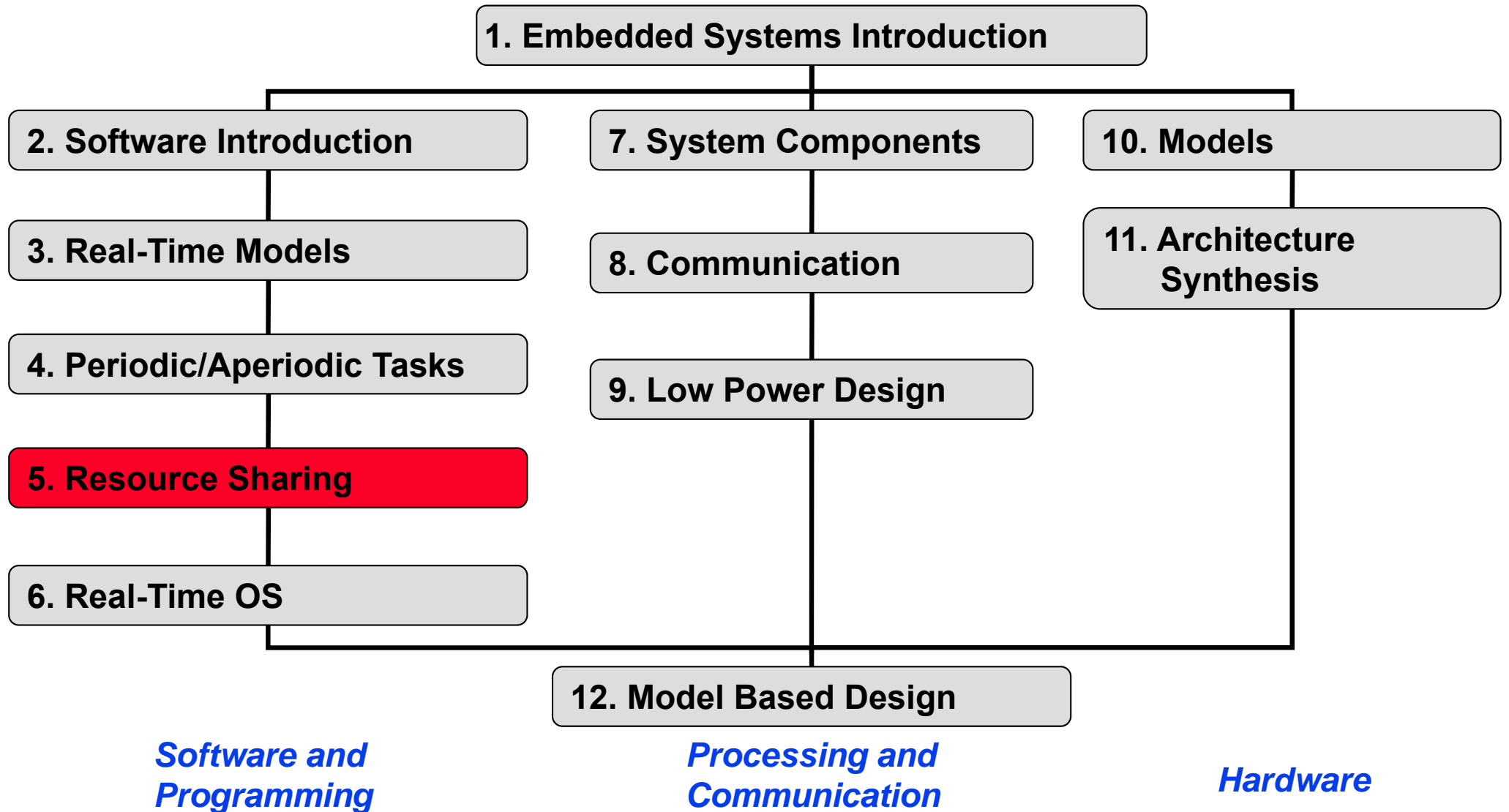


Embedded Systems

5. Resource Sharing

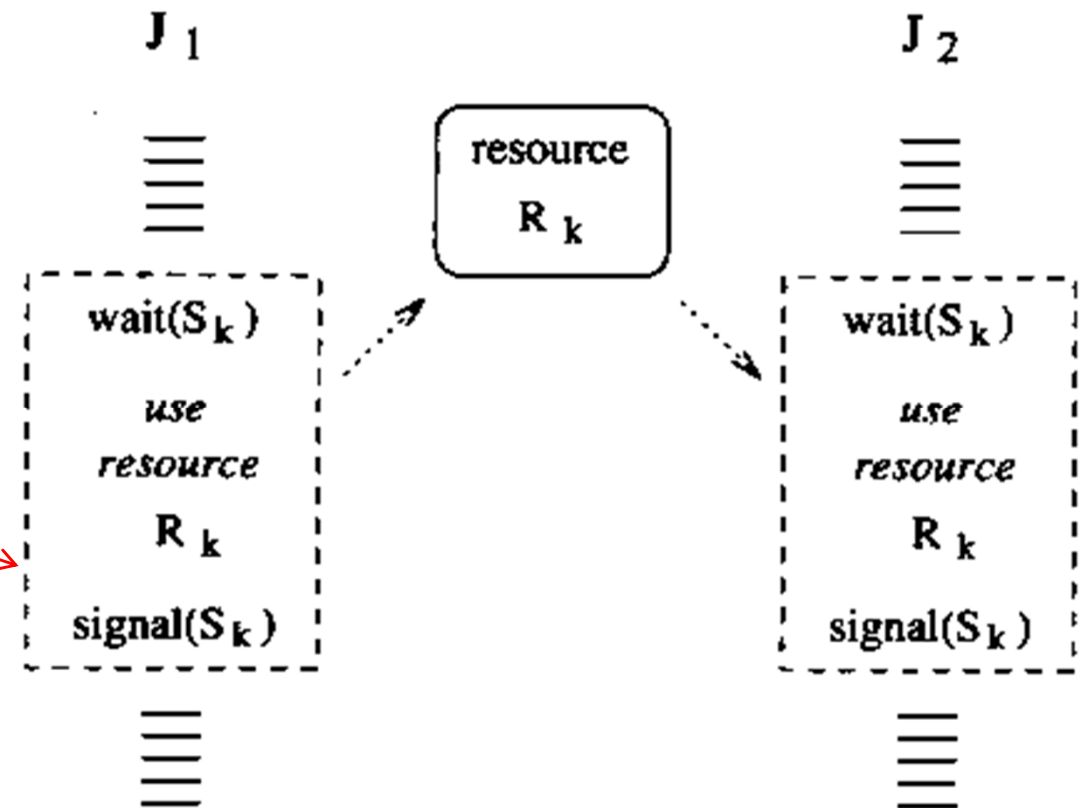
Lothar Thiele

Contents of Course



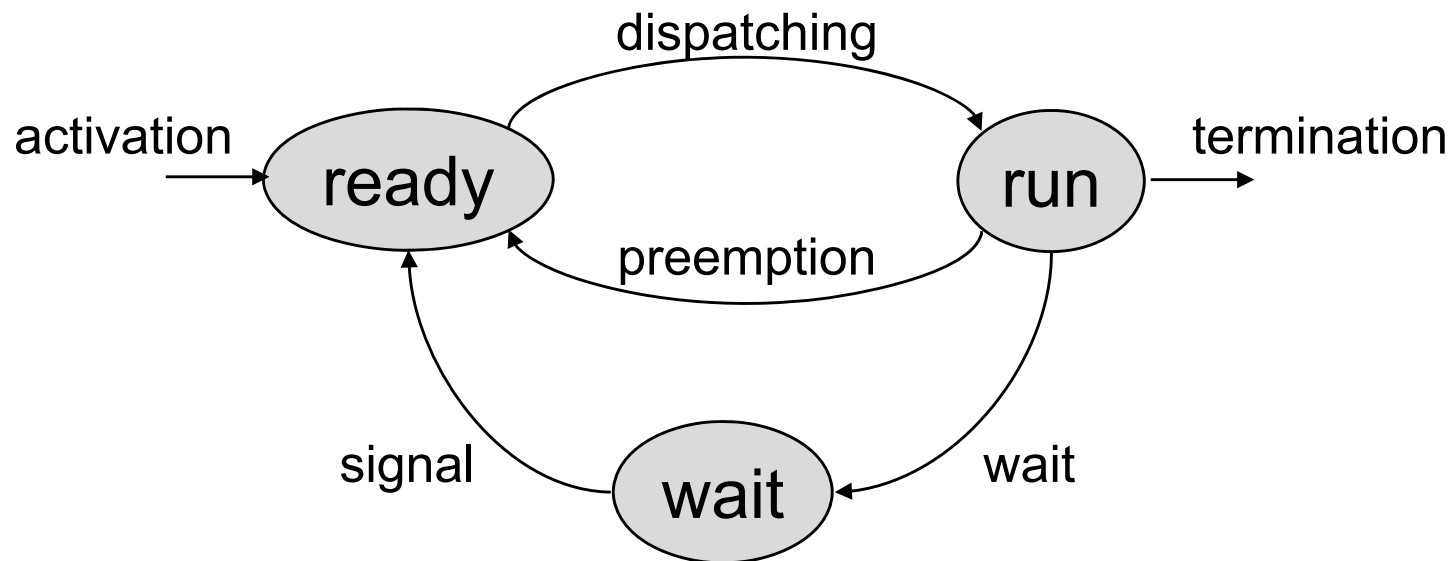
Resource Sharing

- ▶ **Examples** of common resources: data structures, variables, main memory area, file, set of registers, I/O unit,
- ▶ Many shared resources do not allow simultaneous accesses but require **mutual exclusion** (**exclusive resources**). A piece of code executed under mutual exclusion constraints is called a **critical section**.



Terms

- ▶ A task waiting for an exclusive resource is said to be **blocked** on that resource. Otherwise, it proceeds by entering the **critical section** and **holds** the resource. When a task leaves a critical section, the associated resource becomes **free**.
- ▶ Waiting state caused by resource constraints:





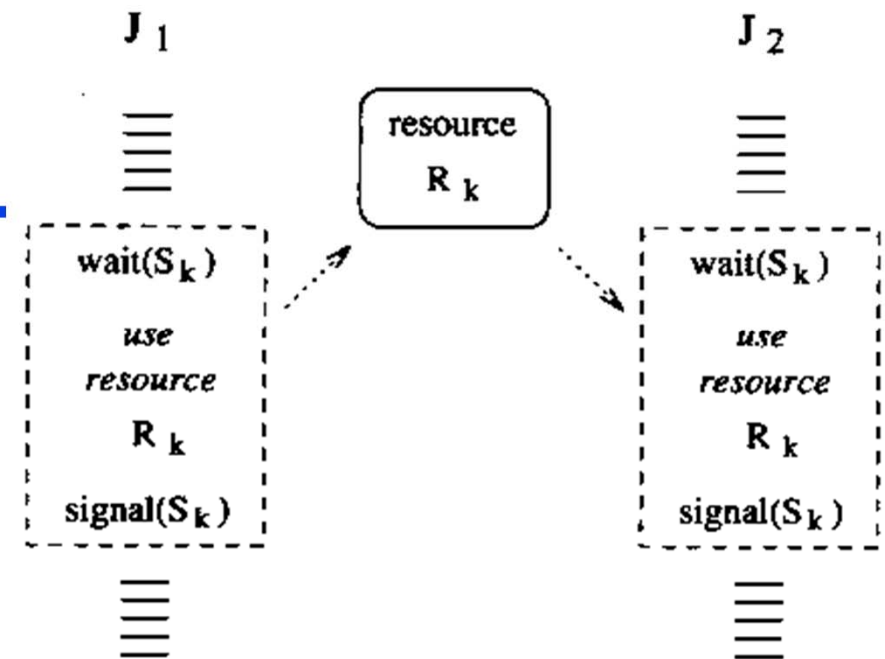
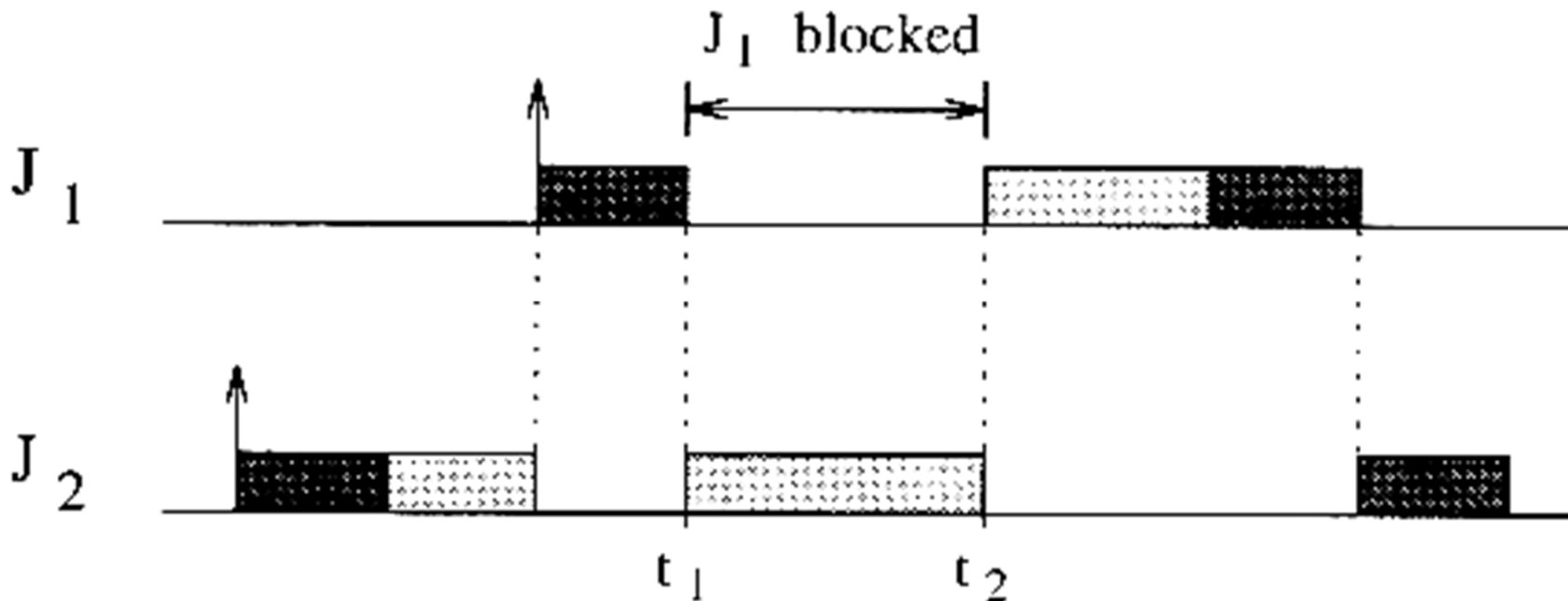
Terms

- ▶ Each **exclusive resource** R_i must be protected by a different **semaphore** S_i and each critical section operating on a resource must begin with a $wait(S_i)$ primitive and end with a $signal(S_i)$ primitive.
- ▶ All tasks blocked on the same resource are kept in a queue associated with the semaphore. When a running task executes a **wait** on a **locked semaphore**, it enters a **waiting state**, until another task executes a **signal** primitive that **unlocks the semaphore**.

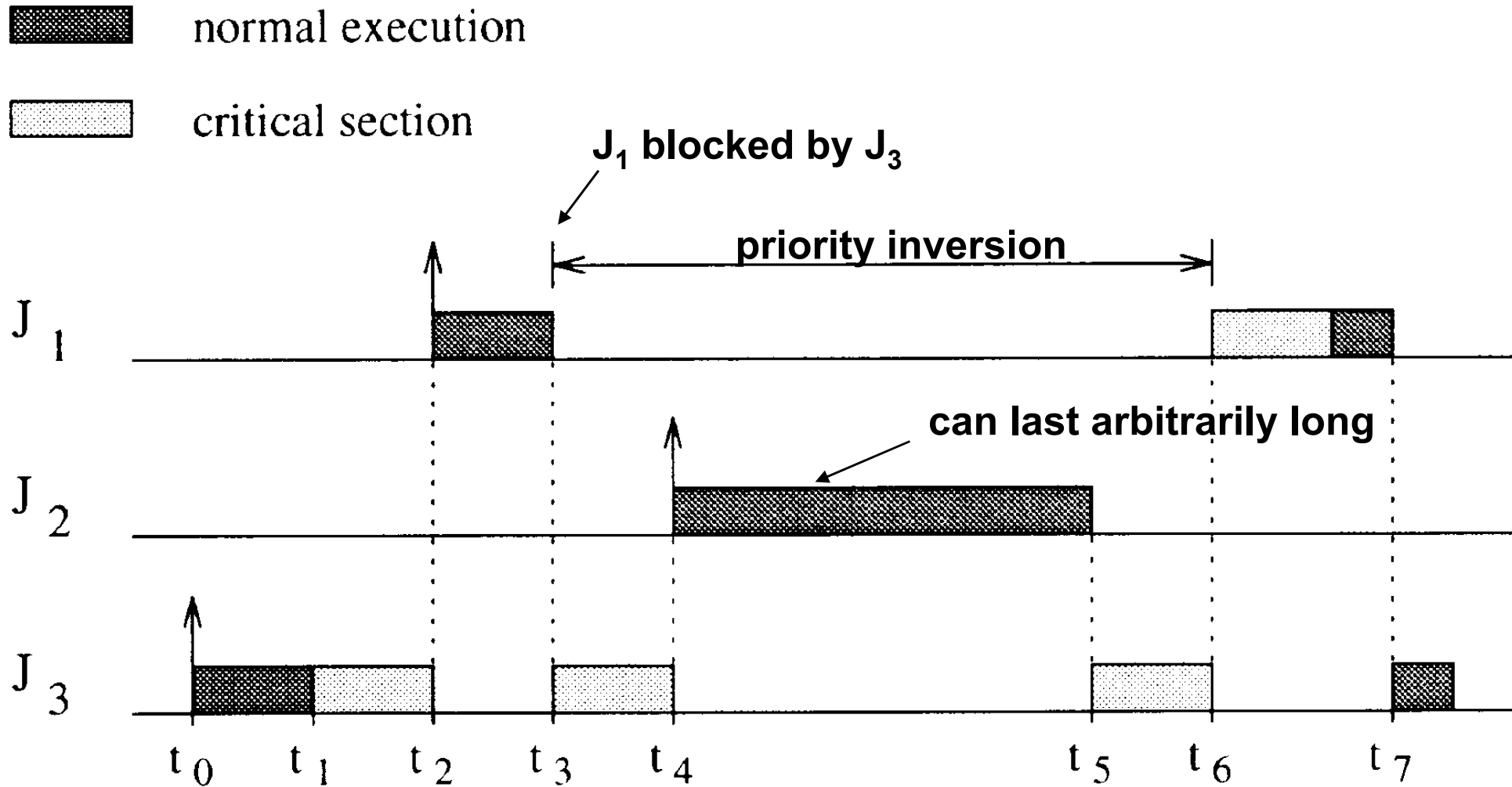
Priority Inversion (1)

► Unavoidable blocking

 normal execution
 critical section



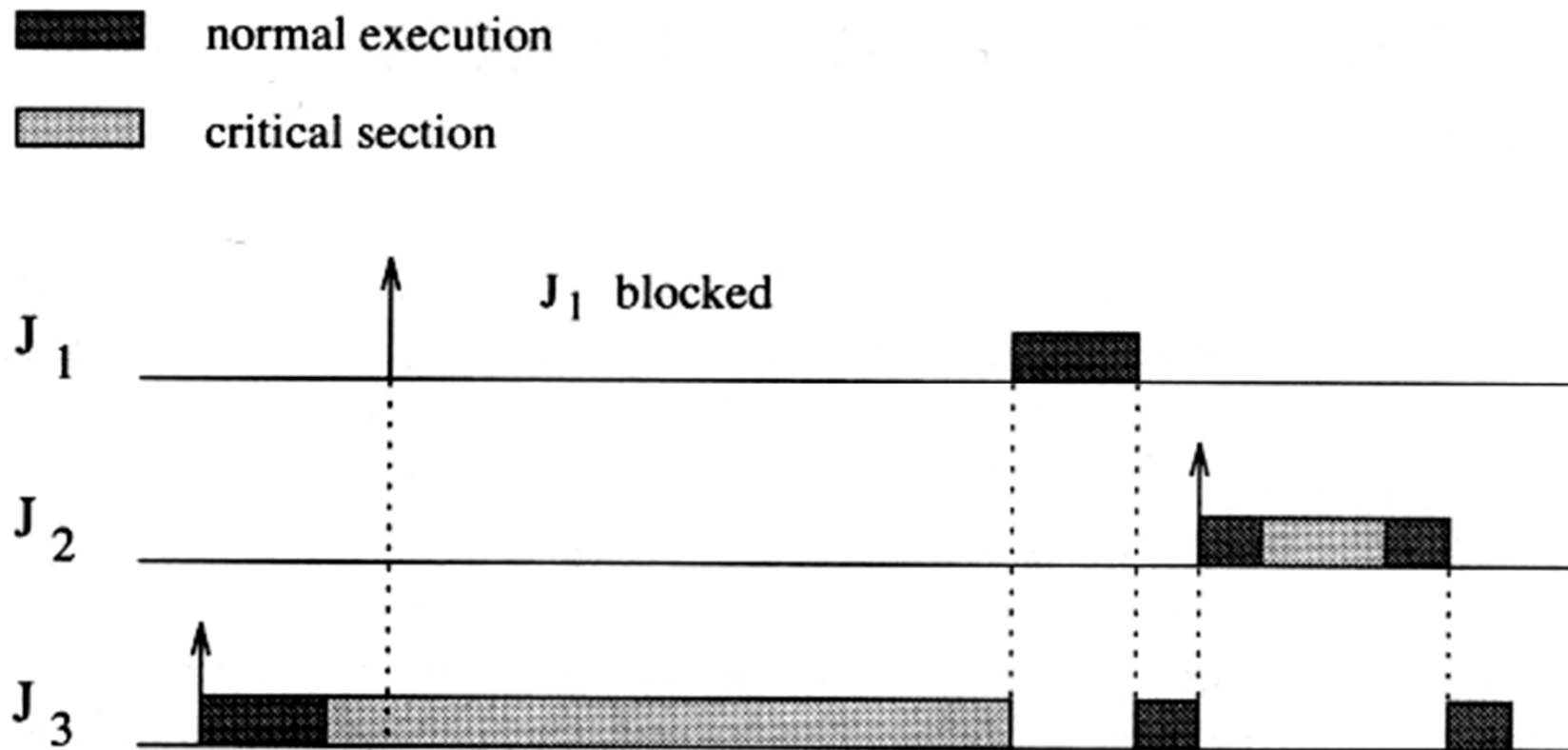
Priority Inversion (2)



[But97, S.184]

Solutions

- ▶ **Disallow preemption** during the execution of all critical sections. Simple, but creates unnecessary blocking as unrelated tasks may be blocked.



Resource Access Protocols

- ▶ **Basic idea:** Modify the priority of those tasks that cause blocking. When a task J_i blocks one or more higher priority tasks, it temporarily assumes a higher priority.
- ▶ **Methods:**
 - **Priority Inheritance Protocol** (PIP), for static priorities
 - Priority Ceiling Protocol (PCP), for static priorities
 - Stack Resource Policy (SRP),
for static and dynamic priorities
 - others ...

Priority Inheritance Protocol (PIP)

- **Assumptions:**

n tasks which cooperate through m shared resources; fixed priorities, all critical sections on a resource begin with a $wait(S_i)$ and end with a $signal(S_i)$ operation.

- **Basic idea:**

When a task J_i blocks one or more higher priority tasks, it temporarily assumes (inherits) the highest priority of the blocked tasks.

- **Terms:**

We distinguish a fixed **nominal priority** P_i and an **active priority** p_i larger or equal to P_i . Jobs J_1, \dots, J_n are ordered with respect to nominal priority where J_1 has **highest priority**. Jobs do not suspend themselves.

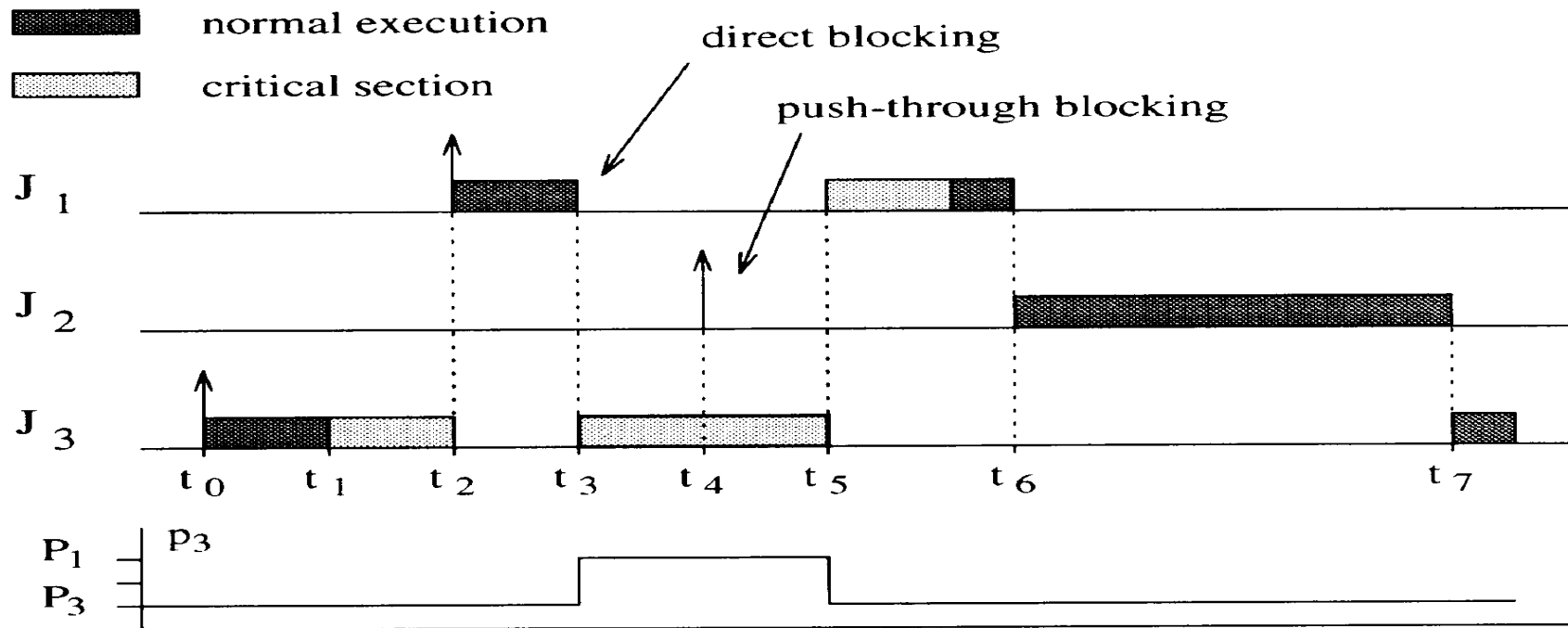
Priority Inheritance Protocol (PIP)

► *Algorithm:*

- Jobs are scheduled based on their **active priorities**. Jobs with the same priority are executed in a FCFS discipline.
- When a job J_i tries to **enter a critical section** and the resource is blocked by a lower priority job, the job J_i is blocked. Otherwise it enters the critical section.
- When a job J_i is **blocked**, it transmits its active priority to the job J_k that holds the semaphore. J_k resumes and executes the rest of its critical section with a priority $p_k = p_i$ (it **inherits** the priority of the highest priority of the jobs blocked by it).
- When J_k exits a critical section, it **unlocks** the semaphore and the highest priority job blocked on that semaphore is awakened. If no other jobs are blocked by J_k , then p_k is set to P_k , otherwise it is set to the highest priority of the jobs blocked by J_k .
- Priority inheritance is **transitive**, i.e. if 1 is blocked by 2 and 2 is blocked by 3, then 3 inherits the priority of 1 via 2.

Priority Inheritance Protocol (PIP)

► Example:

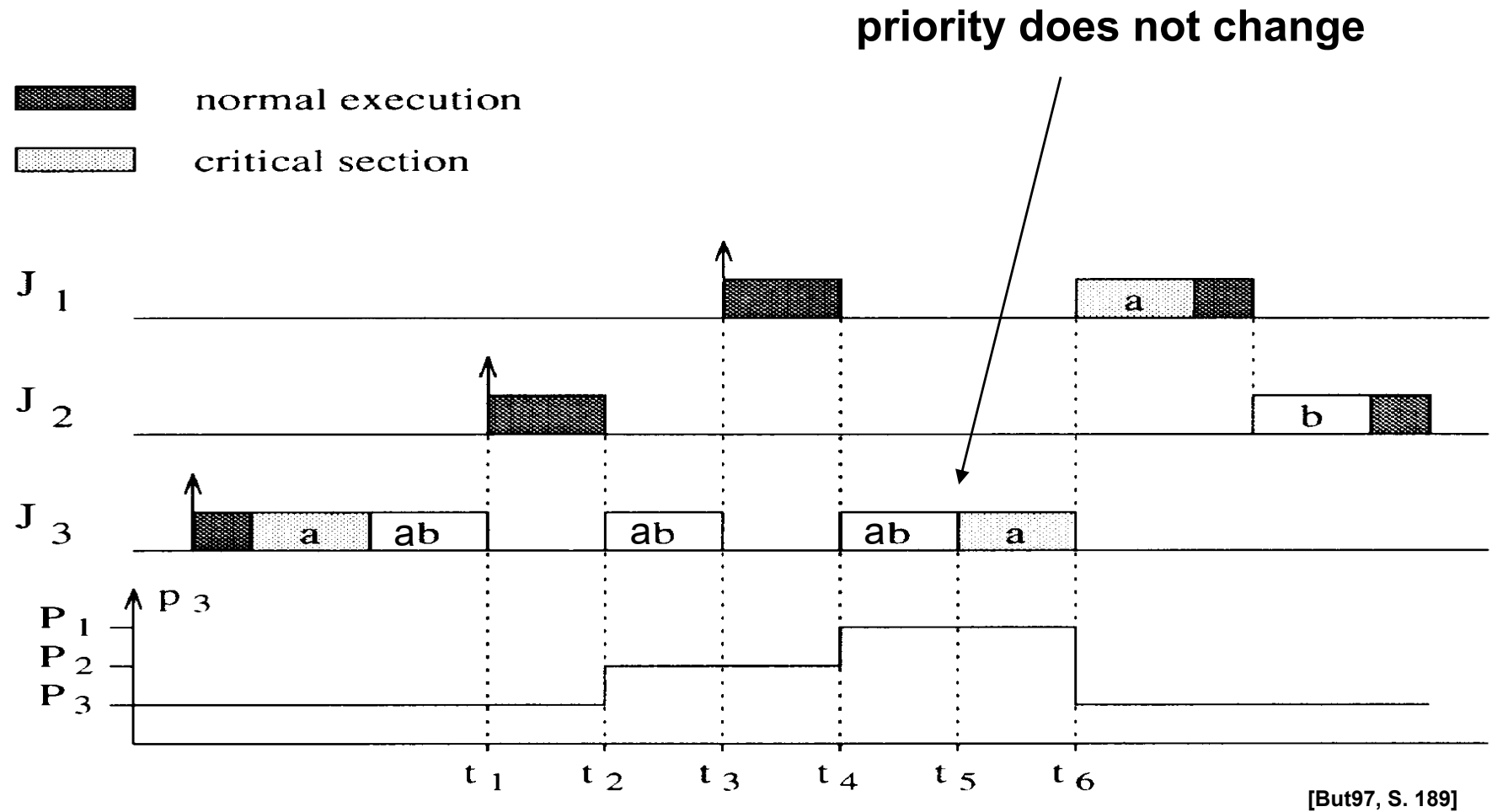


Direct Blocking: higher-priority job tries to acquire a resource held by a lower-priority job

Push-through Blocking: medium-priority job is blocked by a lower-priority job that has inherited a higher priority from a job it directly blocks

Priority Inheritance Protocol (PIP)

- ▶ Example with nested critical sections:

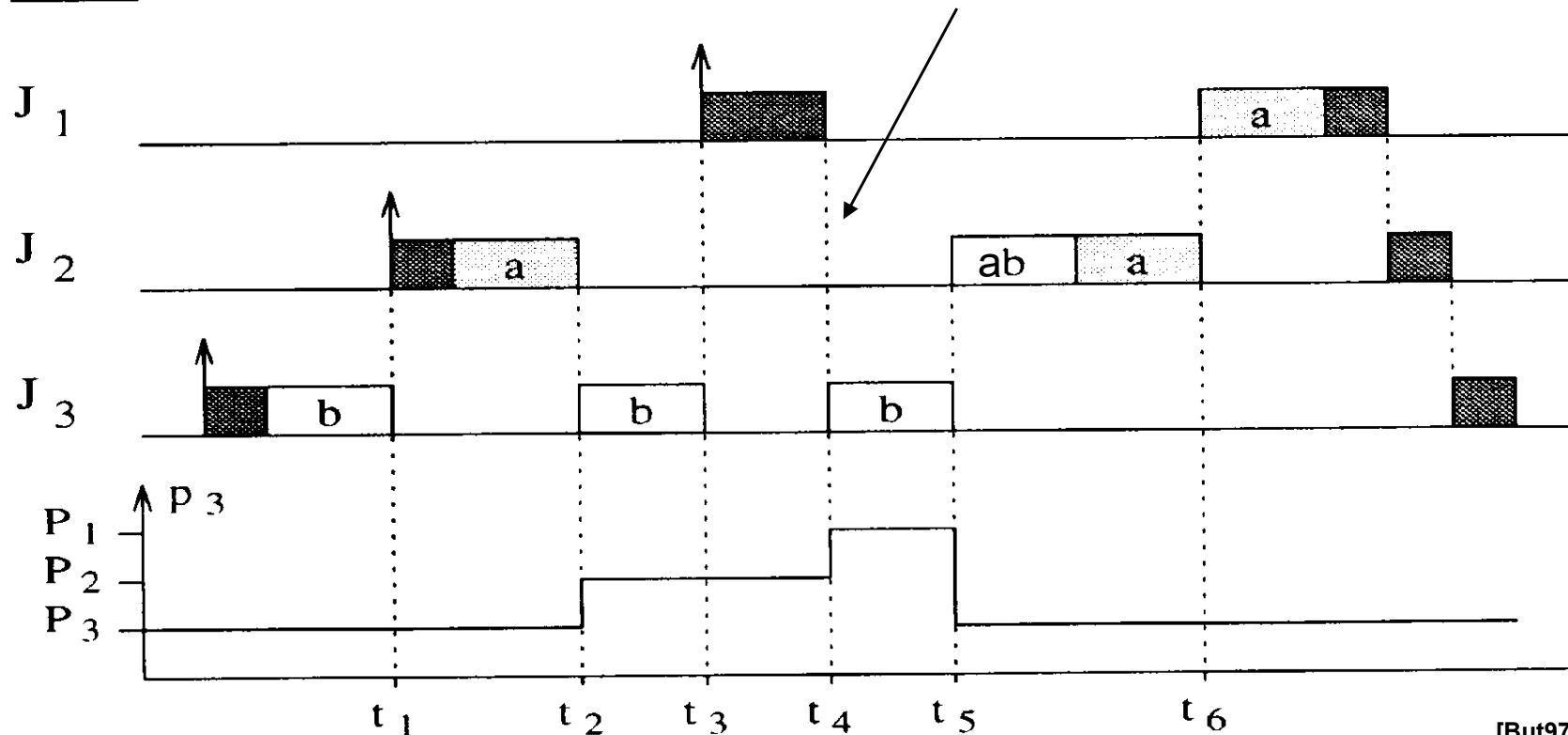


Priority Inheritance Protocol (PIP)

- ▶ Example of transitive priority inheritance:

■ normal execution
▨ critical section

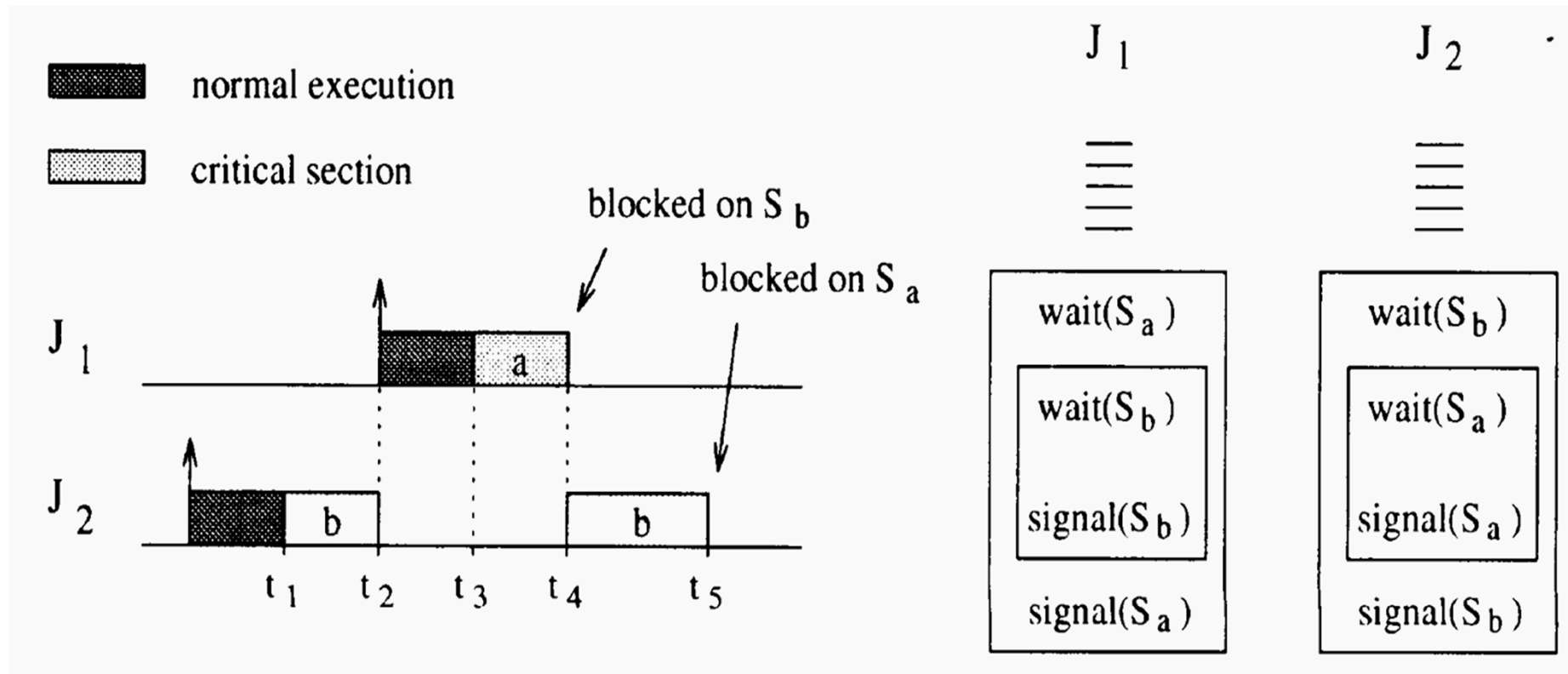
J1 blocked by J2, J2 blocked by J3.
J3 inherits priority from J1 via J2.



[But97, S. 190]

Priority Inheritance Protocol (PIP)

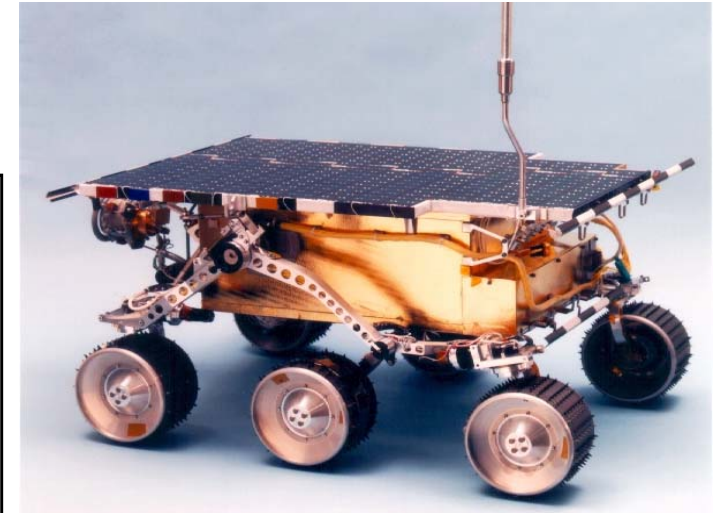
- Problem: *Deadlock*



[But97, S. 200]

The MARS Pathfinder problem (1)

“But a few days into the mission, not long after Pathfinder started gathering meteorological data, the spacecraft began experiencing total system resets, each resulting in losses of data.



The MARS Pathfinder problem (2)

“VxWorks provides preemptive priority scheduling of threads. Tasks on the Pathfinder spacecraft were executed as threads with priorities that were assigned in the usual manner reflecting the relative urgency of these tasks.”

“Pathfinder contained an "information bus", which you can think of as a shared memory area used for passing information between different components of the spacecraft.”

- A bus management task ran frequently with high priority to move certain kinds of data in and out of the information bus. Access to the bus was synchronized with mutual exclusion locks (mutexes).”

The MARS Pathfinder problem (3)

- The meteorological data gathering task ran as an infrequent, low priority thread, ... When publishing its data, it would acquire a mutex, do writes to the bus, and release the mutex.
- The spacecraft also contained a communications task that ran with medium priority.”



High priority: retrieval of data from shared memory

Medium priority: communications task

Low priority: thread collecting meteorological data

The MARS Pathfinder problem (4)

“Most of the time this combination worked fine. However, very infrequently it was possible for an interrupt to occur that caused the (medium priority) communications task to be scheduled during the short interval while the (high priority) information bus thread was blocked waiting for the (low priority) meteorological data thread. In this case, the long-running communications task, having higher priority than the meteorological task, would prevent it from running, consequently preventing the blocked information bus task from running. After some time had passed, a watchdog timer would go off, notice that the data bus task had not been executed for some time, conclude that something had gone drastically wrong, and initiate a total system reset. This scenario is a classic case of priority inversion.”

Priority inversion on Mars

Priority inheritance also solved the Mars Pathfinder problem: the VxWorks operating system used in the pathfinder implements a flag for the calls to mutex primitives. This flag allows priority inheritance to be set to “on”. When the software was shipped, it was set to “off”.

The problem on Mars was corrected by using the debugging facilities of VxWorks to change the flag to “on”, while the Pathfinder was already on the Mars [Jones, 1997].

