Resource sharing and blocking

The Mars Pathfinder
Unbounded priority inversion
Priority inheritance

Lecture overview

- Unbounded priority inversion problem (occurred in the Mars Pathfinder)
- Blocking and the priority inheritance protocol

- Multithreading and synchronization
  - **Semaphores** are the most common mechanism
  - Supported by the PThreads library

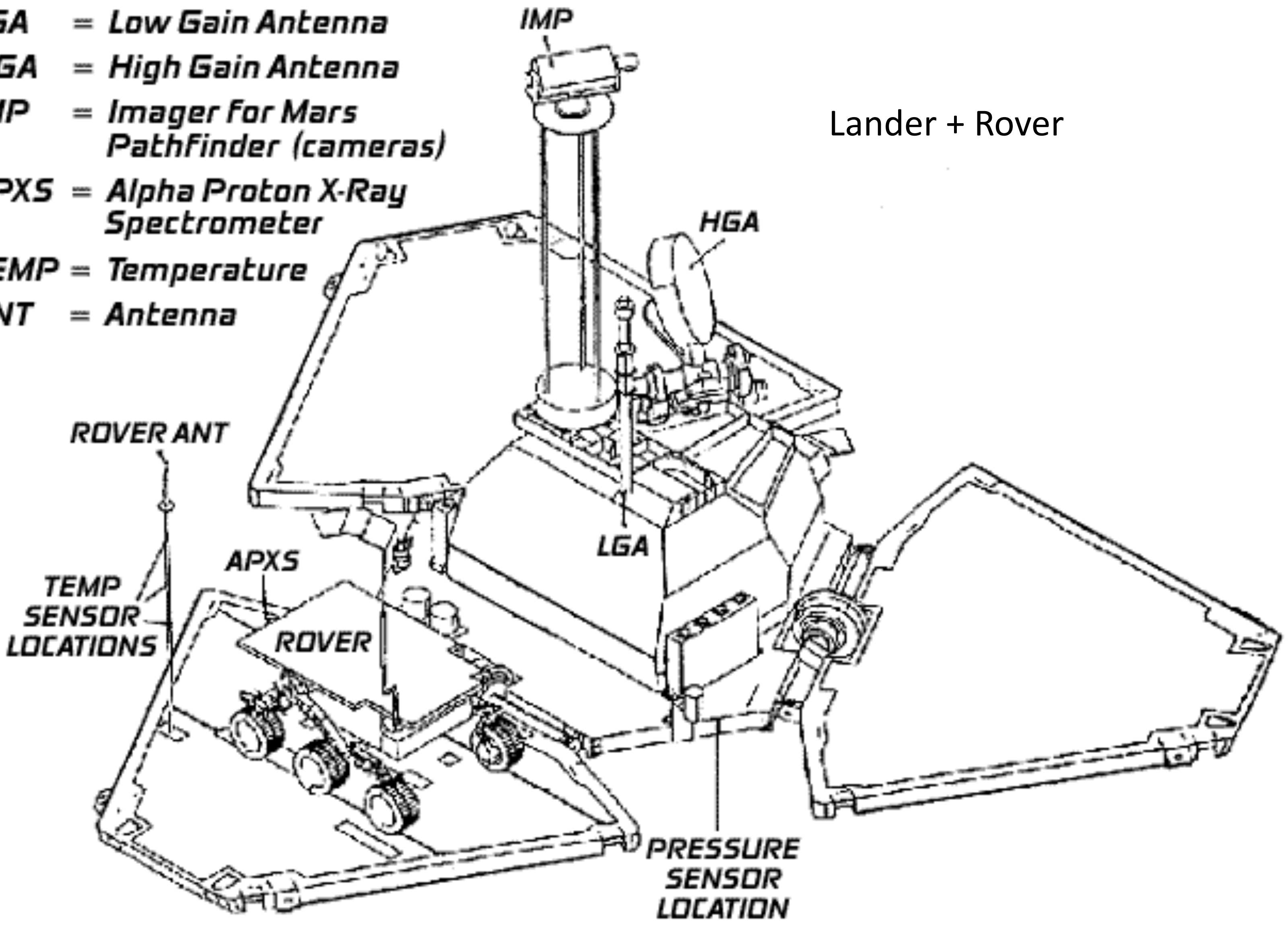
What really happened on Mars?

Source: NASA JPL

Mars Pathfinder

LGA	=	Low Gain Antenna
HGA	=	High Gain Antenna
IMP	=	Imager for Mars Pathfinder (cameras)
APXS	=	Alpha Proton X-Ray Spectrometer
TEMP	=	Temperature
ANT	=	Antenna

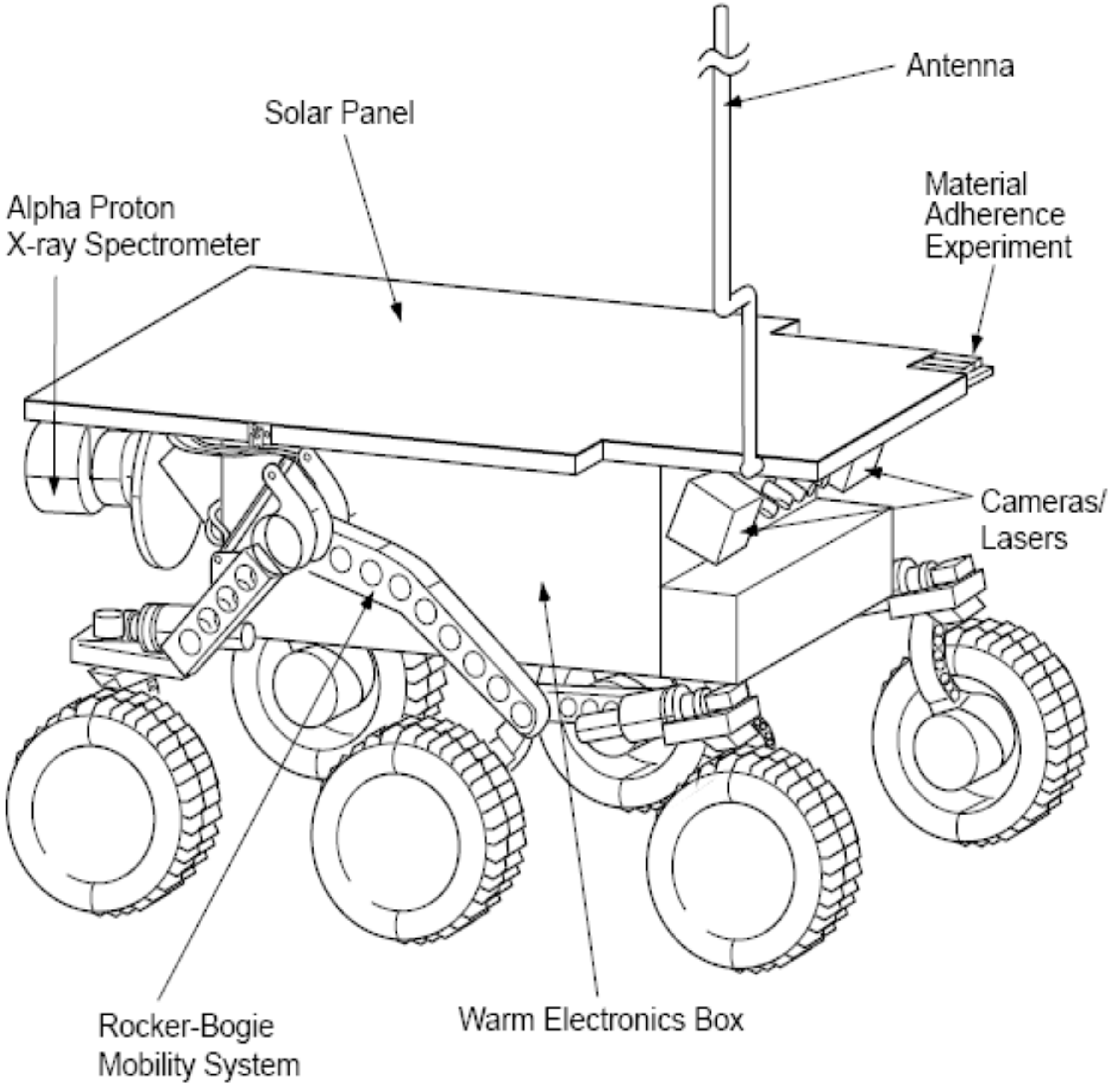
Lander + Rover



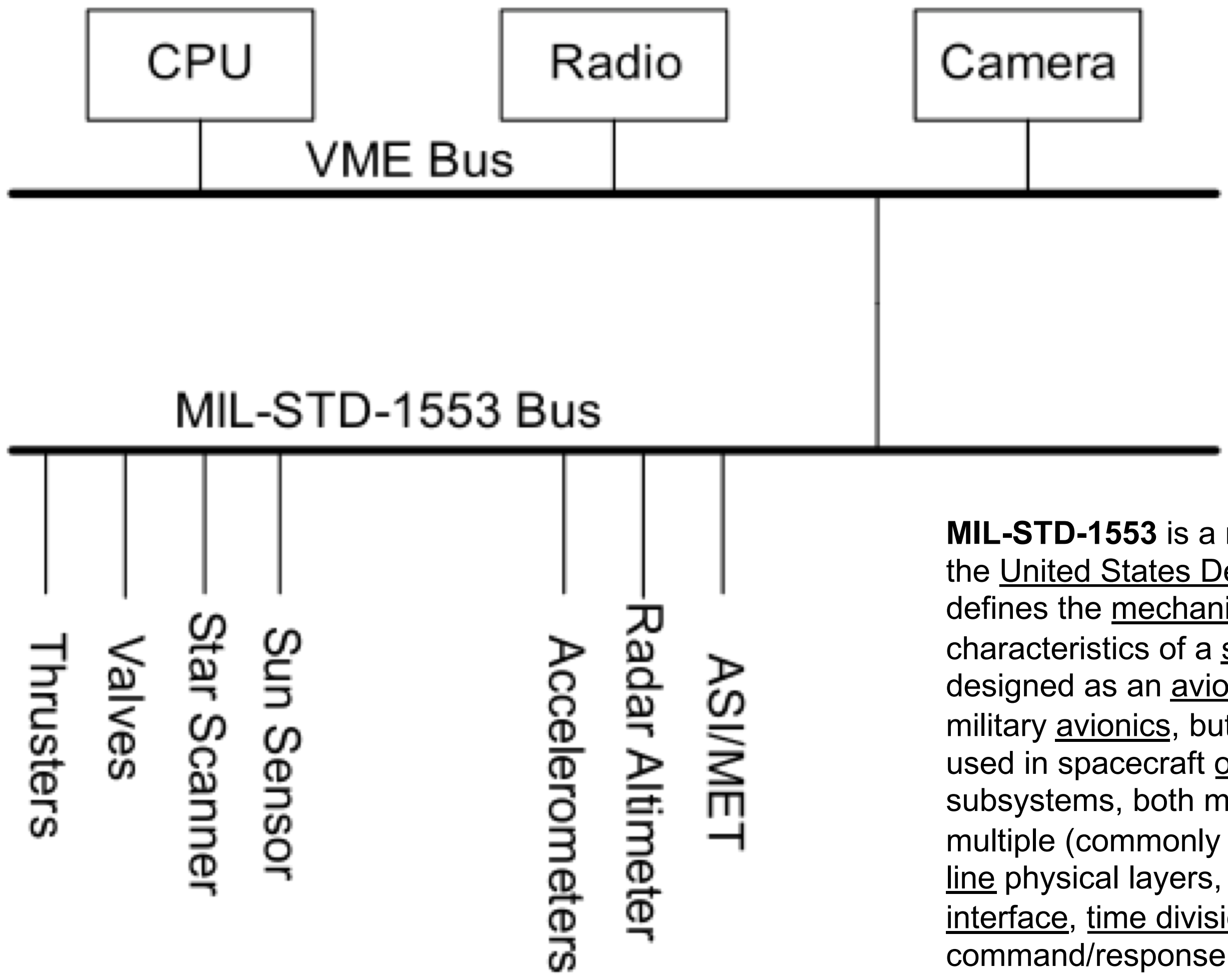
Source: Jet Propulsion Laboratory, 1994

NASA's 25-pound Sojourner Mars rover

Covered about 330 feet (100 meters) over 83 days on the Red Planet in 1997.


Antenna
Solar Panel
Alpha Proton X-ray Spectrometer
Material Adherence Experiment
Cameras/ Lasers
Rocker-Bogie Mobility System
Warm Electronics Box

Mars Pathfinder



MIL-STD-1553 is a military standard published by the United States Department of Defense that defines the mechanical, electrical, and functional characteristics of a serial data bus. It was originally designed as an avionic data bus for use with military avionics, but has also become commonly used in spacecraft on-board data handling (OBDH) subsystems, both military and civil. It features multiple (commonly dual) redundant balanced line physical layers, a (differential) network interface, time division multiplexing, half-duplex command/response protocol, and can handle up to 30 Remote Terminals (devices).

What really happened on Mars?

- Unconventional “landing” -- bouncing onto the Martian surface surrounded by airbags
  - deploying the Sojourner rover
  - gathering and transmitting voluminous data back to Earth
  - panoramic pictures that were such a hit on the Web

- **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.
  - reported in the press as “software glitches” and “the computer was trying to do too many things at once”

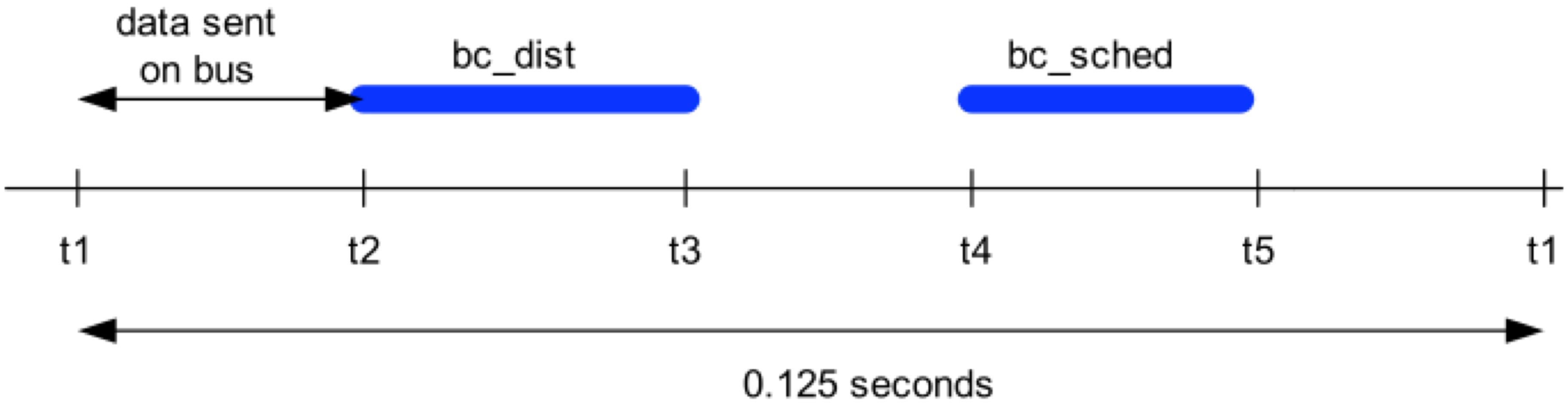
- **The truth:** The failure was a priority inversion failure!

Mars Pathfinder

- Many important tasks
- bc_sched: plans transactions on the 1553 bus
    - highest priority task
- bc_dist: gathers data from the 1553 bus
    - third-highest priority task
- lots of medium-priority tasks
    - asi/met: handles data collection for scientific experiments
    - low priority task

asi/met: Atmospheric Structure Instrument/Meteorology Package
Measured the Martian atmosphere during Pathfinder's descent to the surface, and provided meteorological measurements at the lander.

Schedule

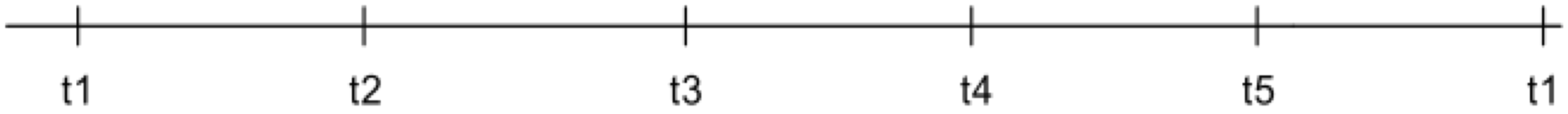


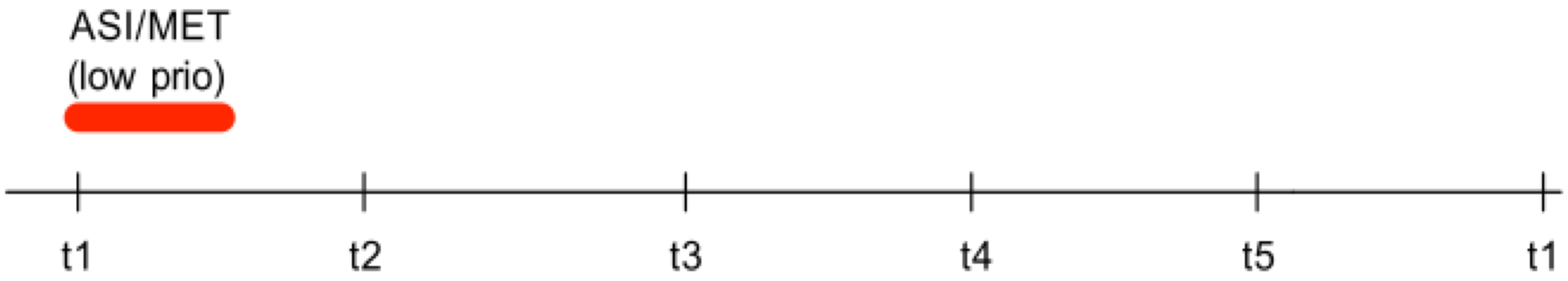
T = 0.125 seconds

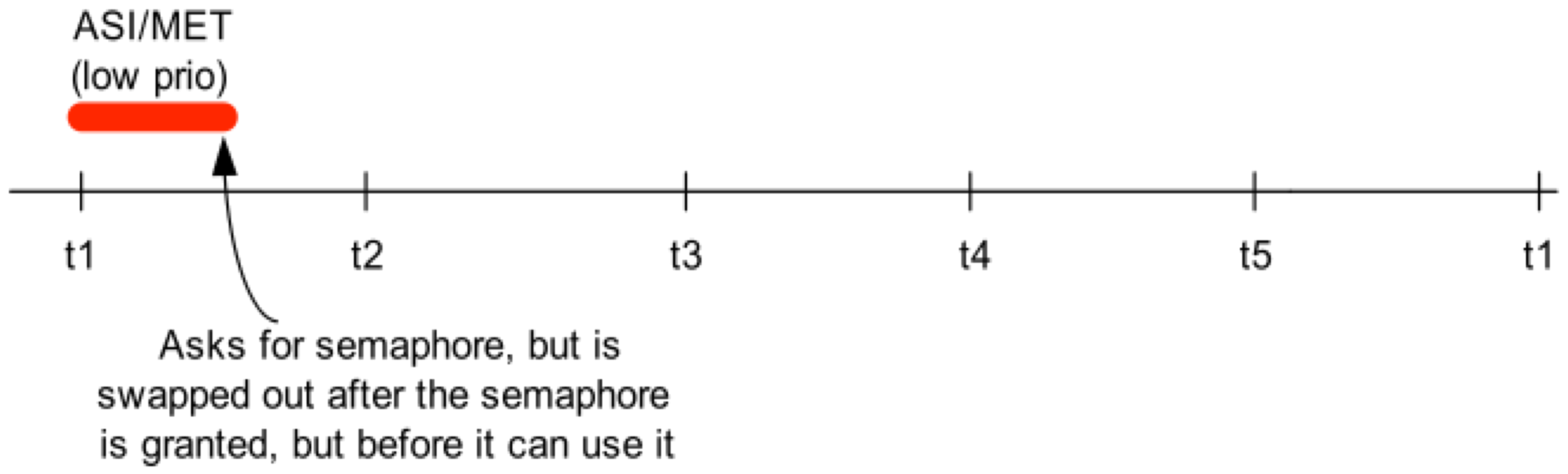
the Deadline for **bc_dist** was t4

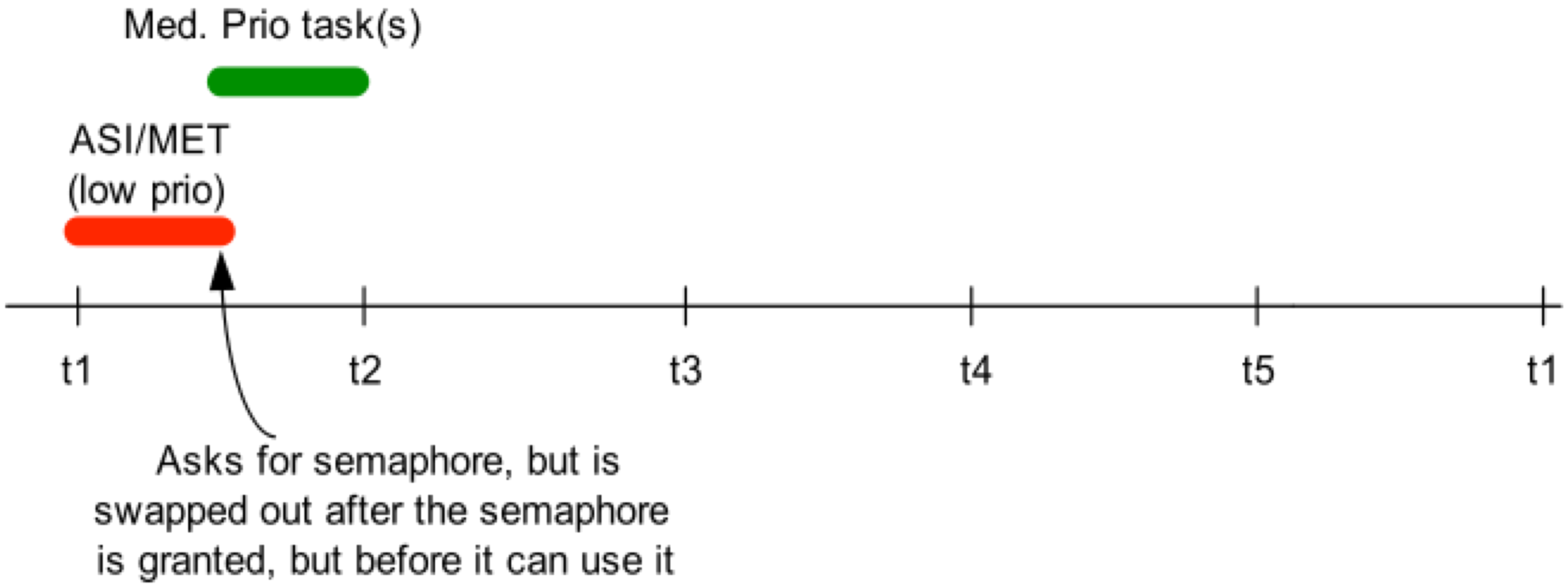
The first thing **bc_sched** would do is make sure that **bc_dist** had *finished*;

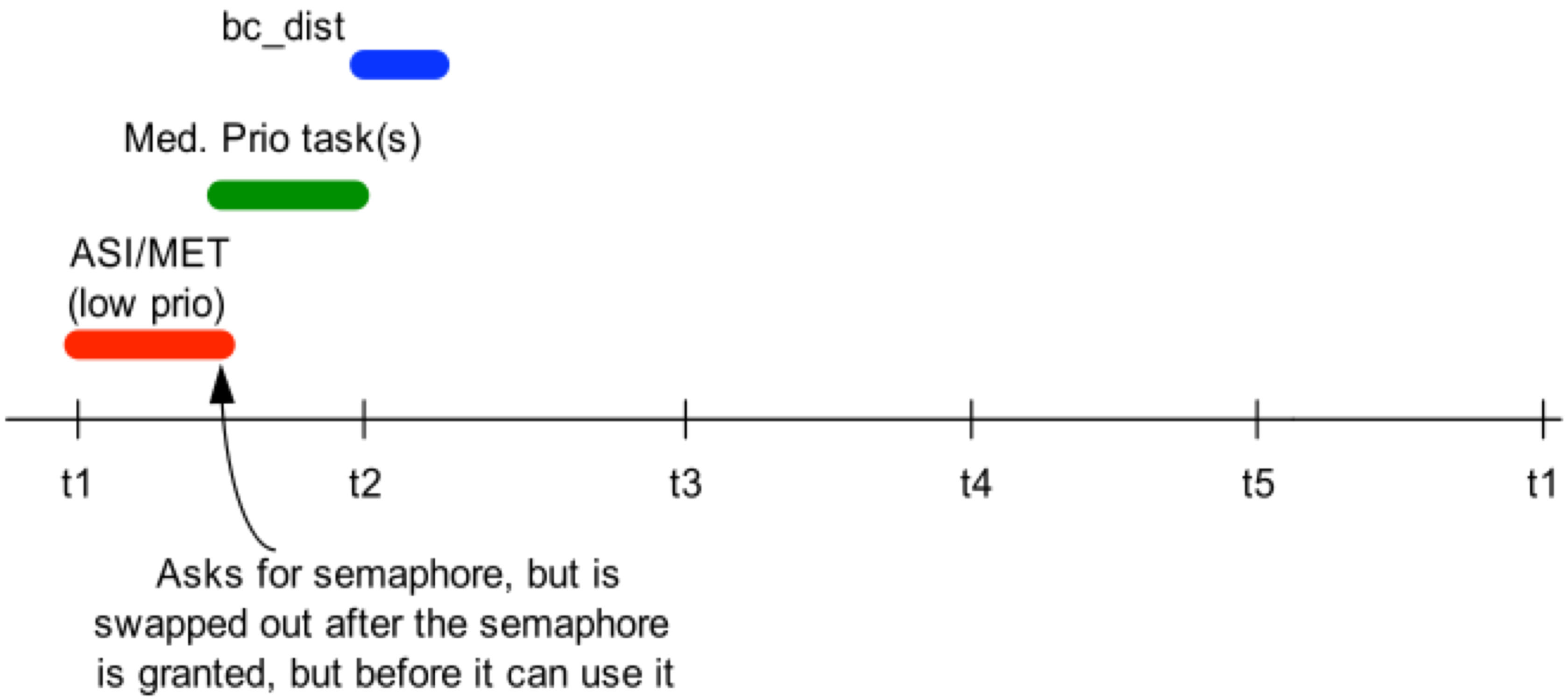
if not, it would reset the system

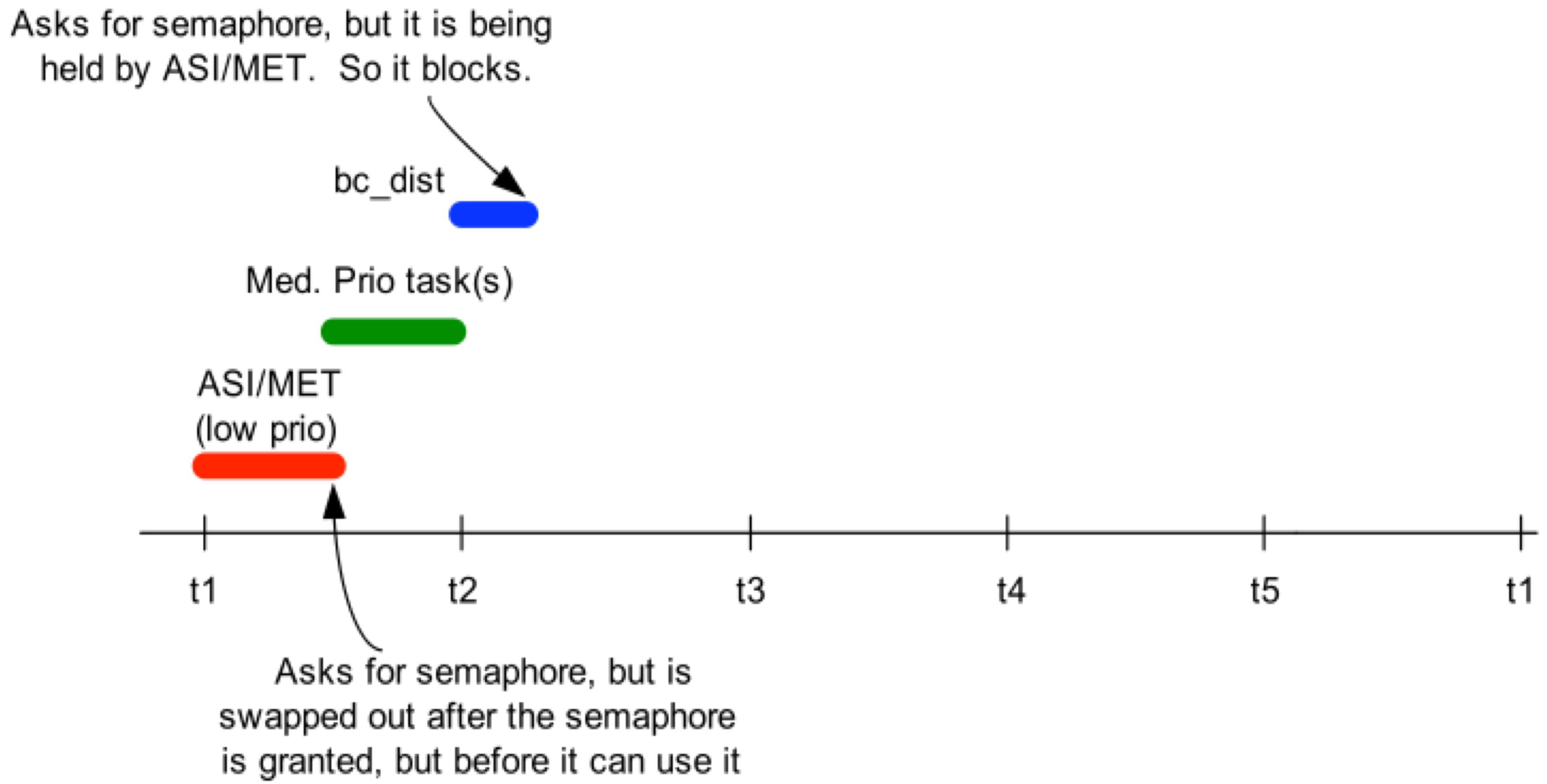

t1
t2
t3
t4
t5
t1

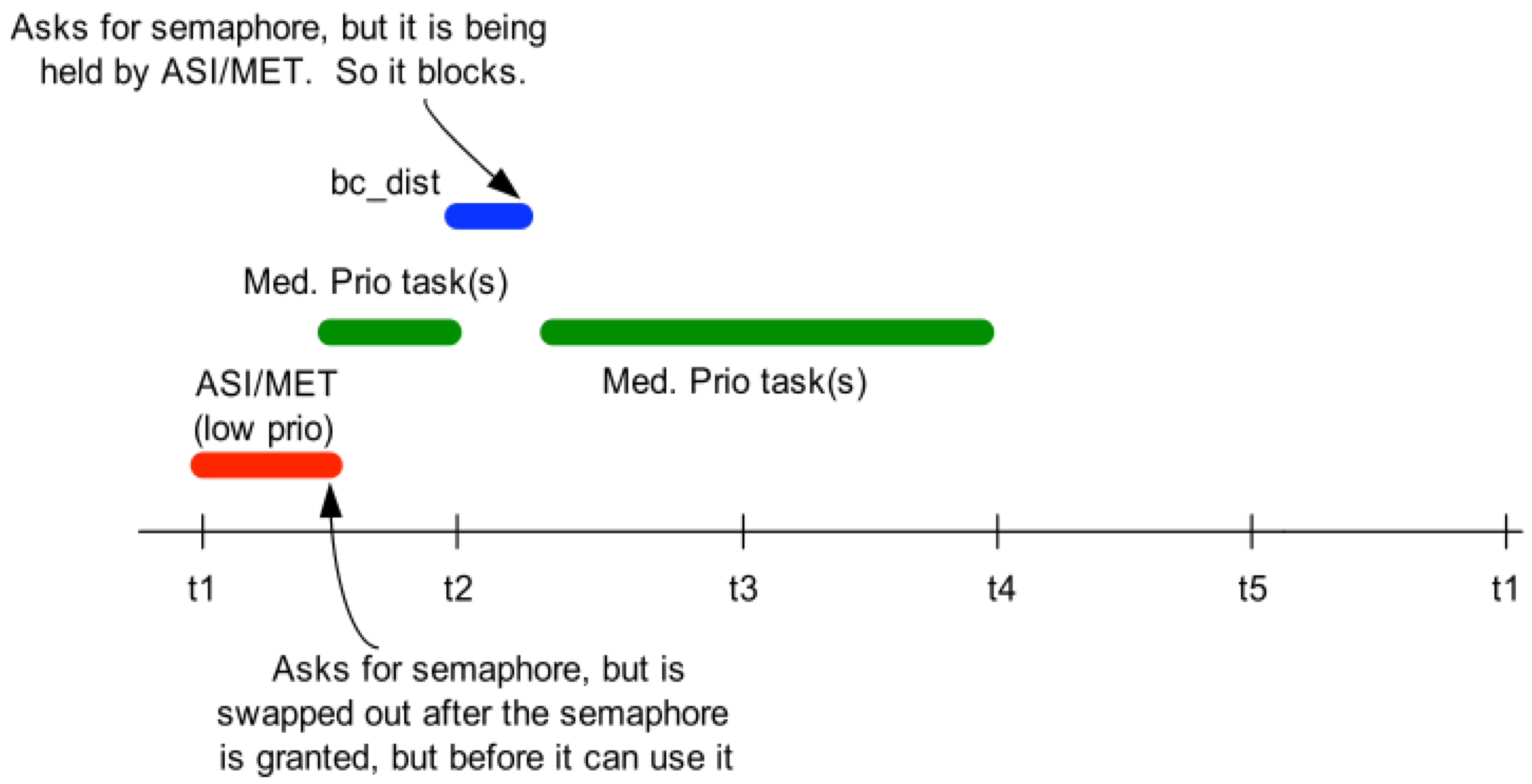

ASI/MET
(low prio)
t1
t2
t3
t4
t5
t1

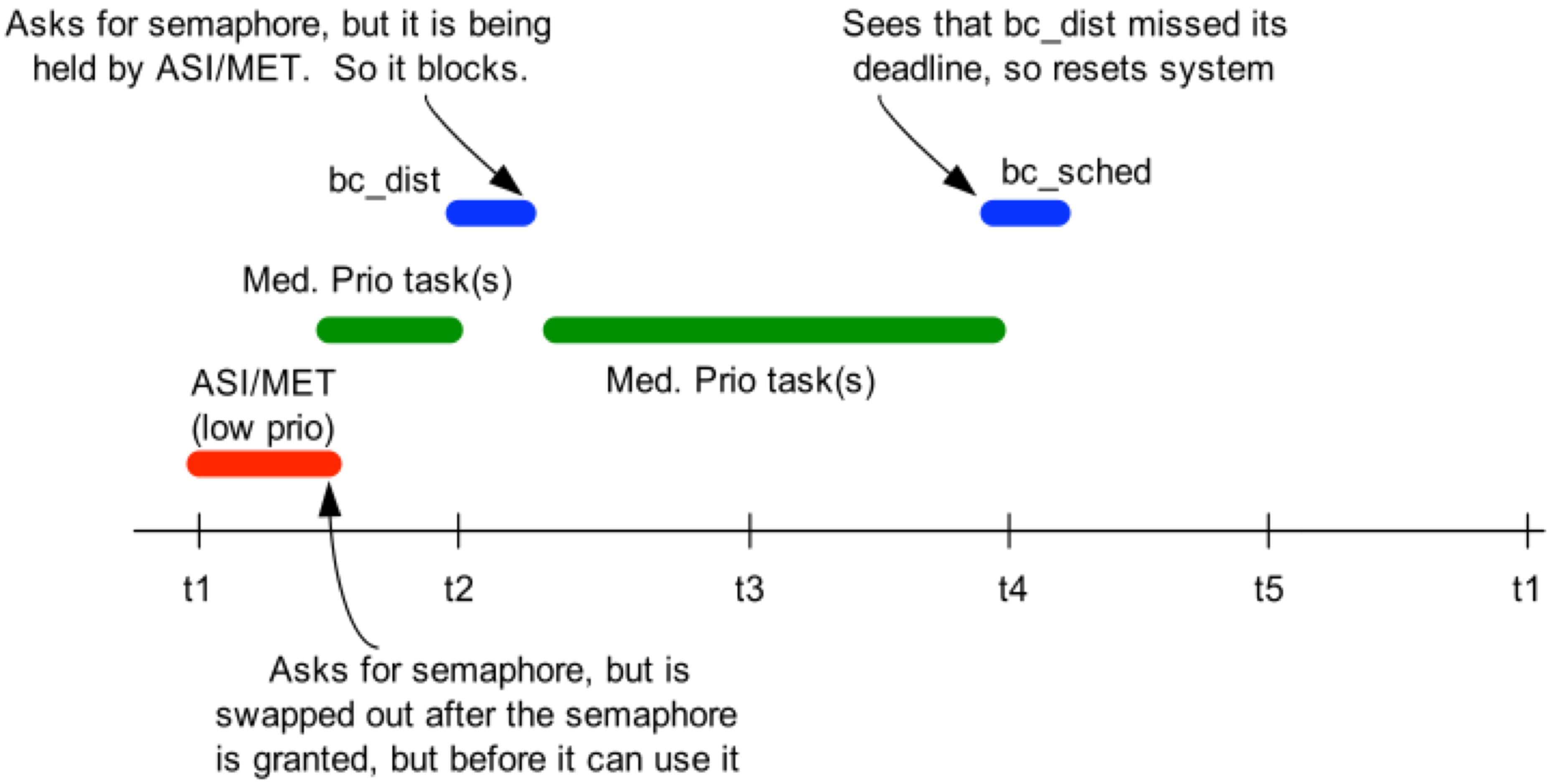

ASI/MET
(low prio)
t1
t2
t3
t4
t5
t1
Asks for semaphore, but is swapped out after the semaphore is granted, but before it can use it


Med. Prio task(s)
ASI/MET
(low prio)
t1
t2
t3
t4
t5
t1
Asks for semaphore, but is swapped out after the semaphore is granted, but before it can use it


bc_dist
Med. Prio task(s)
ASI/MET
(low prio)
t1
t2
t3
t4
t5
t1
Asks for semaphore, but is swapped out after the semaphore is granted, but before it can use it


Asks for semaphore, but it is being held by ASI/MET. So it blocks.
bc_dist
Med. Prio task(s)
ASI/MET (low prio)
t1
t2
t3
t4
t5
t1
Asks for semaphore, but is swapped out after the semaphore is granted, but before it can use it


Asks for semaphore, but it is being held by ASI/MET. So it blocks.
bc_dist
Med. Prio task(s)
ASI/MET (low prio)
Med. Prio task(s)
t1
t2
t3
t4
t5
t1
Asks for semaphore, but is swapped out after the semaphore is granted, but before it can use it


Asks for semaphore, but it is being held by ASI/MET. So it blocks.
Sees that bc_dist missed its deadline, so resets system
bc_dist
bc_sched
Med. Prio task(s)
ASI/MET (low prio)
Med. Prio task(s)
t1
t2
t3
t4
t5
t1
Asks for semaphore, but is swapped out after the semaphore is granted, but before it can use it

This phenomenon is called **Priority Inversion.**

What saved the day? The **priority inheritance protocol**

- How did they find the problem?
  - Trace/log facility + a replica on earth
- How did they fix it?
  - changed the creation flags for the semaphore so as to enable "**priority inheritance**"
- **vxWorks** supplies global configuration variables for parameters such as the "options" parameter for the `semMCreate` used by the select service
  - Turns out that the Pathfinder code was such that this global change worked with minimal performance impact
- Spacecraft code was patched: sent "diff"
  - custom software on the spacecraft (with a whole bunch of validation) modified the onboard copy

Diagnosis of the problem

- Engineer’s initial analysis that *“the data bus task executes very frequently and is time-critical -- we shouldn’t spend the extra time in it to perform priority inheritance”* was exactly wrong
- Did see the problem before landing but could not get it to repeat when they tried to track it down [Heisenbug]
  - neither reproducible or explainable
  - attributed to “hardware glitches”
  - lower priority: focus was on the entry and landing software
- Diagnosing this problem as a black box would have been impossible
- Only detailed traces of actual system behavior enabled the faulty execution sequence to be captured and identified

The importance of good theory

- [Speaking on the Mars Pathfinder problem at the Real-Time Systems Symposium 1997] David [David Wilner, CTO, WindRiver Systems and makers of VxWorks] also said that some of the real heroes of the situation were some people from CMU who had published a paper he'd heard presented many years ago who first identified the priority inversion problem and proposed the solution. He apologized for not remembering the precise details of the paper or who wrote it. Bringing things full circle, it turns out that the three authors of this result were all in the room, and at the end of the talk were encouraged by the program chair to stand and be acknowledged. They were Lui Sha, John Lehoczky, and Raj Rajkumar. When was the last time you saw a room of people cheer a group of computer science theorists for their significant practical contribution to advancing human knowledge? :-) It was quite a moment.

- From “What really happened on Mars?”

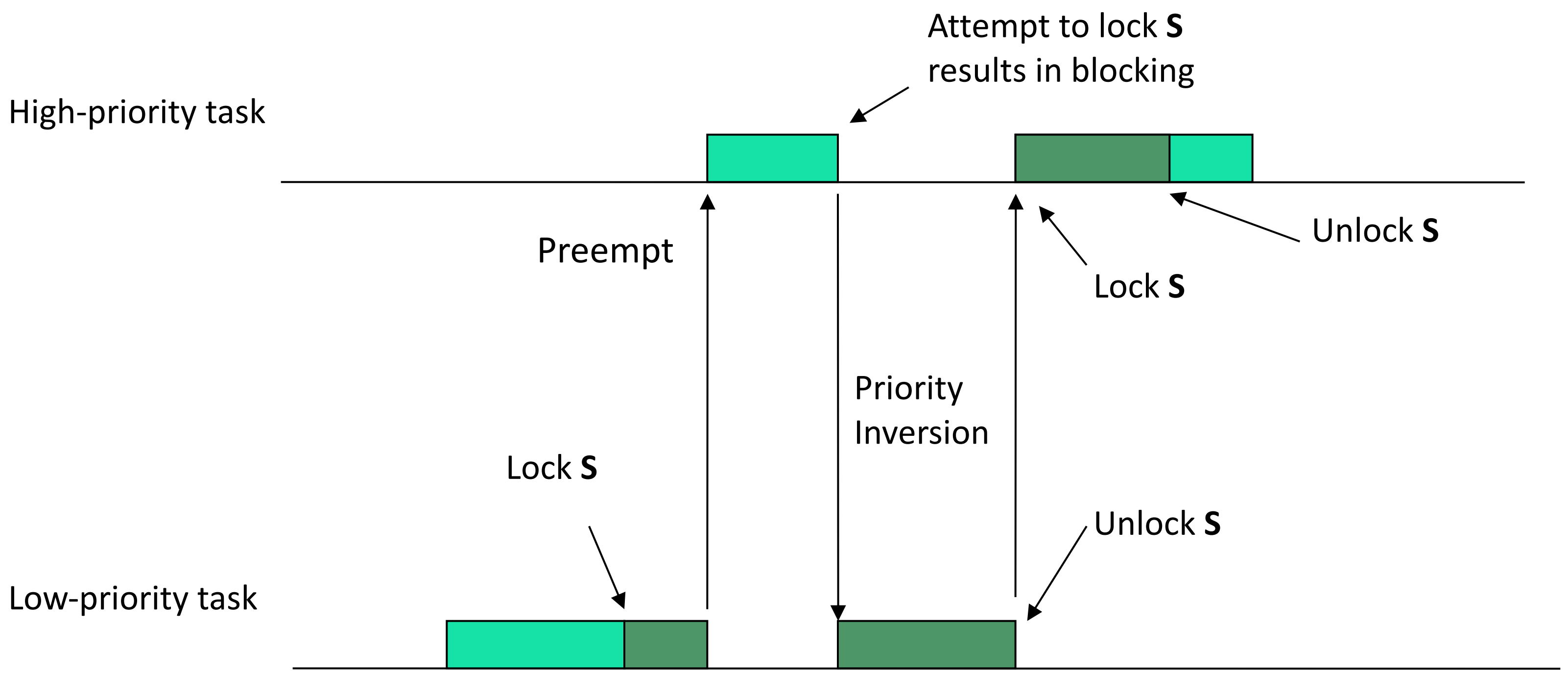
  - Mike B. Jones, Microsoft; http://research.microsoft.com/~mbj/Mars_Pathfinder/Mars_Pathfinder.html

  - For the record, the paper was: L. Sha, R. Rajkumar, and J. P. Lehoczky. Priority Inheritance Protocols: An Approach to Real-Time Synchronization. In IEEE Transactions on Computers, vol. 39, pp. 1175-1185, Sep. 1990.

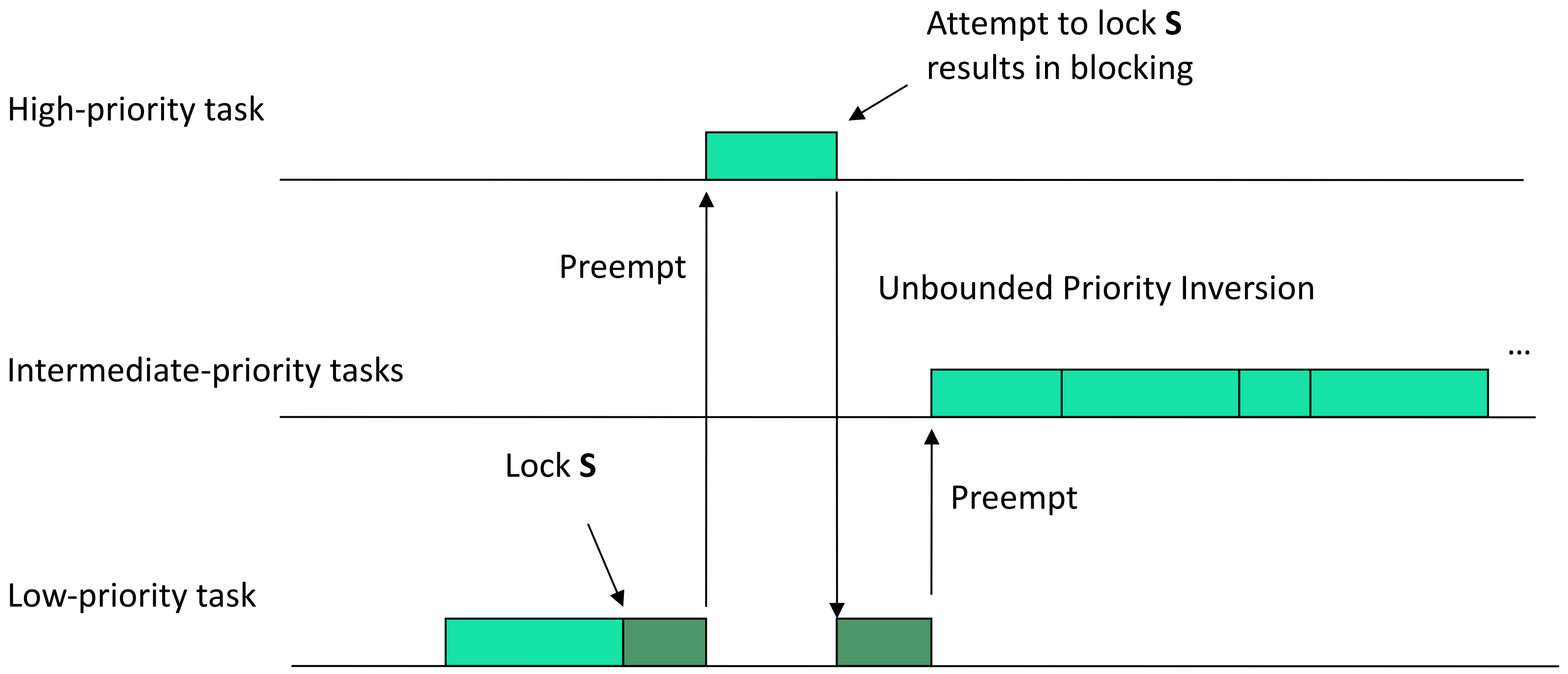
Blocking

- Tasks have synchronization constraints
  - Use semaphores to protect critical sections
- Blocking can cause a *higher priority task to wait for a lower priority task to unlock a resource*
  - We always assumed that higher priority tasks can preempt lower priority tasks
  - As it turns out, that may not be the case... so how do we make the priority rules consistent

Priority Inversion

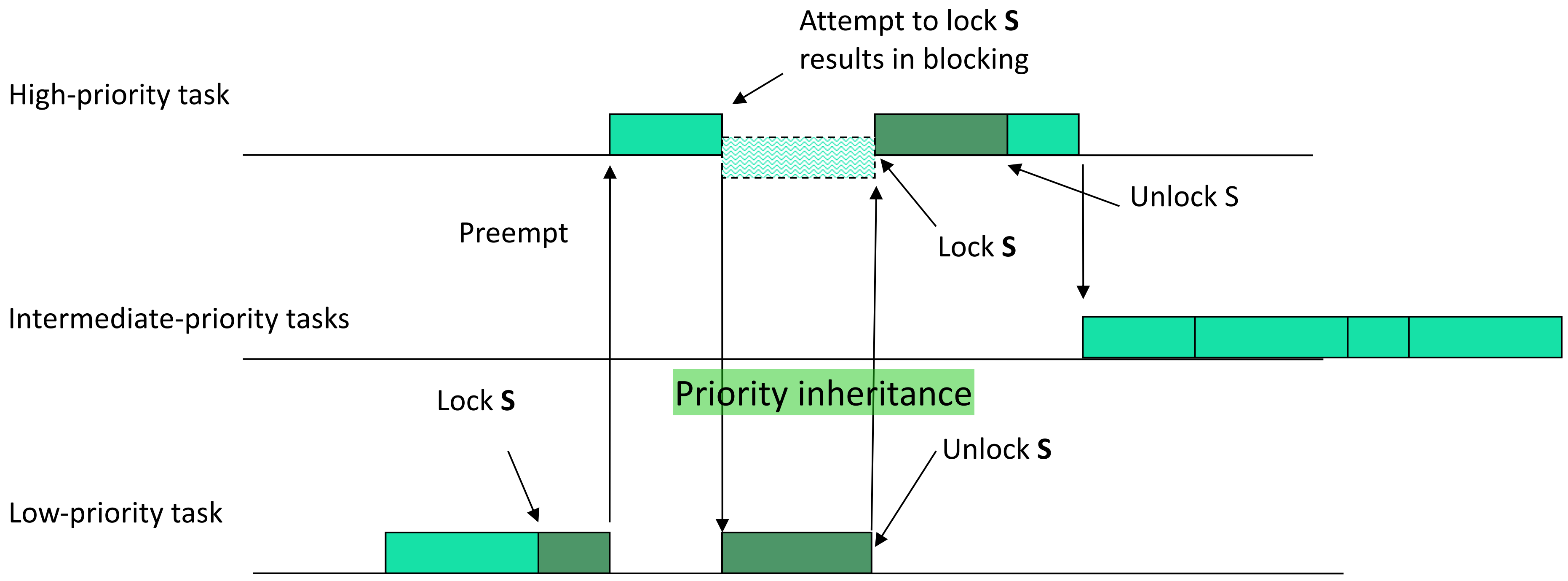


Unbounded Priority Inversion


Attempt to lock **S** results in blocking
High-priority task
Preempt
Unbounded Priority Inversion
…
Intermediate-priority tasks
Lock **S**
Preempt
Low-priority task

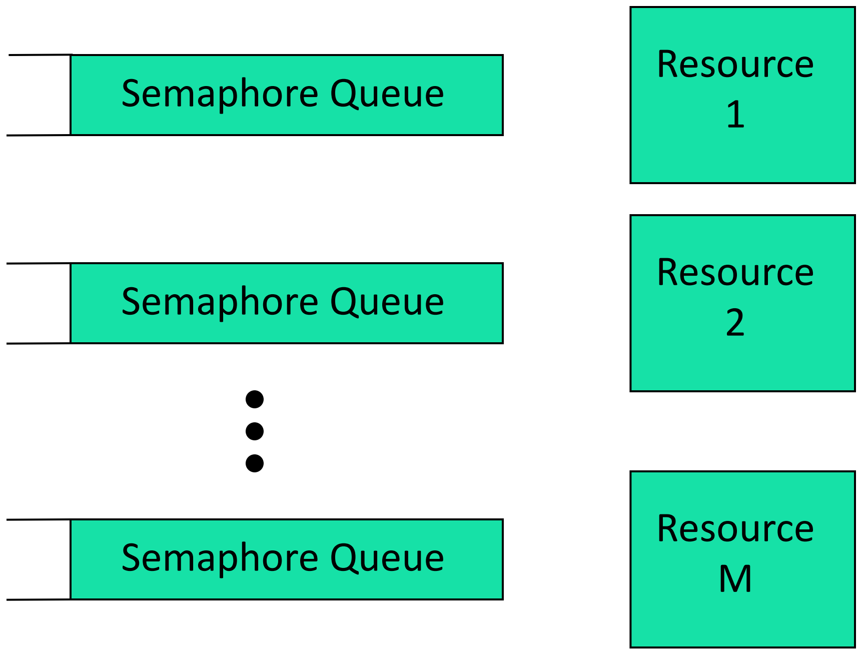
What saved the day on Mars? The ***priority inheritance*** protocol

- Allow a task to **inherit the priority** of the highest priority task that it is blocking



Blocking time

- Consider the instant when a high-priority task arrives
  - What is the maximum length of time it may need to wait for a lower priority task to finish?



If I am a task, priority inversion occurs when

(a) Lower priority task holds a resource I need (**direct blocking**);

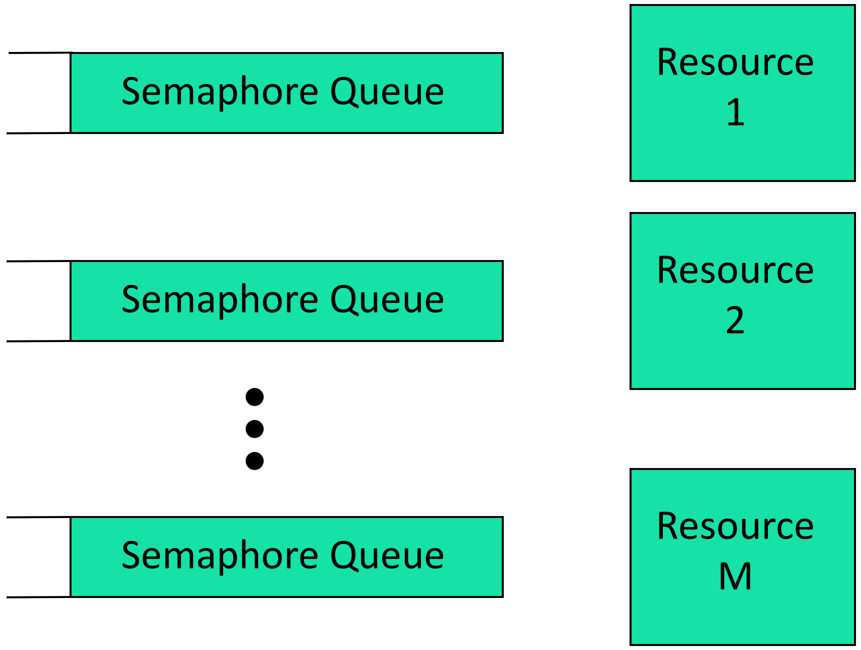
(b) Lower priority task inherits a higher priority than me because it holds a resource the higher-priority task needs (**push-through blocking**)

Blocking time

- What is the longest time a task can be blocked (waiting for lower priority tasks to release a resource)?
  - **Observe:** a task can be blocked only by a lower priority task
  - **Priority Inheritance:** *If there are ℓ_i lower-priority tasks that can block task τ_i and s_i distinct semaphores that can block task τ_i, then τ_i* can be blocked for the duration of at most $\min(\ell_i, s_i)$ critical sections
  - Let the length of the *largest* critical section of Task T_i be b_i
  - The total priority inversion (blocking) time experienced by Task T_i is denoted B_i

Maximum blocking time

- If all critical sections of task *i* are of equal length, *b*$_i$
  - Blocking time of task i is $B_i = b_i \times \min(\ell_i, s_i)$
  - Why?
  - And what if the critical sections are of differing lengths?



If I am a task, priority inversion occurs when

(a) Lower priority task holds a resource I need (**direct blocking**);

(b) Lower priority task inherits a higher priority than me because it holds a resource the higher-priority task needs (**push-through blocking**)

Maximum blocking time

- If all critical sections of task *i* are of equal length, *b_i*
    - Blocking time of task i is $B_i = b_i \times \min(\ell_i, s_i)$
    - Why?
- And what if the critical sections are of differing lengths?
    - Find the maximum length critical section for each resource
    - Add the top $\min(\ell_i, s_i)$ sections in size (exponential time!)
- **Remember: when computing the blocking time, you need only consider tasks with lower priority.**

Highlights

- We discussed the **unbounded priority inversion problem** and the impact of **blocking** on a high-priority task.
- A high-priority task is blocked when a low-priority task holds a resource (maybe a semaphore) that the high-priority task needs.
- Unbounded priority inversion can be avoided if we use the **priority inheritance protocol**.
- For schedulability analysis, we need to determine the **maximum blocking time** a task can experience. This can be computed by considering the resources used by lower priority tasks.