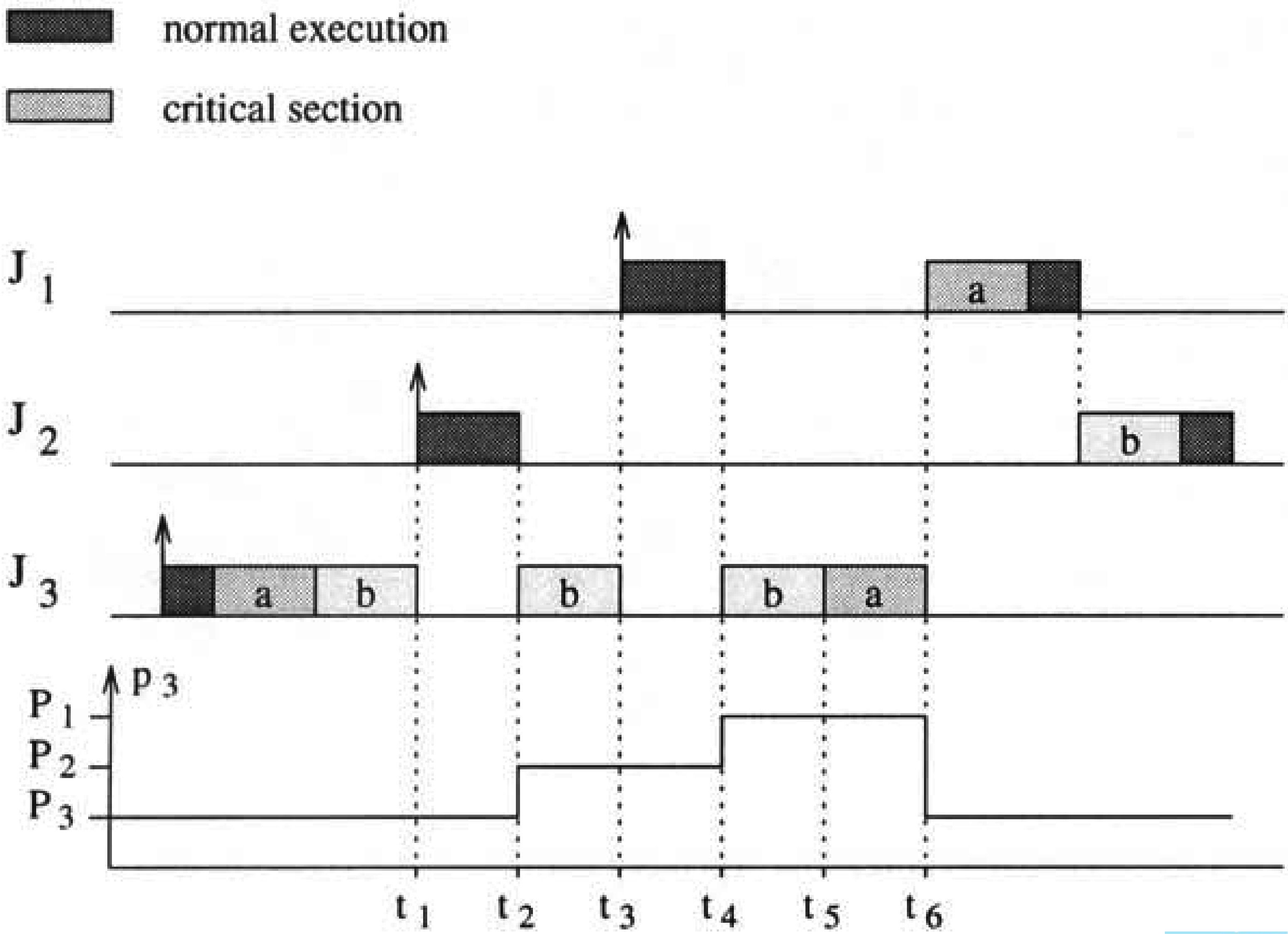
Priority Inheritance Protocol

- When job J_i tries to enter a critical section $Z_{i,j}$ and resource $R_{i,j}$ is already held by a lower-priority job, J_i will be blocked. Otherwise, J_i enters the critical section.
- When a job J_i is blocked on a semaphore, it transmits its active priority to the job, say J_k, that holds that semaphore. Hence, J_k resumes and executes the rest of its critical section with a priority $p_k = p_i$.
- When J_k exits a critical section, it unlocks the semaphore, and the highest-priority job, if any, blocked on that semaphore is awakened. Moreover, the active priority of J_k is updated as follows: if no other jobs are blocked by J_k, p_k is set to its nominal priority P_k, otherwise it is set to the highest priority of the jobs blocked by J_k.
- Priority inheritance is transitive.

Priority Inheritance Protocol



Nested Critical Sections



Transitive Priority Inheritance

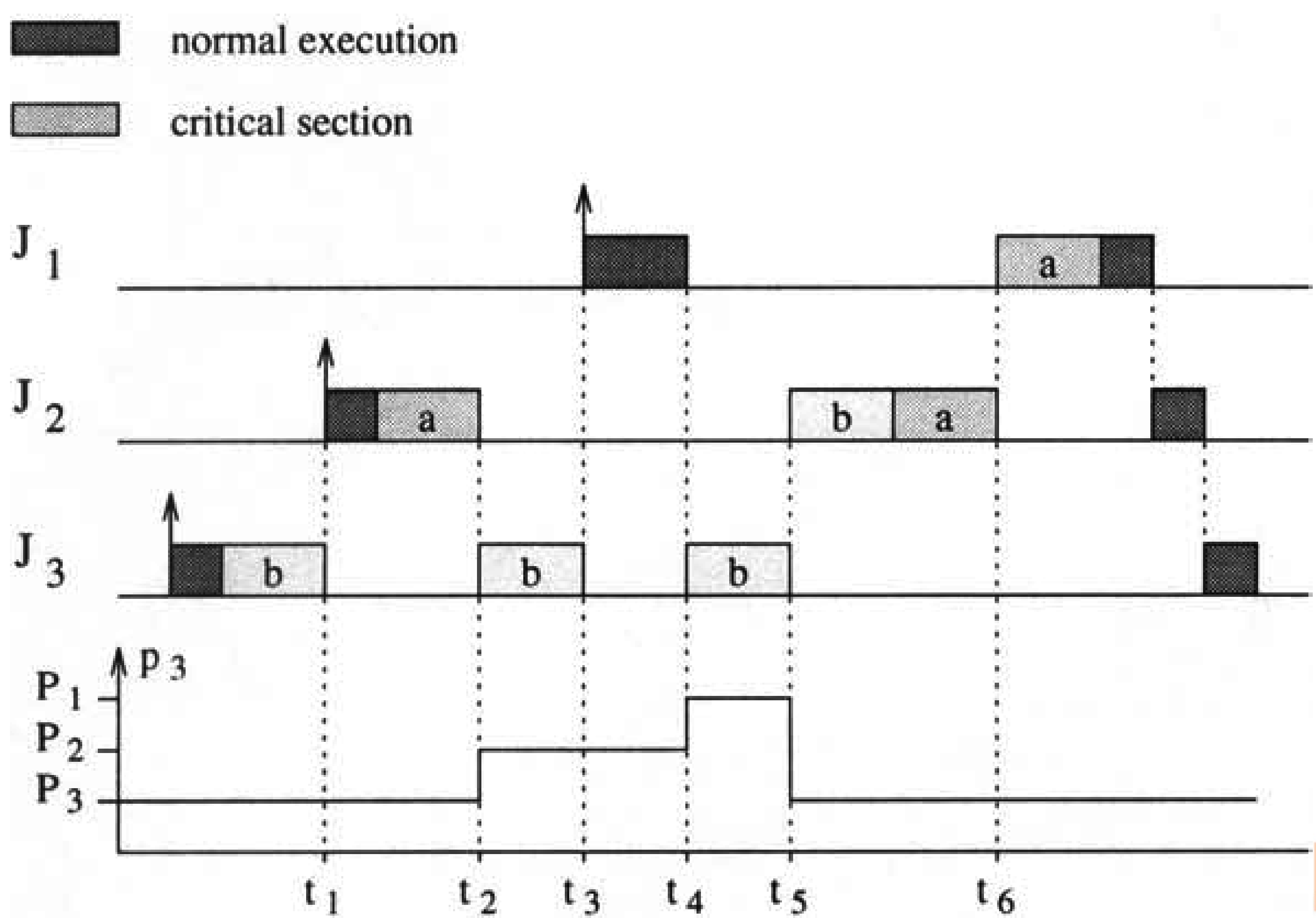


Figure from Buttazzo, G.:Hard Real-Time Computing Systems

Properties of PIP

- If there are n lower-priority jobs that can block a job J_i, then J_i 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 J_i.
- If there are m distinct semaphores that can block a job J_i, then J_i can be blocked for at most the duration of m critical sections, one for each of the m semaphores.
- 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.

Schedulability Analysis of PIP

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

$$\forall i, 1 \leq i \leq n\colon \sum_{k=1}^{i} \frac{C_k}{T_k} + \frac{B_i}{T_i} \leq i\left(2^{1/i} - 1\right)$$

- The schedulability test based on response times can be also performed similarly, however it is no longer a necessary condition.

$$R_i = C_i + B_i + \sum_{j=1}^{i-1} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

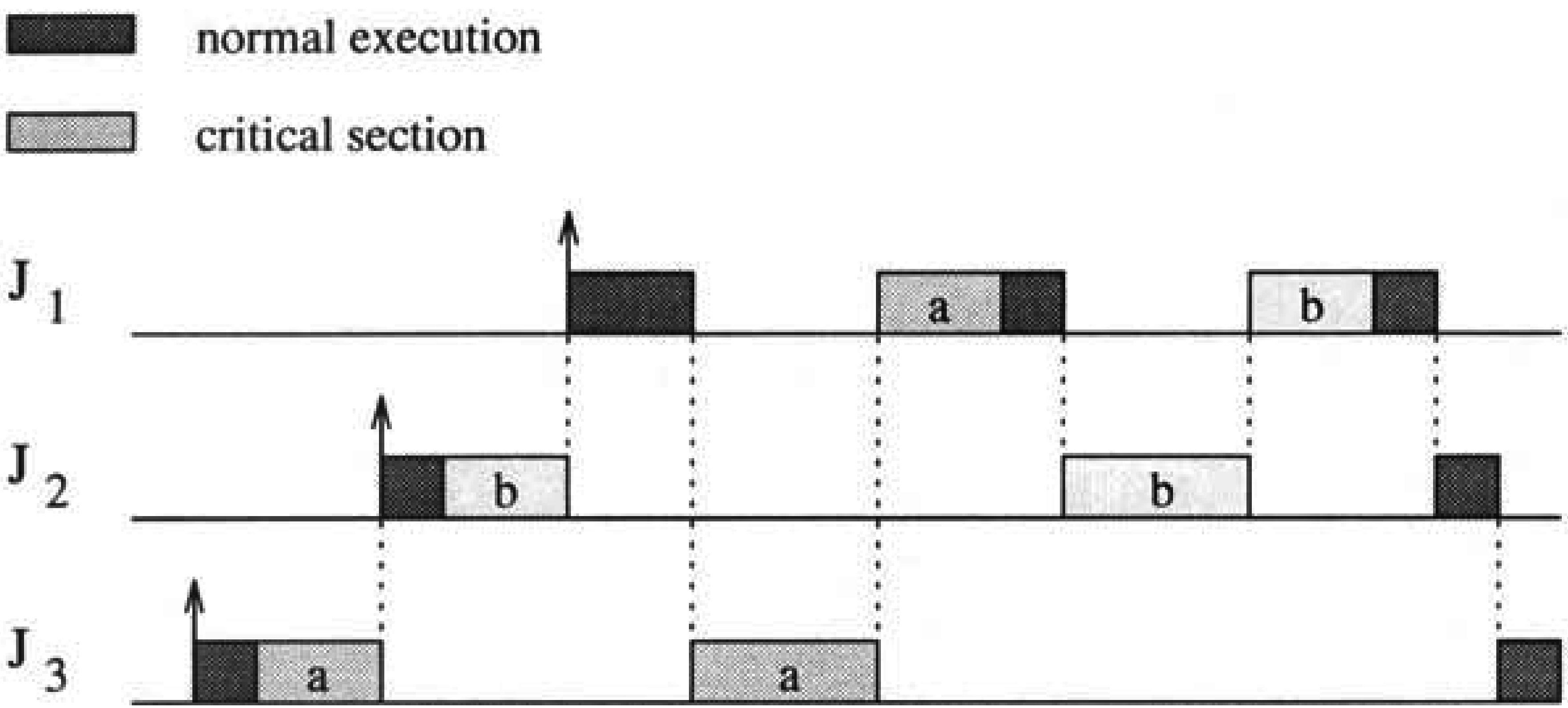
Department of Distributed and Dependable Systems

Blocking Time Computation

- Computed as the minimum of these two:
  - Sum of the durations of the longest critical section for any job with lower priority that can block our task
  - Sum of the durations of the longest critical section for any semaphore on which can our task can wait

Problems of PIP

- Chained blocking



Problems of PIP

- Deadlock

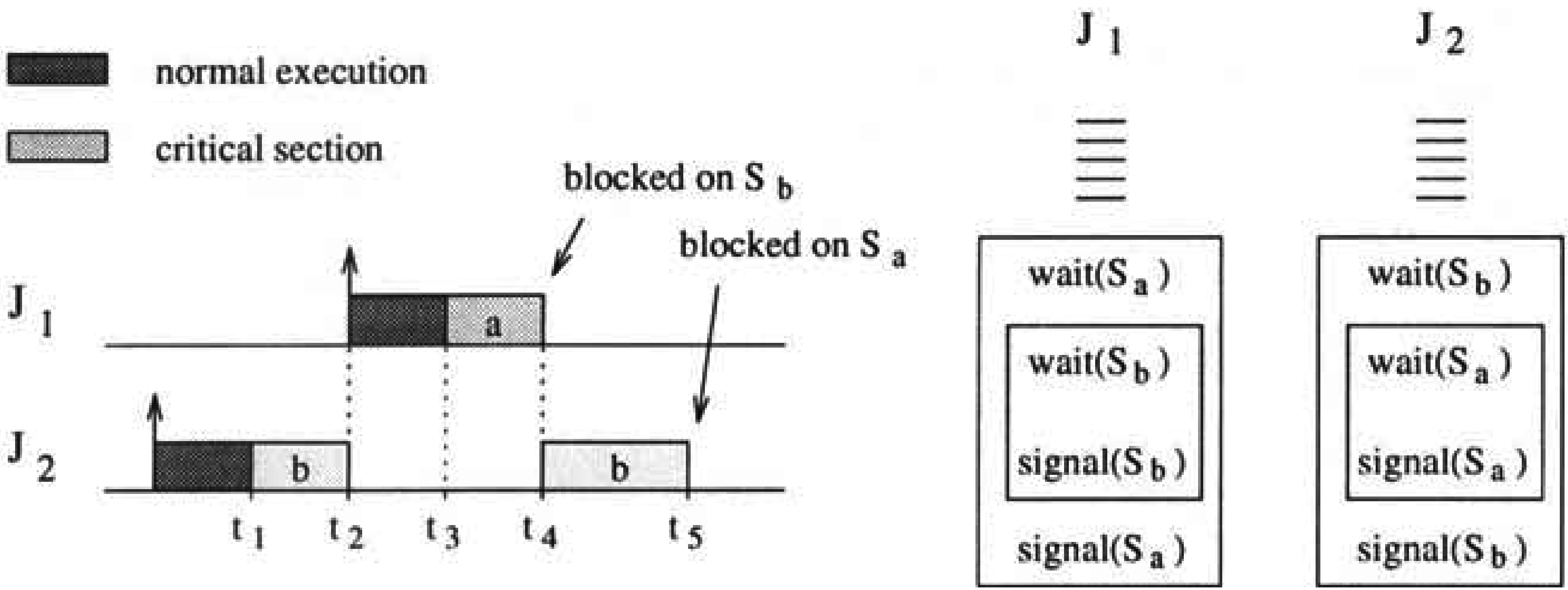


Figure from Buttazzo, G.:Hard Real-Time Computing Systems

Priority Ceiling Protocol

- Each semaphore S_k is assigned a priority ceiling $C(S_k)$ equal to the priority of the highest-priority job, that can lock it. (It is a static values that can be computed off-line.)
- S^* is the semaphore with the highest priority ceiling among all the semaphores currently locked by jobs other than J_i. $C(S^*)$ is its ceiling.
- To enter a critical section guarded by semaphore S_k, J_i must have a priority higher than $C(S^*)$.
  If $P_i \leq C(S^*)$, the lock on S_k is denied and J_i is blocked on semaphore S^* by the job that has a lock on it.
- The rest is same as in PIP.

PCP – Example

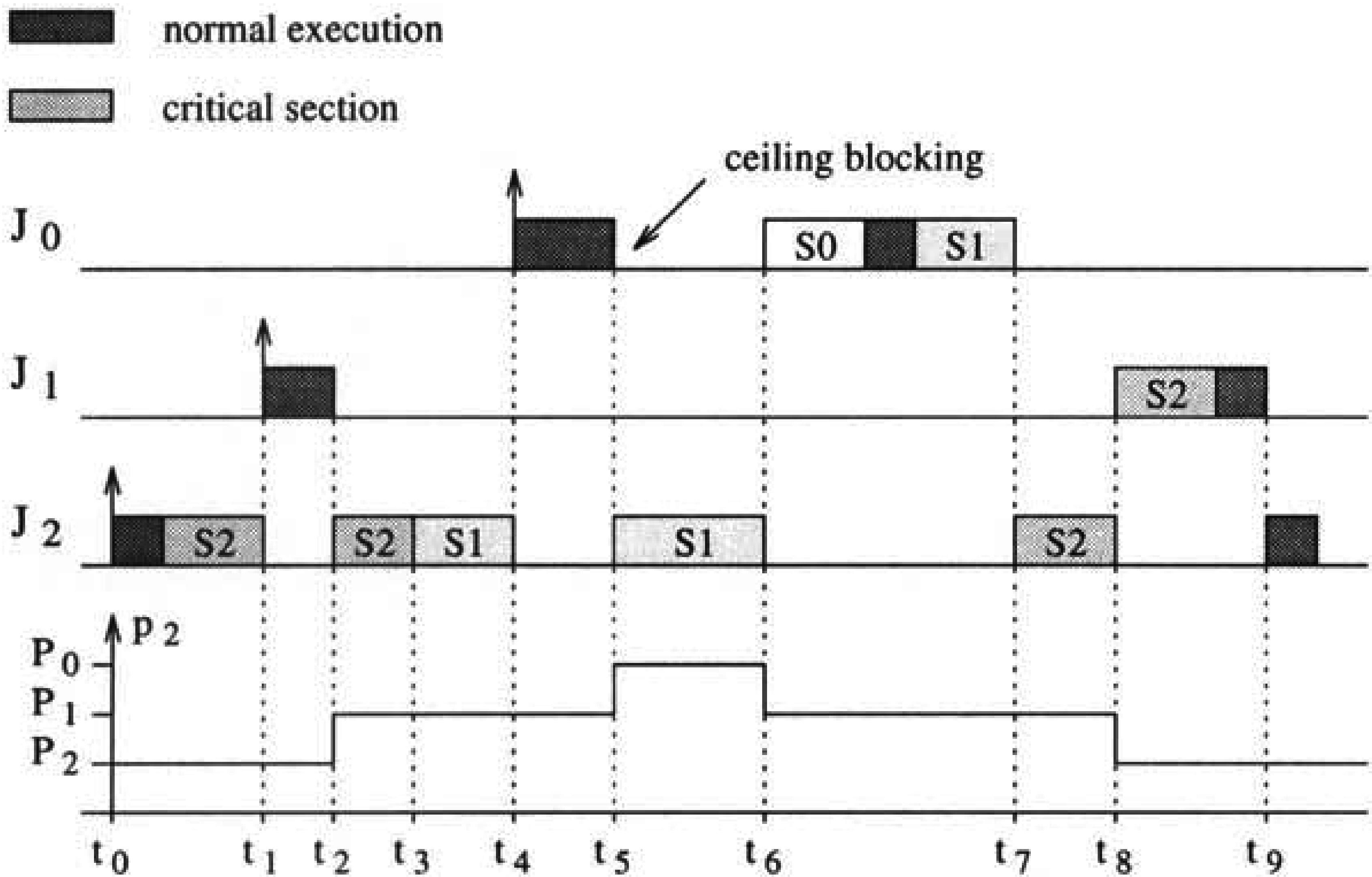


Figure from Buttazzo, G.:Hard Real-Time Computing Systems

Properties of PCP

- *PCP prevents transitive blocking*
  - *i.e.* J_k *blocks* J_j *and* J_j *blocks* J_i
  - *It would mean that* J_k *locks* S_a*, then comes* J_j *locks* S_b *and then* S_a*, then comes* J_i *and locks* S_b
  - *This cannot happen as* J_j *would be blocked at the time it tries to lock* S_b

- *PCP prevents deadlocks*
  - *Forming a cycle among tasks is not possible due to the ceiling*

Properties of PCP

- Under the Priority Ceiling Protocol, a job J_i can be blocked for at most the duration of one critical section.
  - Suppose that J_i is blocked by two lower priority jobs J_j and J_k, where $P_k < P_j < P_i$. Let J_k enter its blocking critical section first, and let C_k^* be the highest-priority ceiling among all the semaphores locked by J_k. In this situation, if job J_j enters its critical section we must have that $P_j > C_k^*$. This means that $P_j > C_k^* \geq P_i$. This contradicts the assumption that $P_i > P_k$.

Schedulability Analysis

- Same as in PIP, only the blocking time is computed differently

$$\forall i, 1 \leq i \leq n: \sum_{k=1}^{i} \frac{C_k}{T_k} + \frac{B_i}{T_i} \leq i\left(2^{1/i} - 1\right)$$

or

$$R_i = C_i + B_i + \sum_{j=1}^{i-1} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

Department of Distributed and Dependable Systems

Blocking Time Computation

- A job can be blocked at most for the duration of the longest critical section among those that can block it.
- **Lemma:** Under PCP, a critical section $Z_{j,k}$ can block a job J_i only if $P_j < P_i$ and $C(S_k) \geq P_i$.
- Using the lemma, we can compute the maximum blocking time as:

$$B_i = \max_{j,k}\{D_{j,k} | P_j < P_i, C(S_k) \geq P_i\}$$

Department of Distributed and Dependable Systems

Example

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

- $B_1 = \max(8,6,9,7,5) = 9$
- $B_2 = \max(8,6,7,5,4) = 8$
- $B_3 = \max(6,5,4) = 6$
- $B_4 = 0$

Immediate Ceiling Priority Protocol

- The same assumptions as for PCP, but…
  - In PCP, the priority is raised when a higher priority task is blocked
  - In ICPC, when a task locks a semaphore it immediately raises its own priority to the ceiling of the semaphore

- As a consequence, a task will only suffer a block at the very beginning of its execution
  - Once the task starts actually executing, all the resources it needs must be free; if they were not, then some tasks would have an equal or higher priority and the tasks's execution would be postponed

Immediate Ceiling Priority Protocol

- In fact, a task doesn't have to really lock or unlock the semaphore, it just has to get the priority
- ICPP is less complex then PCP and has fewer context switches
- PCP gives better concurrency
  - Doesn't block medium priority task which doesn't lock
- The worst-case timing performance of ICPP is the same as PCP
  - Thus the response time analysis for PCP may be used

Department of Distributed and Dependable Systems