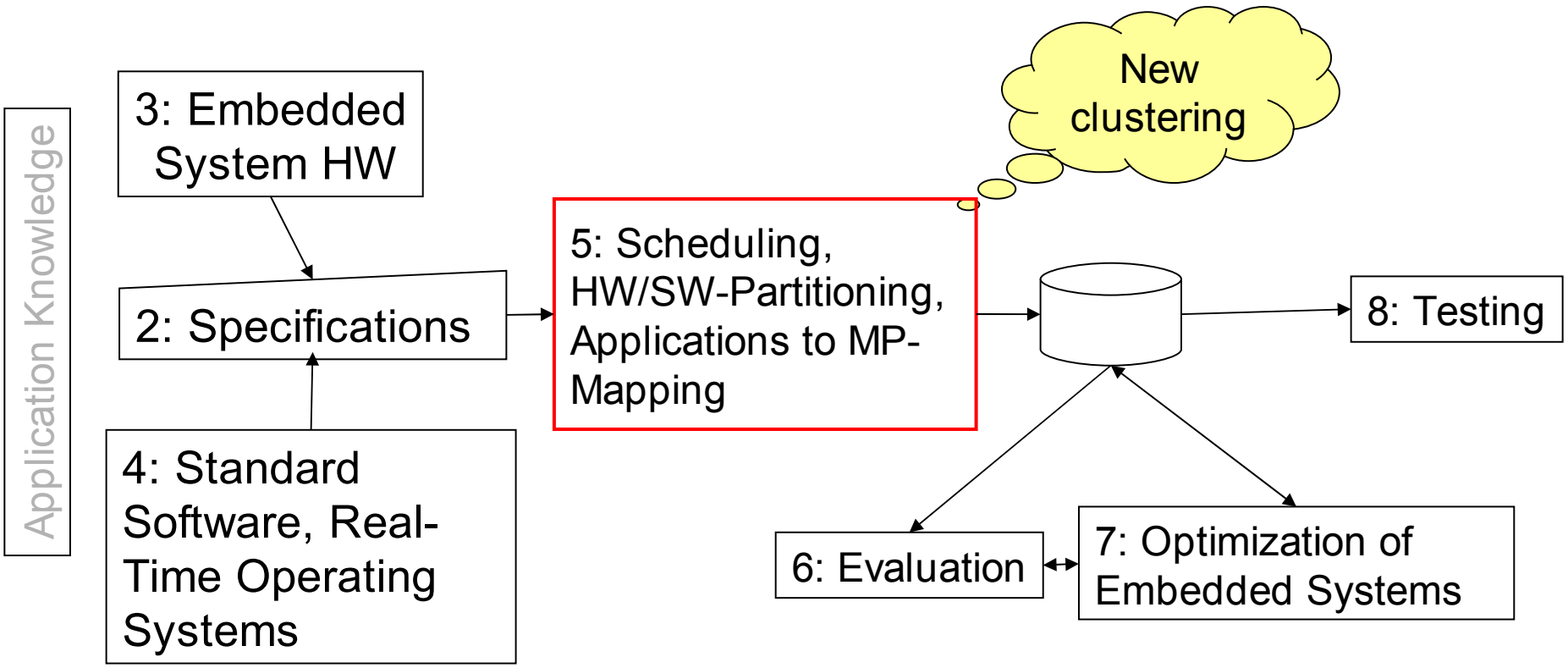
Resource Access Protocols

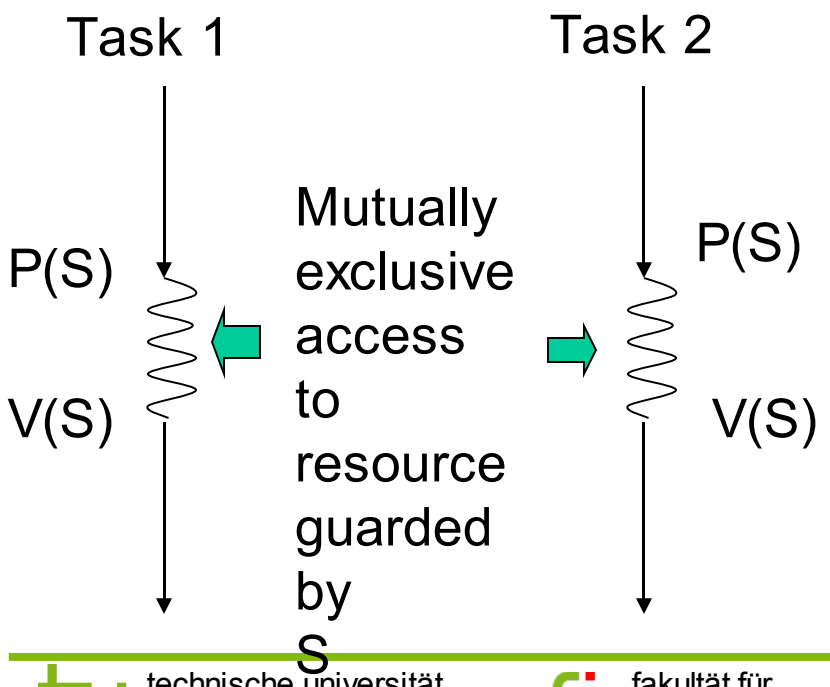
Peter Marwedel
Informatik 12
TU Dortmund
Germany

Structure of this course



Resource access protocols

Critical sections: sections of code at which exclusive access to some resource must be guaranteed. Can be guaranteed with semaphores S or “mutexes”.

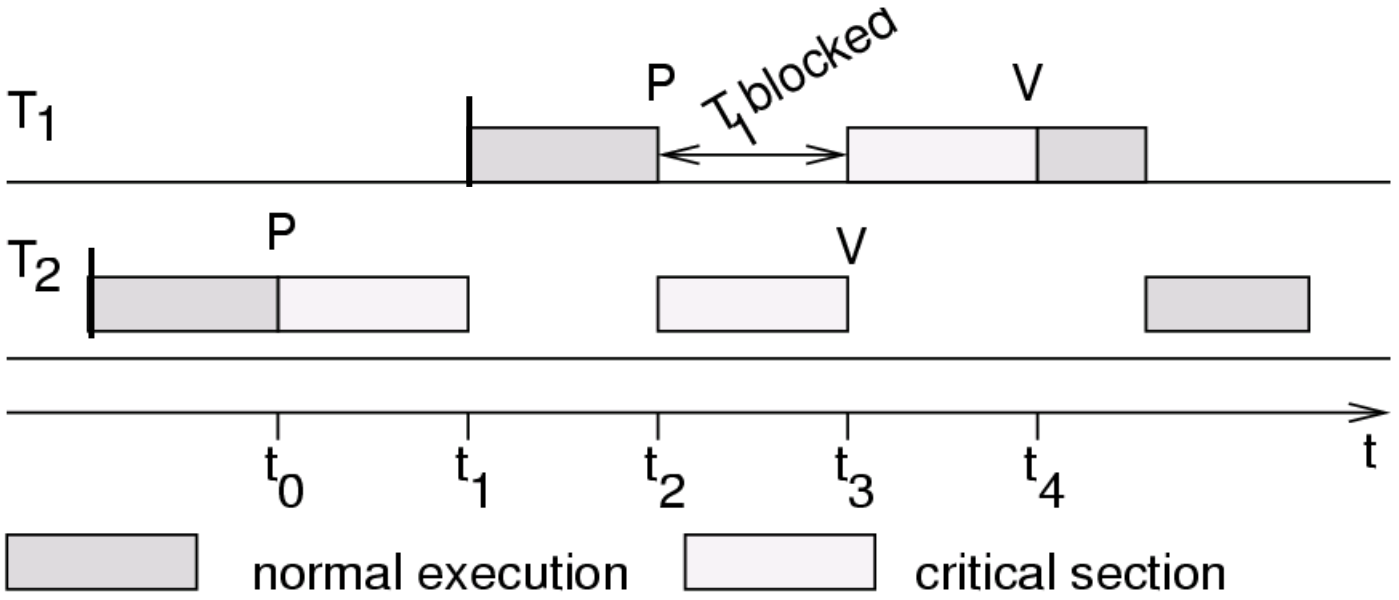


P(S) checks semaphore to see if resource is available and if yes, sets S to „used“. Uninterruptible operations! If no, calling task has to wait.

V(S): sets S to „unused“ and starts sleeping task (if any).

Priority inversion

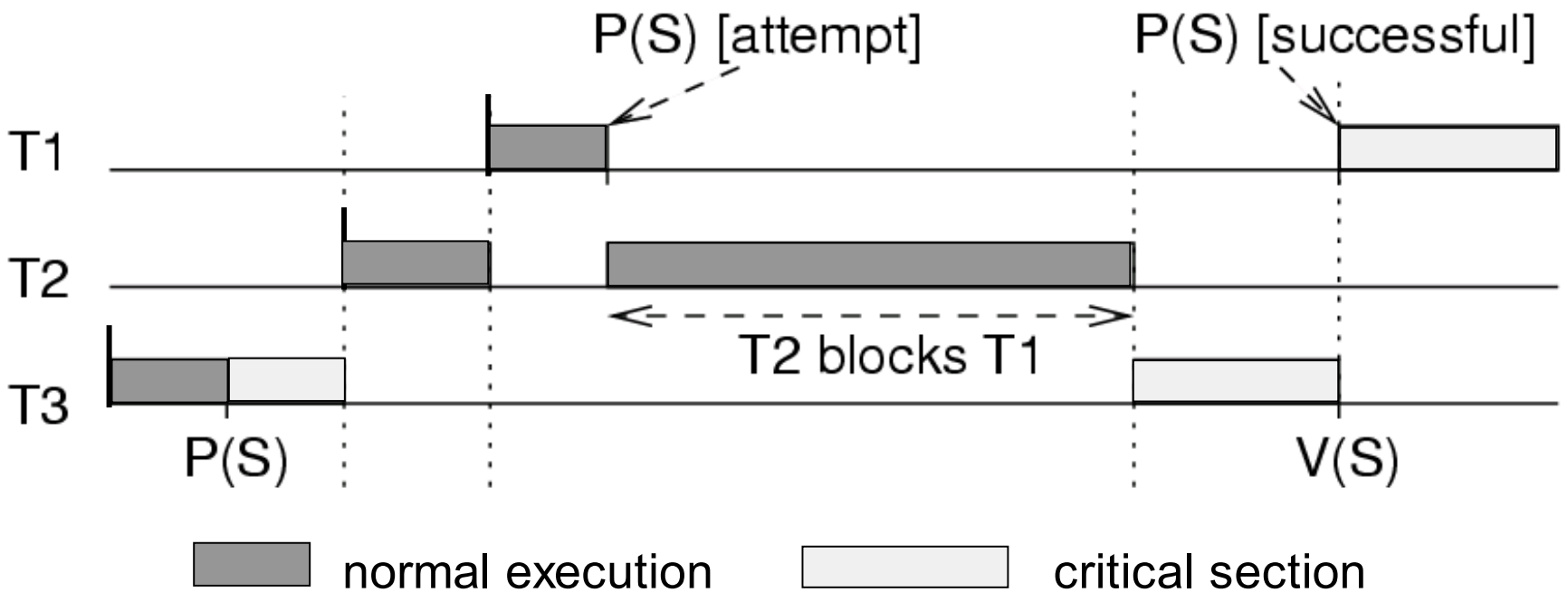
Priority T_1 assumed to be > than priority of T_2.
If T_2 requests exclusive access first (at t_0), T_1 has to wait until T_2 releases the resource (time t_3), thus inverting the priority:



In this example:
duration of inversion bounded by length of critical section of T_2.

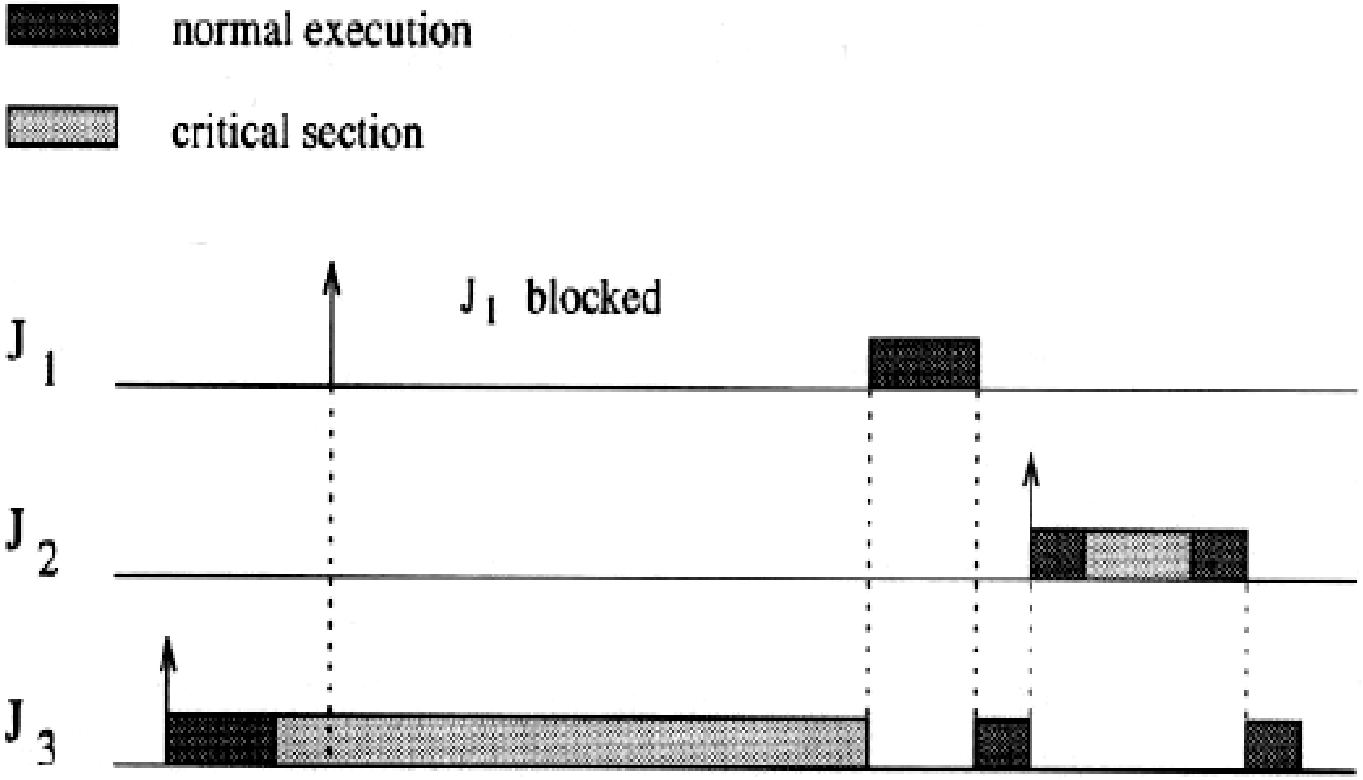
Duration of priority inversion with >2 tasks can exceed the length of any critical section

Priority of T1 > priority of T2 > priority of T3.
T2 preempts T3:
T2 can prevent T3 from releasing the resource.



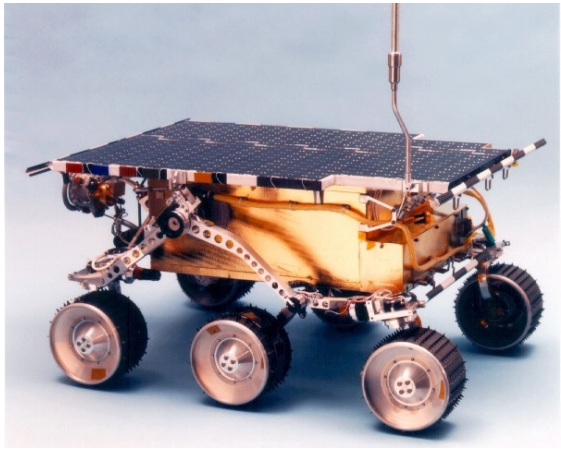
Solutions

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



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 press reported these failures in terms such as "software glitches" and "the computer was trying to do too many things at once".” …



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)

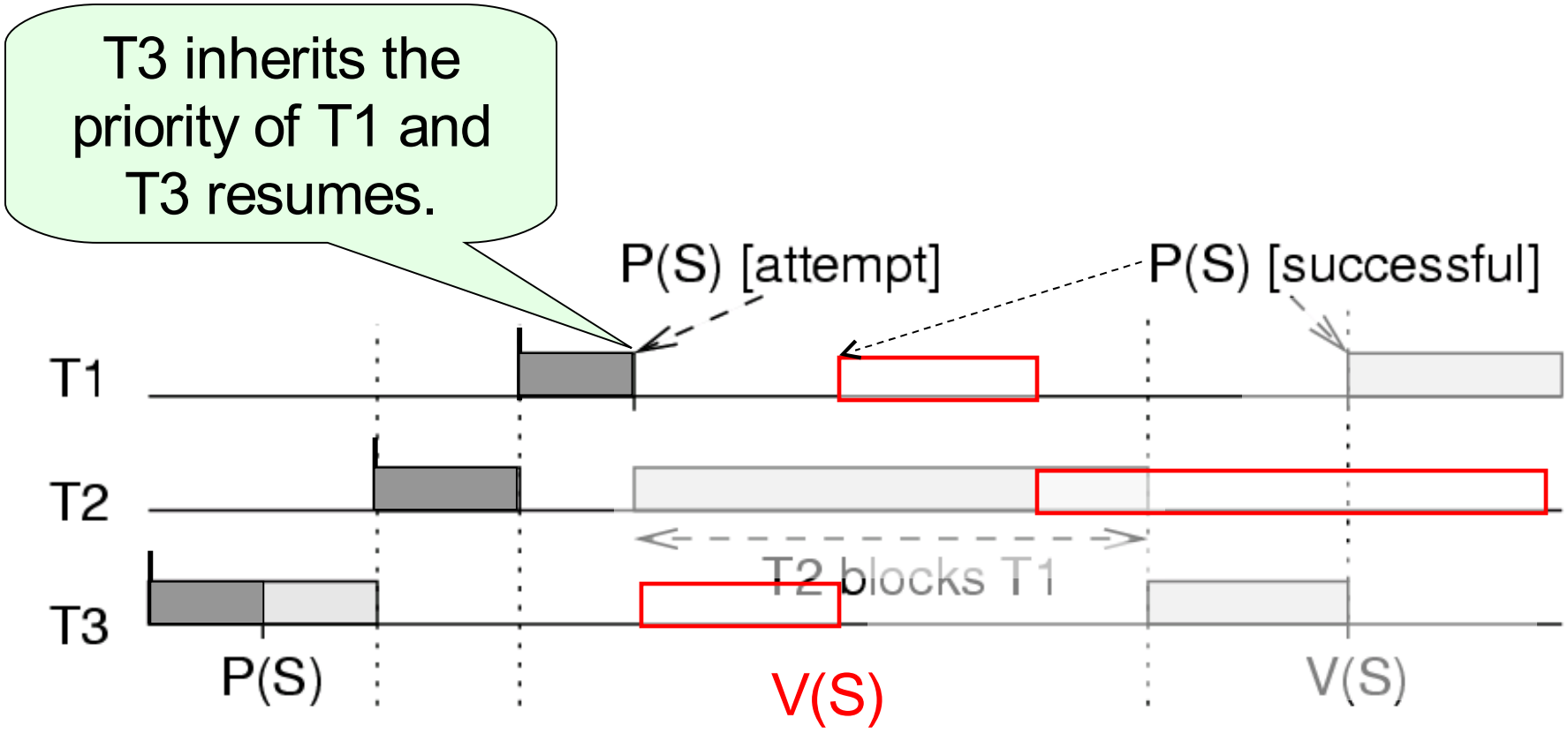
"Most of the time this combination worked fine. However, very infrequently it was possible for an interrupt to occur that caused the (medium priority) communications task to be scheduled during the short interval while the (high priority) information bus thread was blocked waiting for the (low priority) 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."

Coping with priority inversion: the priority inheritance protocol

- Tasks are scheduled according to their active priorities. Tasks with the same priorities are scheduled FCFS.
- If task T1 executes P(S) & exclusive access granted to T2:
  T1 will become blocked.
  If priority(T2) < priority(T1): T2 inherits the priority of T1.
  ☞ T2 resumes.
  Rule: tasks inherit the highest priority of tasks blocked by it.
- When T2 executes V(S), its priority is decreased to the highest priority of the tasks blocked by it.
  If no other task blocked by T2: priority(T2):= original value.
  Highest priority task so far blocked on S is resumed.
- Transitive: if T2 blocks T1 and T1 blocks T0,
  then T2 inherits the priority of T0.

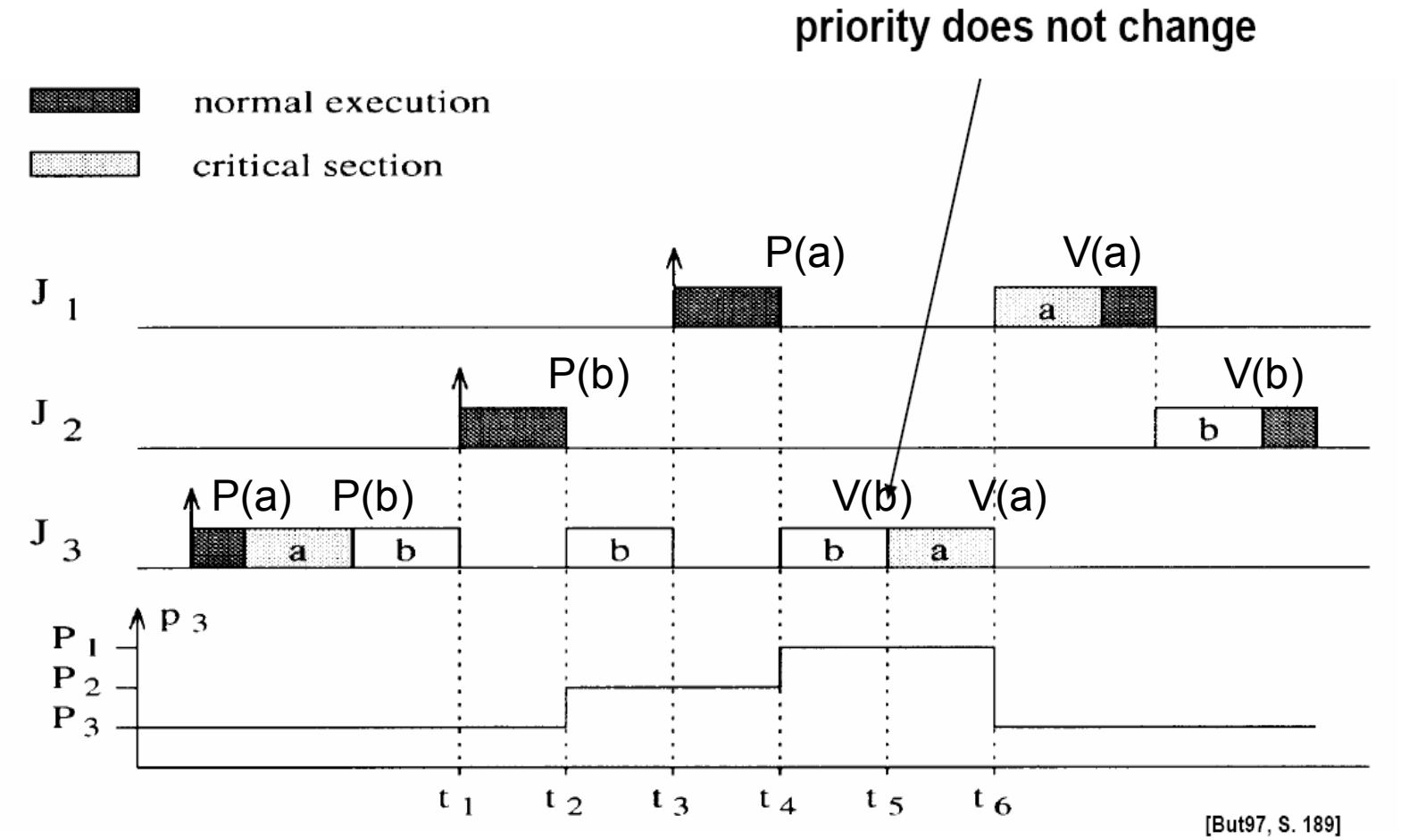
Example

How would priority inheritance affect our example with 3 tasks?



Priority Inheritance Protocol (PIP)

- Example with nested critical sections:

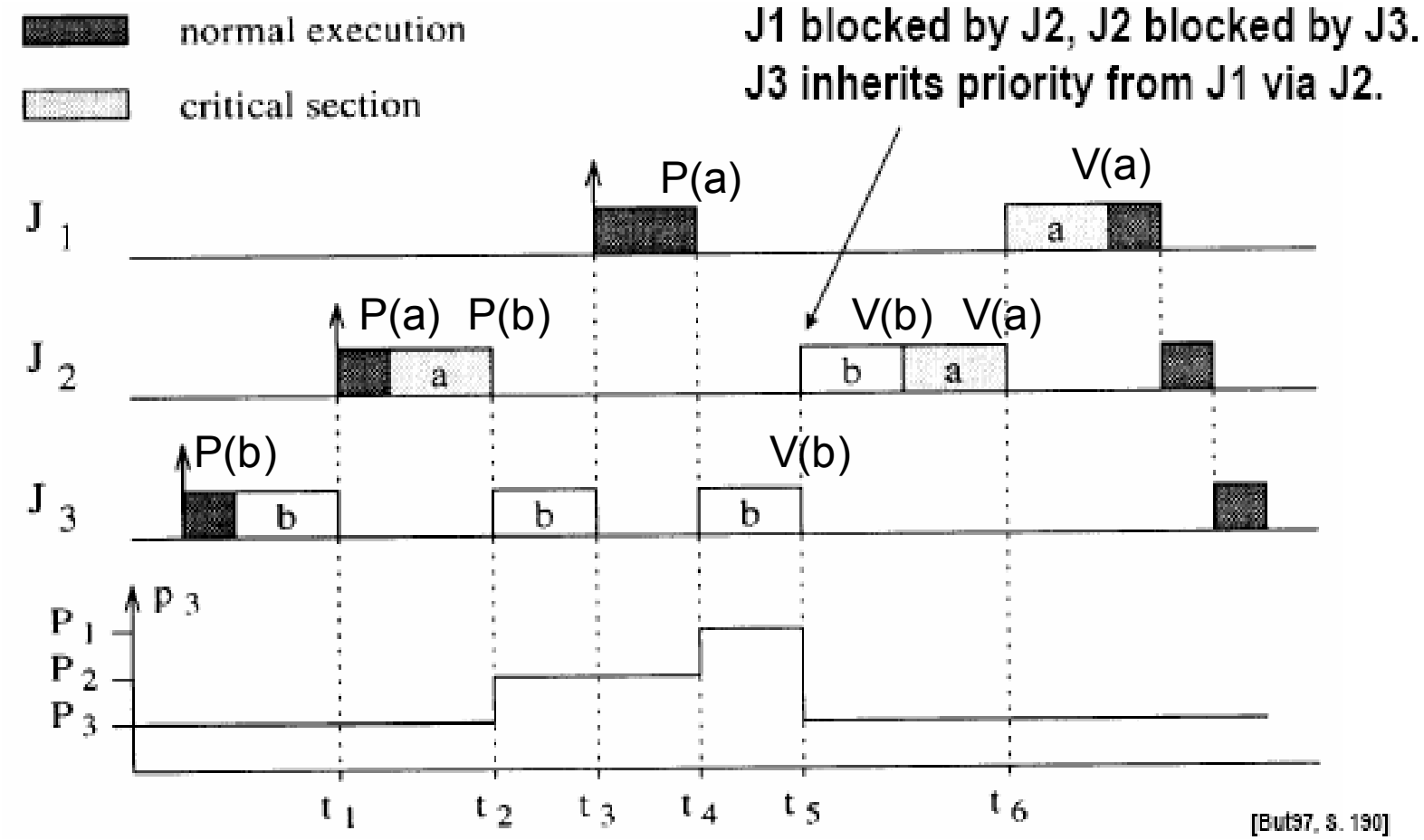


[But97, S. 189]

[P/V added@unido]

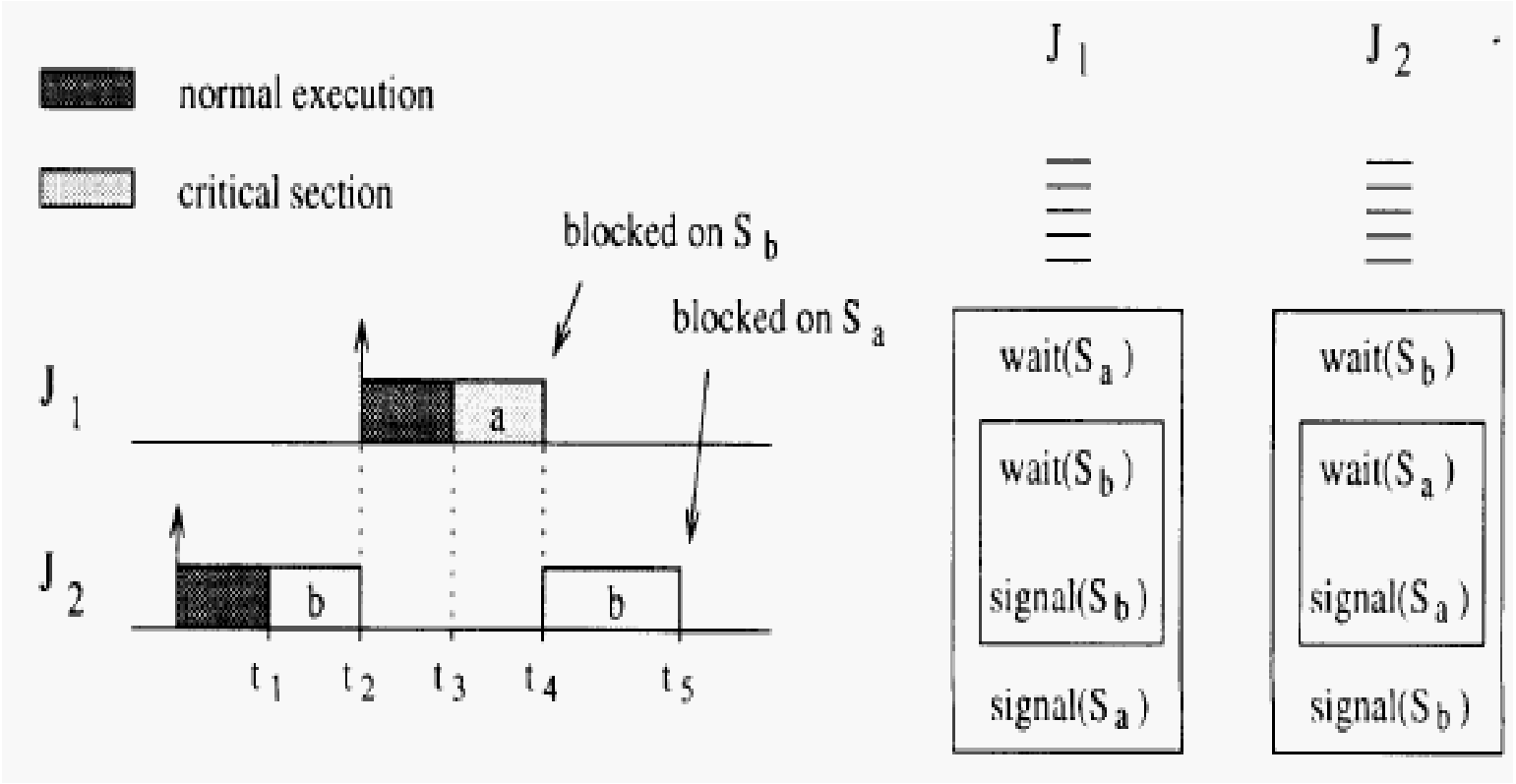
Priority Inheritance Protocol (PIP)

- Example of transitive priority inheritance:



Priority Inheritance Protocol (PIP)

- Problem: ***Deadlock***

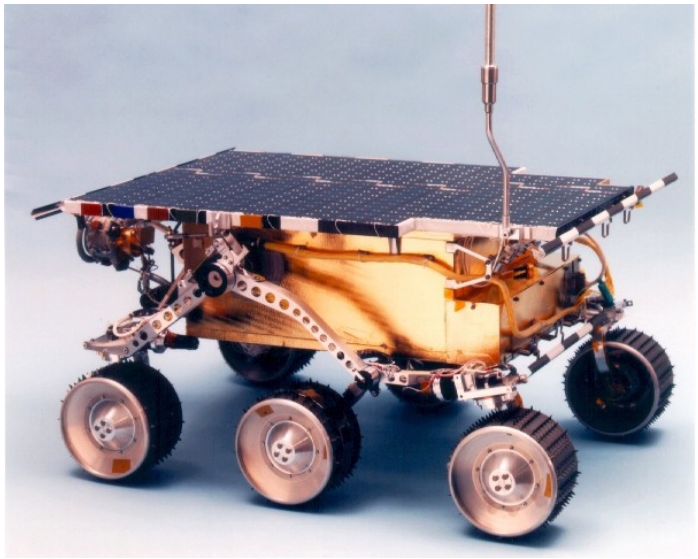


[But97, S. 200]

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].



Remarks on priority inheritance protocol

Possible large number of tasks with high priority.

Possible deadlocks.

Ongoing debate about problems with the protocol:

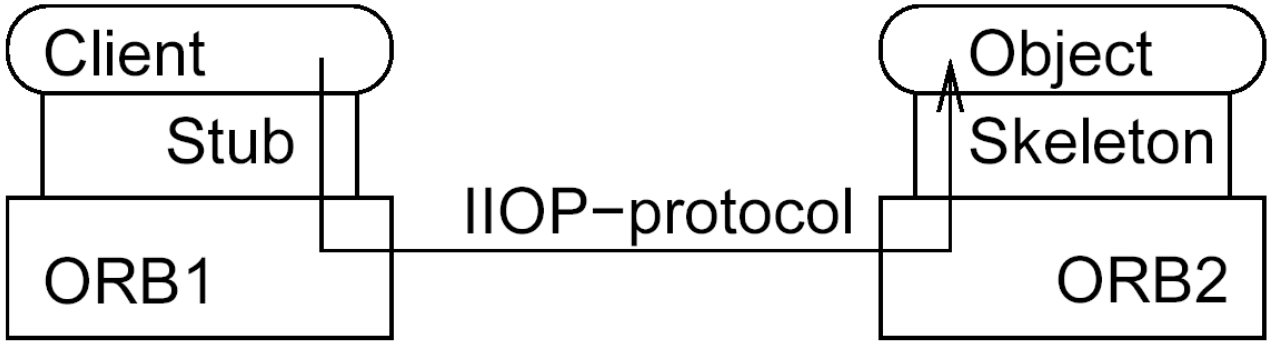
Victor Yodaiken: Against Priority Inheritance,
http://www.fsmlabs.com/articles/inherit/inherit.html

Finds application in ADA: During *rendez-vous*,
task priority is set to the maximum.

More sophisticated protocol: priority ceiling protocol.

Impact on access methods for remote objects

Software packages for access to remote objects;
Example:
CORBA (Common Object Request Broker Architecture).
Information sent to Object Request Broker (ORB) via local stub.
ORB determines location to be accessed and sends information via the IIOP I/O protocol.



Access times not predictable.

Real-time (RT-) CORBA

A very essential feature of RT-CORBA is to provide

- *end-to-end predictability of timeliness in a fixed priority system*.
- This involves *respecting thread priorities between client and server for resolving resource contention*,
- and bounding the latencies of operation invocations.
- Thread priorities might not be respected when threads obtain mutually exclusive access to resources (priority inversion).
- RT-CORBA includes provisions for bounding the time during which such priority inversion can happen.

Real-time CORBA
- Thread priority management -

- RT-CORBA includes facilities for thread priority management.
- Priority independent of the priorities of the underlying OS, even though it is compatible with the RT-extensions of the POSIX standard for OSs [Harbour, 1993].
- The thread priority of clients can be propagated to the server side.
- Priority management for primitives for mutually exclusive access to resources. Priority inheritance protocol must be available in implementations of RT-CORBA.
- Pools of preexisting threads avoid the overhead of thread creation and thread-construction.

Similar issues for “message passing interface” (MPI)

- Message passing interface (MPI): alternative to CORBA
- MPI/RT: a real-time version of MPI [MPI/RT forum, 2001].
- MPI-RT does not cover issues such as thread creation and termination.
- MPI/RT is conceived as a potential layer between the operating system and standard (non real-time) MPI.

Summary

- Priority inversion is a (nasty) effect resulting from the combination of preemptive scheduling with mutual exclusion.
- Protocols such as the priority inheritance protocol (PIP) avoid some problems resulting from priority inheritance.
- However, PIP adds some complexity to the software and, ideally, we would prefer not to have to care about this effect (e.g. by using a dataflow model of computation).