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**
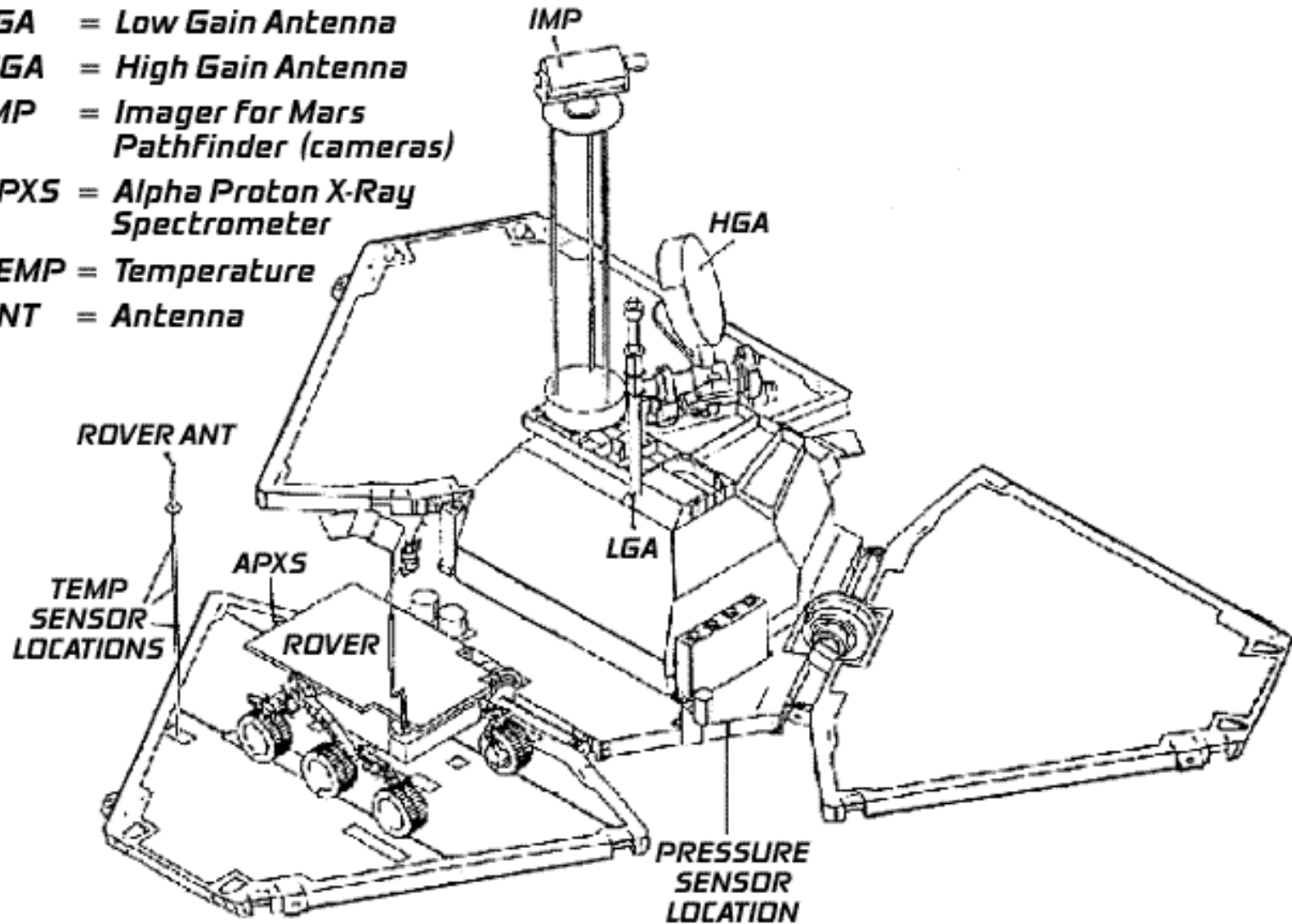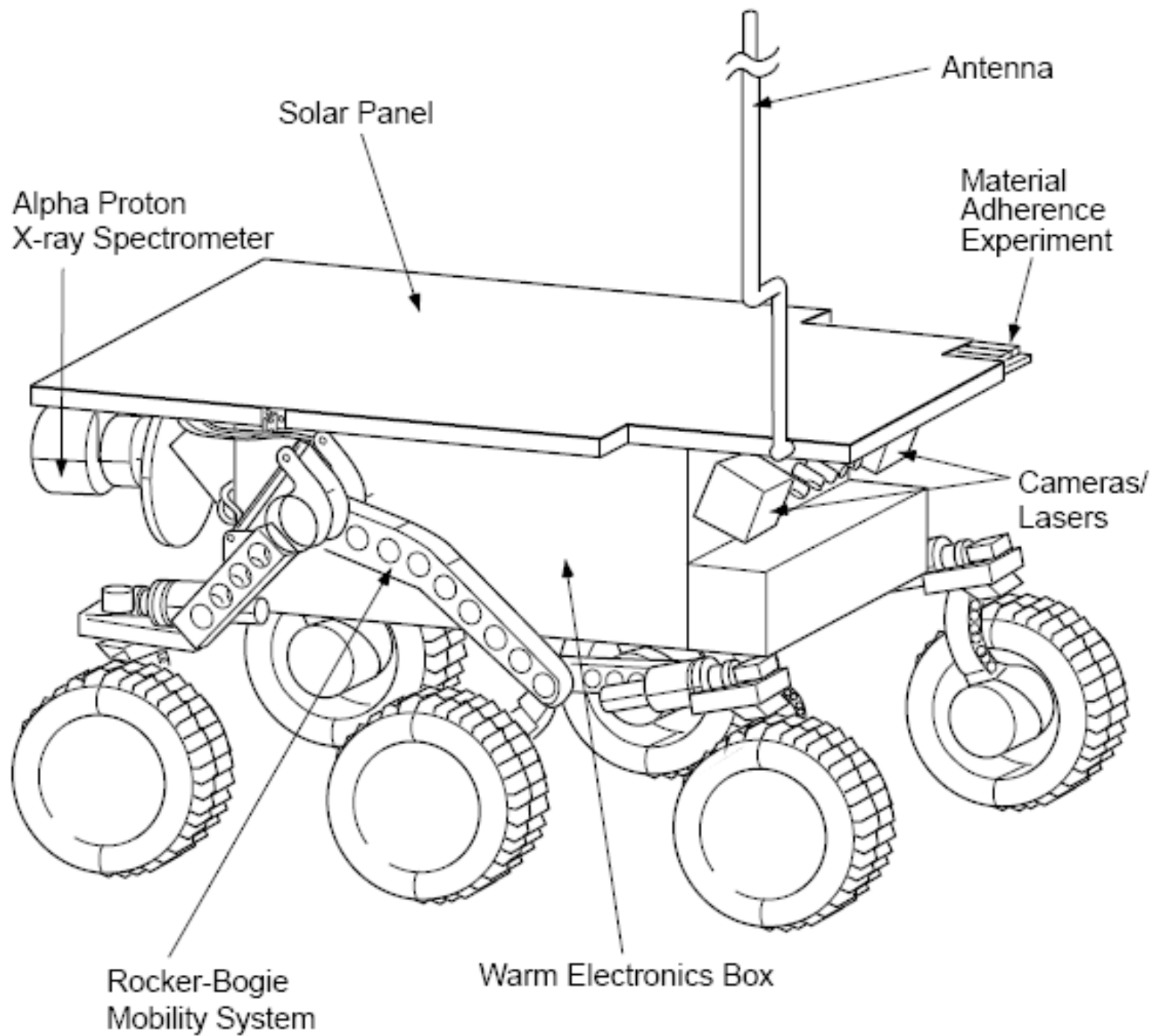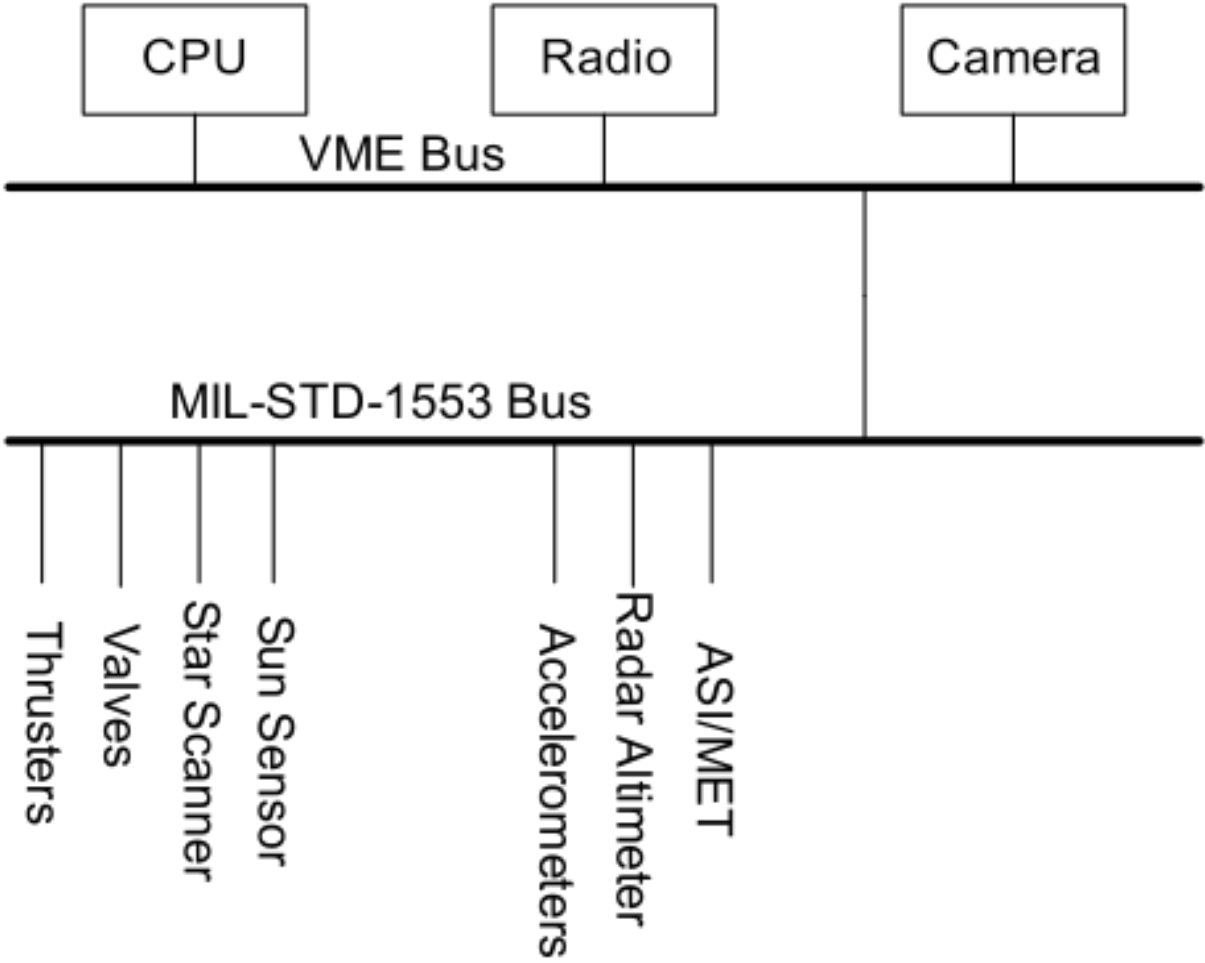
What really happened on Mars?

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

IMP

HGA

ROVER ANT

TEMP
SENSOR
LOCATIONS

APXS

LGA

ROVER

PRESSURE
SENSOR
LOCATION

Source: Jet Propulsion Laboratory, 1994

5

Antenna

Solar Panel

Alpha Proton
X-ray Spectrometer

Material
Adherence
Experiment

Cameras/
Lasers

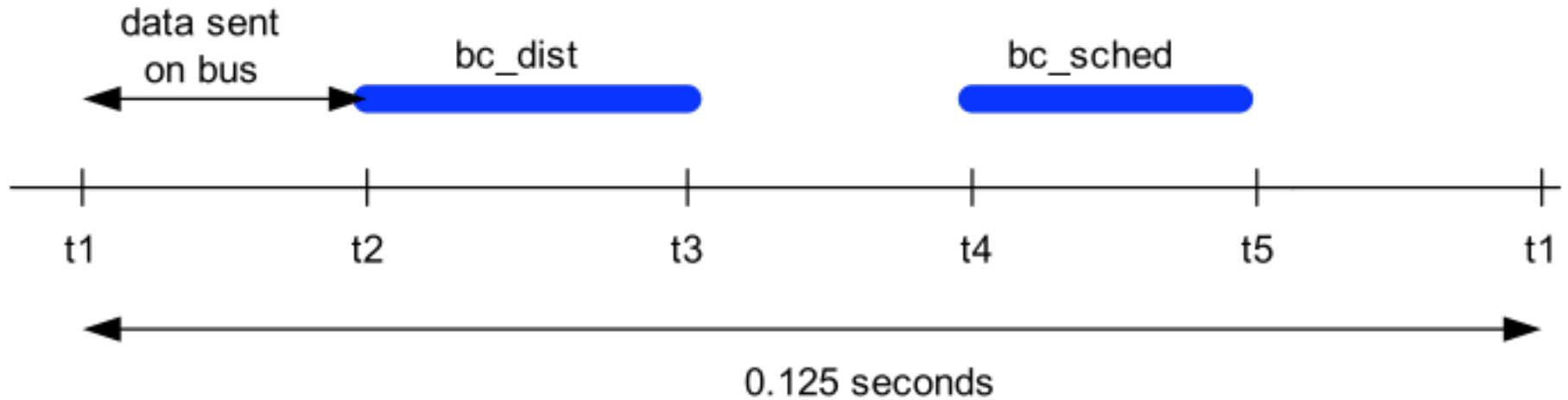Rocker-Bogie
Mobility System

Warm Electronics Box

Mars Pathfinder

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**

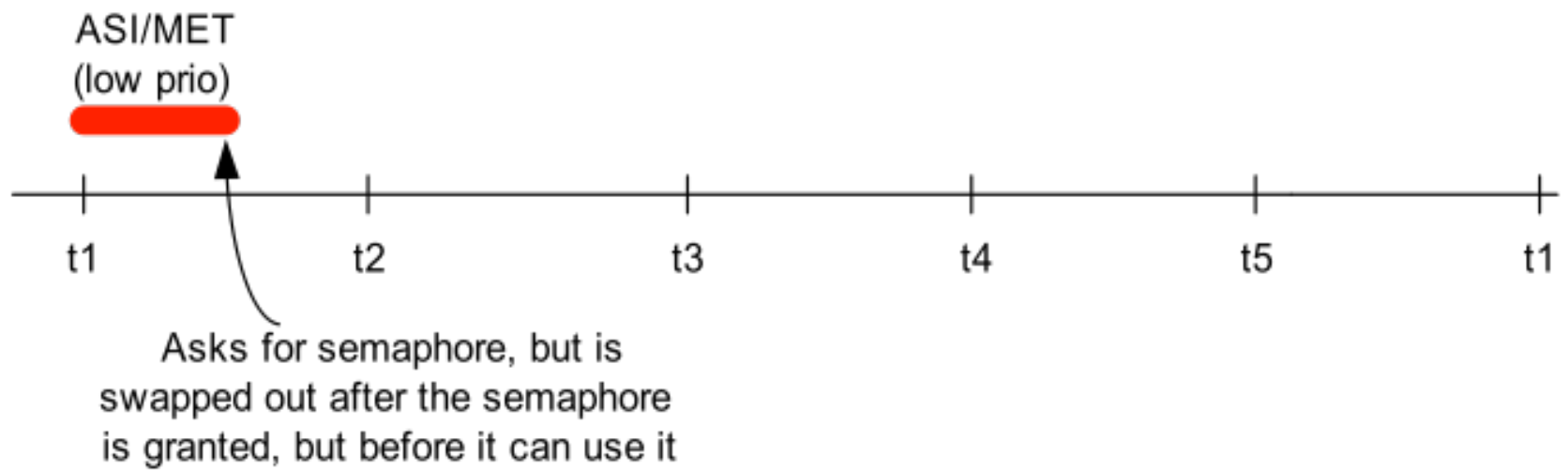 - **low priority task**

Schedule

T = 0.125 seconds

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

if not, it would reset the system (so the Deadline for bc_dist was t4)

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
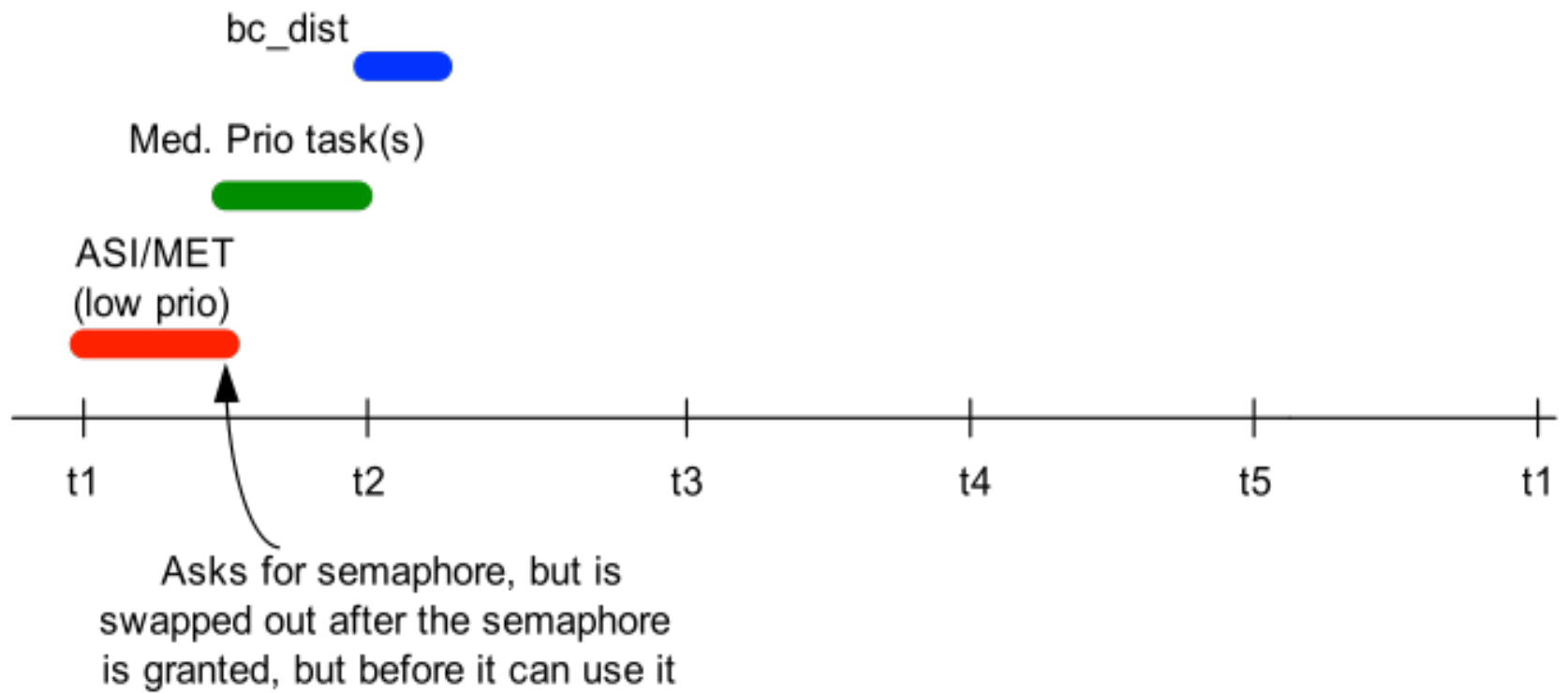
14

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

15

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

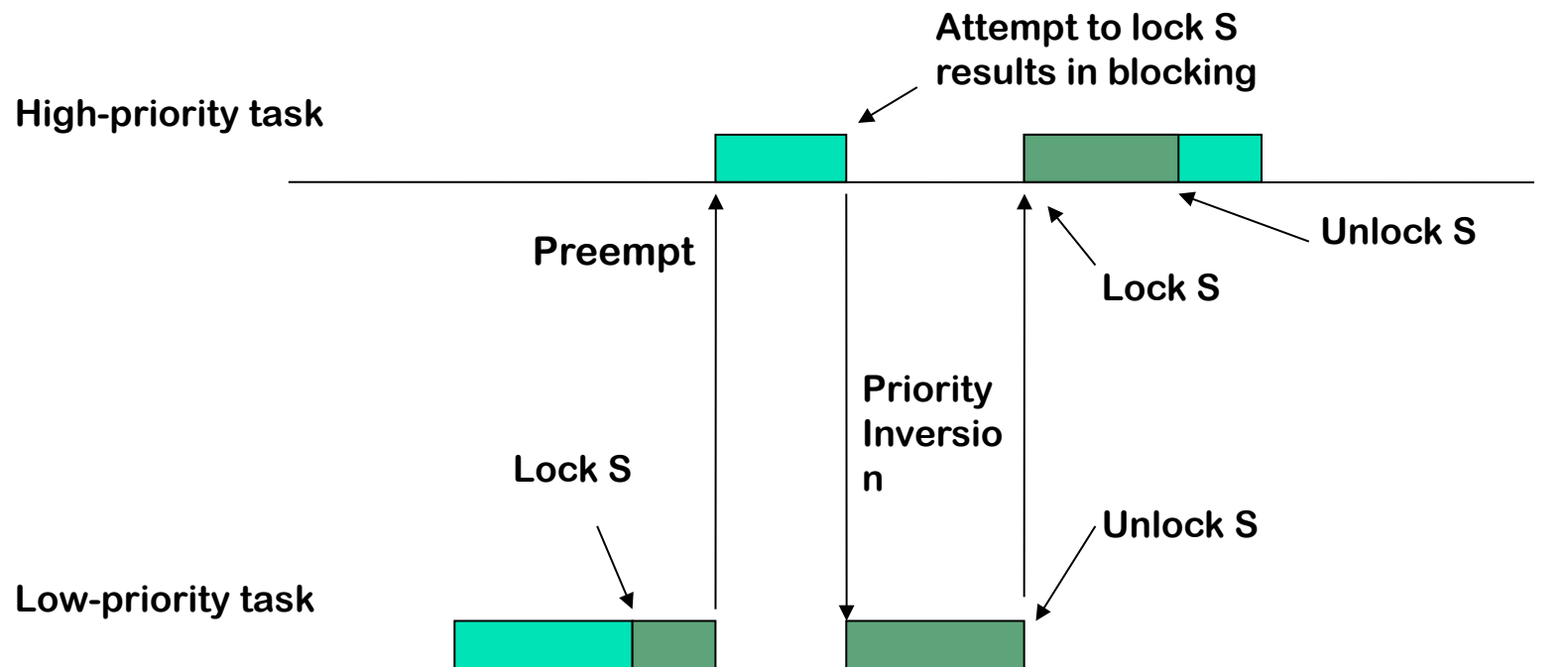Asks for semaphore, but is swapped out after the semaphore is granted, but before it can use it
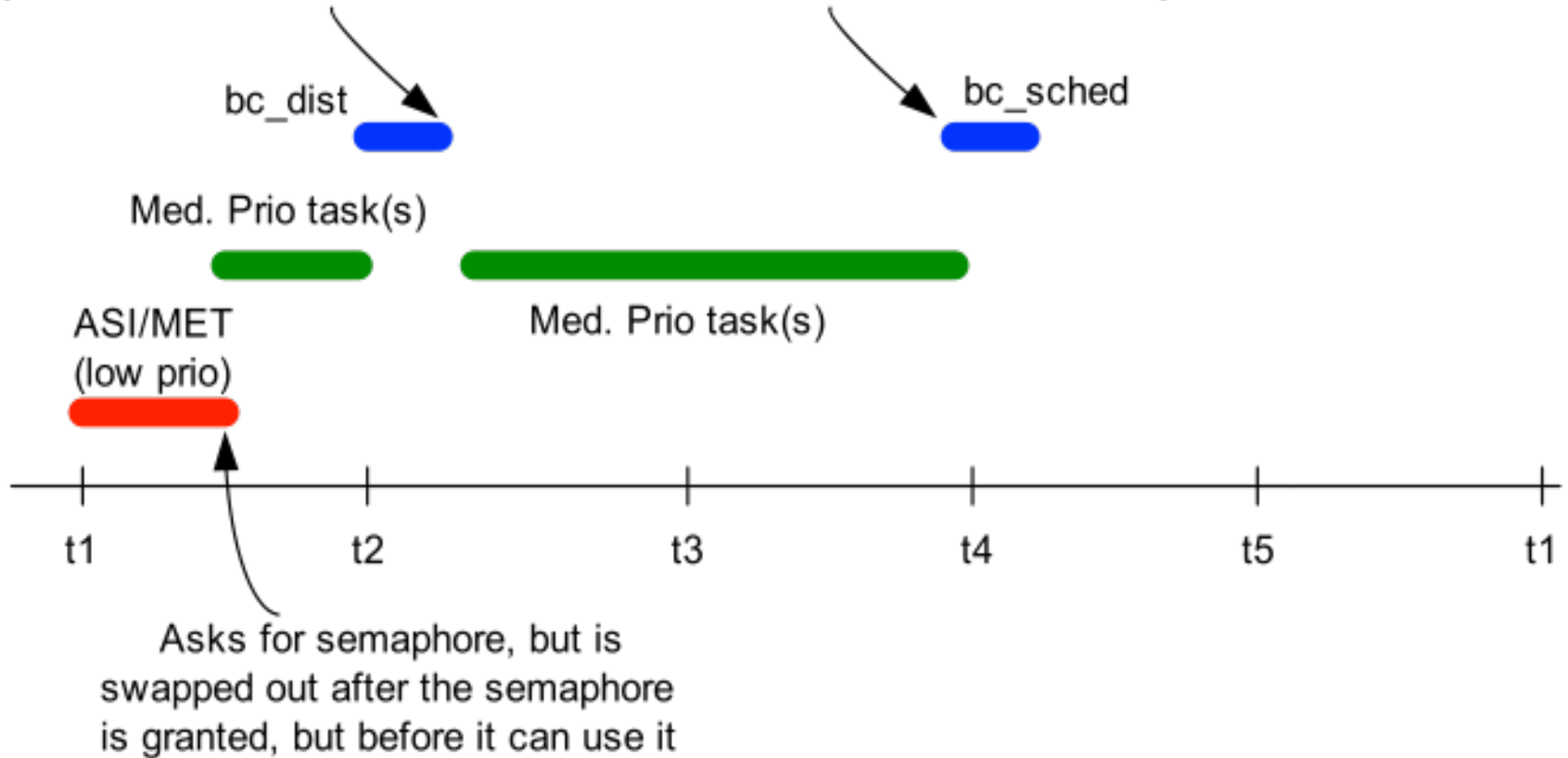
This phenomenon is called Priority Inversion.

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

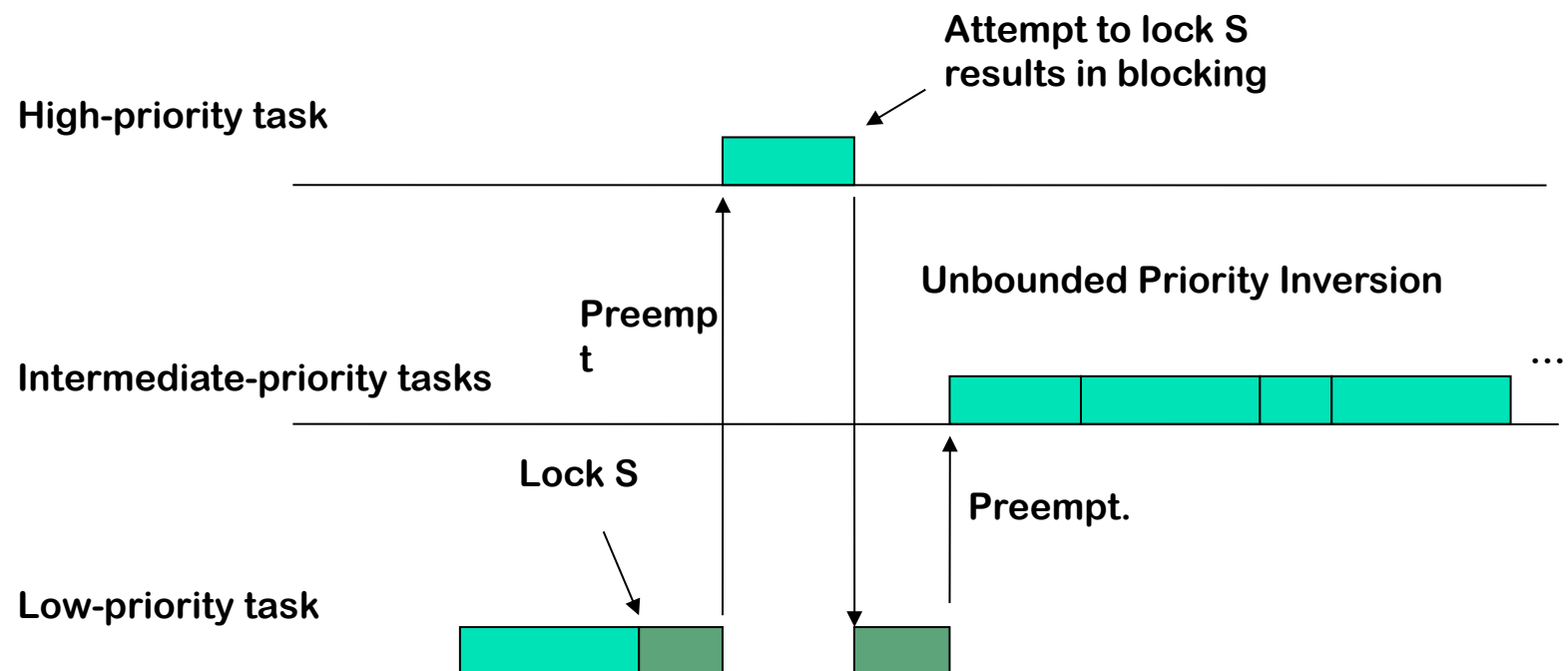 - As it turns out, that may not be the case… so how do we make the priority rules consistent

The priority inheritance protocol

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

**Attempt to lock S
results in blocking**

High-priority task

Preempt

Unlock S

Lock S

**Priority
Inversio
n**

Lock S

Unlock S

Low-priority task

The priority inheritance protocol

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

High-priority task

Attempt to lock S
results in blocking

Intermediate-priority tasks

Unbounded Priority Inversion

Preempt

Preempt.

Lock S

Low-priority task

...

The priority inheritance protocol

- Allow a task to **inherit the priority** of the highest priority task that it is blocking
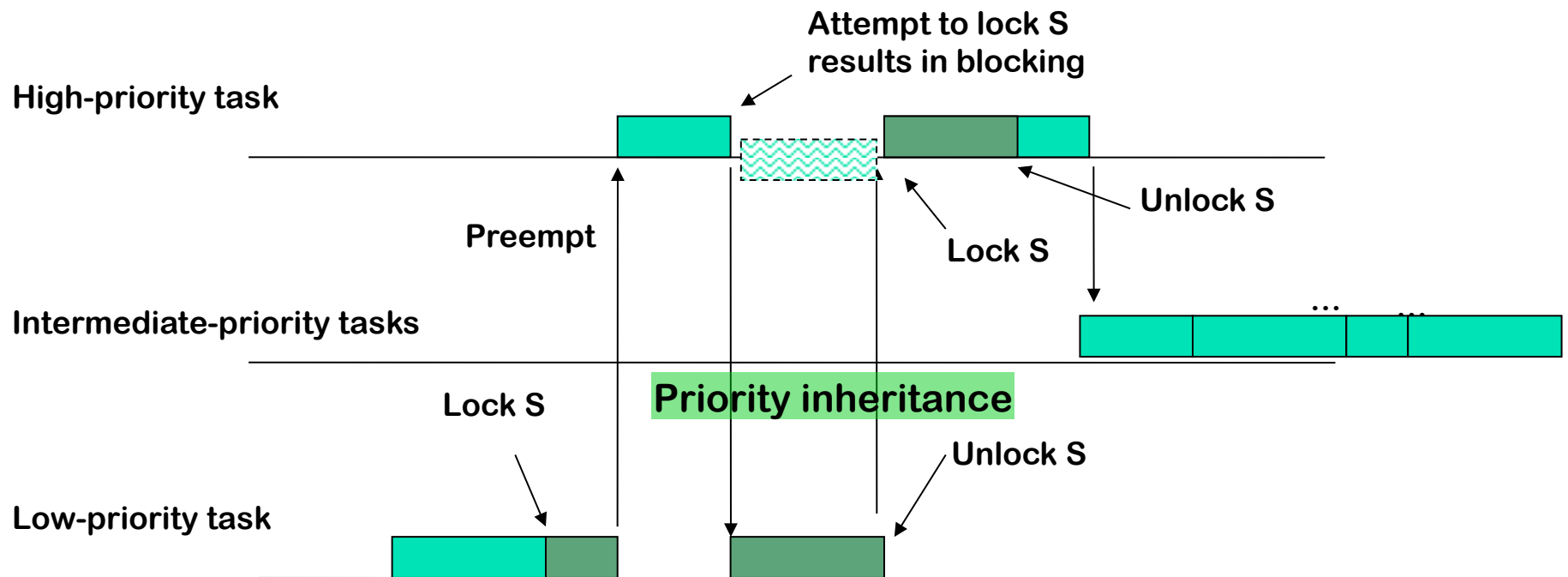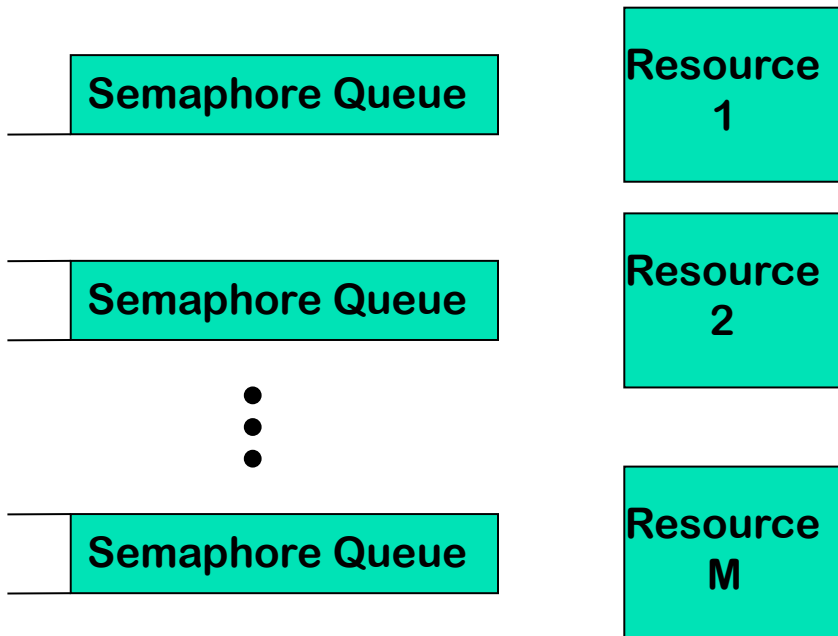
The importance of good theory

- [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 time

- What is the longest time a task can be blocked (waiting for lower priority tasks to release a resource)?

    - Let there be N tasks and M semaphores

    - The length of the largest critical section of Task T_i is B_i

Blocking time

- Consider the instant when a high-priority task arrives

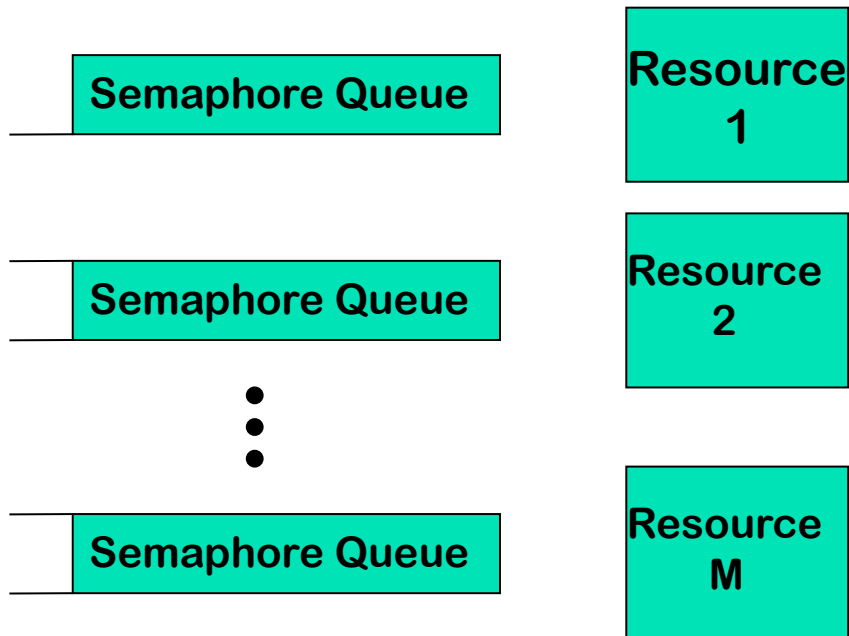 - What is the maximum length of time it may need to wait for a lower priority task to finish?

| Semaphore Queue |

| Semaphore Queue |

•
•
•

| Semaphore Queue |

| Resource 1 |

| Resource 2 |

| Resource M |

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 are of equal length, B

  - Blocking time = *B x min(N, M)*

  - Why?

  - And what if the critical sections are of differing lengths?

| Semaphore Queue |

| Semaphore Queue |

...

| Semaphore Queue |

| Resource 1 |

| Resource 2 |

| Resource M |

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 are of equal length, B

  - Blocking time = *B x min(N, M)*

  - Why?

- And what if the critical sections are of differing lengths?

  - Find the maximum length critical section for each resource

  - Add the top *min(N, M)* sections in size

  - The total priority inversion time experienced by Task T_i is denoted B_i

- **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.