Lecture 22: Real-Time Systems

MAN-KI YOON & ZHONG SHAO
DEPT. OF COMPUTER SCIENCE
YALE UNIVERSITY

What is a Real-Time System?
What is a Real-Time System?
What is a Real-Time System?

Example of Real-Time Systems

- Avionics and automotive systems
- Radar systems
- Factory process control
- Robotics
- Multi-media systems
- ...
Real-Time Systems vs General-Purpose Systems

Real-Time Systems

Meeting timing requirements
(analyzing the worst-case temporal behavior)

General-Purpose Systems

Optimizing average performance

Correctness depends on both functional and temporal aspects

Tasks and Jobs

• **Task**: A sequence of the same type of jobs (e.g., process or thread)
• **Job**: A unit of computation, e.g.,
  • Reading sensor values
  • Computing control commands
• Sometimes task and job are used interchangeably

<table>
<thead>
<tr>
<th>Task 1</th>
<th>Job 1,1</th>
<th>Job 1,2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task 2</td>
<td>Job 2,1</td>
<td>Job 2,2</td>
</tr>
</tbody>
</table>
Periodic Task Model

A task is said to be **periodic** if its inter-arrival time (i.e., period) is a constant.

“Sporadic” task: inter-arrival time is not fixed, but still lower-bounded.
**Periodic Task Model**

A task is said to be *periodic* if its inter-arrival time (i.e., period) is a constant.

![Diagram showing periodic task model](image)

- Task 1:
  - Job \(_{i,j}\)
  - Job \(_{i,j+1}\)
  - Job \(_{i,j+2}\)

(Worst-case) Execution time

- Time

---

**Periodic Task Model**

A task is said to be *periodic* if its inter-arrival time (i.e., period) is a constant.

![Diagram showing periodic task model](image)

- Task 1:
  - Job \(_{i,j}\)
  - Job \(_{i,j+1}\)
  - Job \(_{i,j+2}\)

(Relative) Deadline

- Time
Periodic Task Model

A task is said to be **periodic** if its inter-arrival time (i.e., period) is a constant.

**Hard** deadline vs **Soft** deadline

---

**Periodic Task Model**

A task is said to be **periodic** if its inter-arrival time (i.e., period) is a constant.

**Schedulable** if all jobs meet the relative deadlines.
Periodic Task Model

### Priority and Criticality

- **Priority**: the *order* we execute ready jobs
  - Fixed-priority vs Dynamic-priority

- **Criticality**: the *penalty* if a task misses its deadline
  - Usually qualitative

- How do we assign priorities to tasks or jobs?
Should we give a higher priority to Task 1? Or Task 2?

Let's say we give a higher priority to Task 1. What happens?
Priority and Criticality

Case 1: Priority(Task 1) > Priority(Task 2)

Deadline miss!

Task 1: Critical task
Task 2: Non-critical task

Case 2: Priority(Task 1) < Priority(Task 2)

Both tasks are schedulable!
Priority and Criticality

• Importance (i.e., criticality) *may or may not* correspond to scheduling priority.
  - Priority is derived from timing requirements
  - Importance matters *only when* tasks can be scheduled without missing deadlines.

Notations

- Task Utilization: \( U_i = \frac{C_i}{p_i} \)
- Period: \( p_i \)
- Task: \( \tau_i \)
- Job: \( J_{i,j} \)
- Worst-case Execution Time: \( C_i \)
- (Relative) Deadline: \( D_i \)
Real-Time Scheduling Algorithms

• **Rate-Monotonic (RM)**
  - Assign higher priority to tasks that have higher-rate (=shorter period)
  - Optimal fixed-priority scheduling

• **Earliest Deadline First (EDF)**
  - Assign higher priority to jobs that have earlier relative deadline
  - Optimal dynamic-priority scheduling

What does it mean by ‘optimal’ scheduling?
Real-Time Scheduling Algorithms

- **Rate-Monotonic (RM)**
  - Assign higher priority to *tasks* that have higher rate (=shorter period)
  - Optimal fixed-priority scheduling

- **Earliest Deadline First (EDF)**
  - Assign higher priority to *jobs* that have earlier relative deadline
  - Optimal dynamic-priority scheduling

**What does it mean by ‘optimal’ scheduling?**

If a task set is not schedulable by the optimal scheduling algorithm, no other scheduling algorithms can schedule the task set.

---

**Rate-Monotonic (RM)**

\[ \tau_1 := (p_1 = 5, C_1 = 2) \]
\[ \tau_2 := (p_2 = 9, C_2 = 4) \]

Which one has a higher priority?
Rate-Monotonic (RM)

\[ \tau_1 := (p_1 = 5, C_1 = 2) \]
\[ \tau_2 := (p_2 = 9, C_2 = 4) \]
Rate-Monotonic (RM)

\( \tau_1 := (p_1 = 5, C_1 = 2) \)
\( \tau_2 := (p_2 = 9, C_2 = 4) \)
Rate-Monotonic (RM)

\[ \tau_1 := (p_1 = 5, C_1 = 2) \]
\[ \tau_2 := (p_2 = 9, C_2 = 4) \]

If response time \( \leq \) deadline, the job is \textit{schedulable}
Earliest Deadline First (EDF)

\[ \tau_1 := (p_1 = 5, C_1 = 2) \]
\[ \tau_2 := (p_2 = 9, C_2 = 4) \]
(assume deadline = period)

Which one has a higher priority?
Earliest Deadline First (EDF)

$\tau_1 := (p_1 = 5, C_1 = 2)$
$\tau_2 := (p_2 = 9, C_2 = 4)$
(assume deadline = period)

Q: What happens next?
Earliest Deadline First (EDF)

\( \tau_1 := (p_1 = 5, C_1 = 2) \)
\( \tau_2 := (p_2 = 9, C_2 = 4) \)
(assume deadline = period)

\( \begin{align*}
\tau_1 &:= (p_1 = 5, C_1 = 2) \\
\tau_2 &:= (p_2 = 9, C_2 = 4) \\
\text{(assume deadline = period)}
\end{align*} \)
Earliest Deadline First (EDF)

\[ \tau_1 := (p_1 = 5, C_1 = 2) \]
\[ \tau_2 := (p_2 = 9, C_2 = 4) \]
(assume deadline = period)

\( \tau_1 \)

\( \tau_2 \)

Earliest Deadline First (EDF)

\[ \tau_1 := (p_1 = 5, C_1 = 2) \]
\[ \tau_2 := (p_2 = 9, C_2 = 4) \]
(assume deadline = period)

\( \tau_1 \)

\( \tau_2 \)
Earliest Deadline First (EDF)

\[ \tau_1 := (p_1 = 5, C_1 = 2) \]
\[ \tau_2 := (p_2 = 9, C_2 = 4) \]
(assume deadline = period)

Schedulability Analysis

*How can we know if a set of periodic tasks is schedulable?*
Schedulability Analysis

- How can we know if a set of periodic tasks is schedulable?
  - Exact test
  - Utilization bound test

Exact Test

- A.k.a. Response time analysis
- For fixed-priority scheduling algorithms
- A task is said to be schedulable if and only if its **worst-case response time** is not greater than its deadline

- When is the worst-case?
Exact Test

- A.k.a. Response time analysis
- For fixed-priority scheduling algorithms
- A task is said to be schedulable if and only if its \textbf{worst-case response time} is not greater than its deadline

\[
\begin{align*}
\text{Release} & \quad \text{Response time} \quad \text{Finish} \\
& \quad \text{Deadline}
\end{align*}
\]

- When is the worst-case?
  - When all higher-priority tasks are released at the same time (\textit{Critical instant theorem} [Liu73])

---


---

\[
\begin{align*}
\tau_{i+1}^{k+1} &= C_i + \sum_{j=1}^{i-1} \left( \frac{r_{i}^{k}}{P_j} \right) C_j \\
\text{where} \quad r_{i}^{0} &= \sum_{j=1}^{i} C_j
\end{align*}
\]

- Iterative method
- Tasks are ordered according to their priority; \( T_1 \) has the highest priority
- If \( r_{i}^{k+1} > D_i \) \quad \text{\Rightarrow \ Schedulable}
- If \( r_{i}^{k+1} = r_{i}^{k} \leq D_i \) \quad \text{for some} \ k \quad \text{\Rightarrow \ Schedulable}
- Test task-by-task. If any task fails the exact test, the task set is unschedulable
Exact Test

\[ r_i^{k+1} = C_i + \sum_{j=1}^{i-1} \left[ \frac{x_j^k}{p_j} \right] C_j \quad , \quad r_i^0 = \sum_{j=1}^{i} C_j \]

\( r_1^0 = \sum_{j=1}^{1} C_j = C_1 = 4 \)

\( r_2^0 = \sum_{j=1}^{2} C_j = C_1 + C_2 = 4 + 4 = 8 \)

\( r_3^0 = \sum_{j=1}^{3} C_j = C_1 + C_2 + C_3 = 4 + 4 + 10 = 18 \)

\( \tau_1 := (p_1 = 10, c_1 = 4) \)

\( \tau_2 := (p_2 = 15, c_2 = 4) \)

\( \tau_3 := (p_3 = 35, c_3 = 10) \)

(assume deadline = period)
Exact Test

\[ r_{i}^{k+1} = C_i + \sum_{j=1}^{i-1} \left[ \frac{r_j^{k}}{p_j} \right] C_j \quad , \quad r_{i}^{0} = \sum_{j=1}^{i} C_j \]

\[ \tau_1 := (p_1 = 10, c_1 = 4) \]

\[ \tau_2 := (p_2 = 15, c_2 = 4) \]

\[ \tau_3 := (p_3 = 35, c_3 = 10) \]

New jobs of Task 1 and 2 arrive before Task 3’s job finishes. -> Additional preemptions

\[ r_3^{1} = 10 + \frac{2}{10} C_j = 10 + \frac{18}{10} + \frac{18}{15} = 10 + 2 \cdot 4 + 2 \cdot 4 = 10 + 8 + 8 = 26 \]

51
Exact Test

\[ r_i^{k+1} = C_i + \sum_{j=1}^{i-1} \left[ \frac{r_j^k}{p_j} \right] C_j \]

, \[ r_i^0 = \sum_{j=1}^{i} C_j \]

\( \tau_1 := (p_1 = 10, c_1 = 4) \)

\( \tau_2 := (p_2 = 15, c_2 = 4) \)

\( \tau_3 := (p_3 = 35, c_3 = 10) \)

New job of Task 1 arrives before Task 3’s job finishes. \( \rightarrow \) Additional preemption

\[ \tau_3^1 = 26 \]

Q: Compute \( r_3^2 \)
Exact Test

\[ r_i^{k+1} = C_i + \sum_{j=1}^{i-1} \left[ \frac{r_j^k}{p_j} \right] C_j \quad , \quad r_i^0 = \sum_{j=1}^{i} C_j \]

\[ r_3^3 = 10 + \sum_{j=1}^{2} \left[ \frac{r_j^2}{p_j} \right] C_j = 10 + \left[ \frac{26}{10} \right] 1 + \left[ \frac{26}{15} \right] 4 = 10 + 3 \cdot 4 + 2 \cdot 4 = 10 + 12 + 8 = 30 \]

\[ r_3^2 = 10 + \sum_{j=1}^{2} \left[ \frac{r_j^1}{p_j} \right] C_j = 10 + \left[ \frac{26}{10} \right] 1 + \left[ \frac{26}{15} \right] 4 = 10 + 3 \cdot 4 + 2 \cdot 4 = 10 + 12 + 8 = 30 \]

\[ r_3^1 = 10 + \sum_{j=1}^{2} \left[ \frac{r_j^0}{p_j} \right] C_j = 10 + \left[ \frac{30}{10} \right] 1 + \left[ \frac{30}{15} \right] 4 = 10 + 3 \cdot 4 + 2 \cdot 4 = 10 + 12 + 8 = 30 \]
Exact Test

\[ r_i^{k+1} = C_i + \sum_{j=1}^{i-1} \left\lfloor \frac{i}{p_j} \right\rfloor C_j, \quad r_i^0 = \sum_{j=1}^{i} C_j \]

\[ \tau_1 := (p_1 = 10, c_1 = 4) \]
\[ \tau_2 := (p_2 = 15, c_2 = 4) \]
\[ \tau_3 := (p_3 = 35, c_3 = 10) \]

Utilization Bound Test

Task Utilization

\[ U_i = \frac{C_i}{p_i} \]

Processor Utilization (n=number of tasks)

\[ U = \sum_{i=1}^{n} U_i = \sum_{i=1}^{n} \frac{C_i}{p_i} \]

Utilization Bound \((U_b)\)

Any task \( \tau_i \in \{\tau_1, \tau_2, \ldots, \tau_n\} \) is guaranteed to be schedulable if \( U \leq U_b \)

\( U_b \) depends on the scheduling algorithm, # of tasks, availability on timing information, ...
RM Utilization Bound

A set of \( n \) tasks is schedulable under RM scheduling if (see [Liu73] for proof)

\[
U \leq U_{RM}(n) = n(2^{1/n} - 1)
\]

Example

<table>
<thead>
<tr>
<th>Task</th>
<th>( C_i ) (Execution Time)</th>
<th>( p_i ) (Period)</th>
<th>( U_i ) (Utilization)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task 1</td>
<td>20</td>
<td>100</td>
<td>( ? )</td>
</tr>
<tr>
<td>Task 2</td>
<td>40</td>
<td>150</td>
<td>( ? )</td>
</tr>
<tr>
<td>Task 3</td>
<td>100</td>
<td>350</td>
<td>( ? )</td>
</tr>
</tbody>
</table>


1) Check the schedulability of \{task 1\}:

\[
U_1 = 0.2 < U_{RM}(1) = 1
\]
RM Utilization Bound

A set of \( n \) tasks is schedulable under RM scheduling if (see [Liu73] for proof)

\[
U \leq U_{RM}(n) = n(2^{1/n} - 1)
\]

Example

<table>
<thead>
<tr>
<th>C (Execution Time)</th>
<th>p (Period)</th>
<th>U (Utilization)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task 1</td>
<td>20</td>
<td>100</td>
</tr>
<tr>
<td>Task 2</td>
<td>40</td>
<td>150</td>
</tr>
<tr>
<td>Task 3</td>
<td>100</td>
<td>350</td>
</tr>
</tbody>
</table>

2) Check the schedulability of \{task 1, task 2\}:

\[
U_1 + U_2 \approx U_{RM}(2) = 0.828
\]
RM Utilization Bound

A set of $n$ tasks is schedulable under RM scheduling if (see [Liu73] for proof)

$$U \leq U_{RM}(n) = n (2^{1/n} - 1)$$

Example

<table>
<thead>
<tr>
<th></th>
<th>$C_i$ (Execution Time)</th>
<th>$p_i$ (Period)</th>
<th>$U_i$ (Utilization)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task 1</td>
<td>20</td>
<td>100</td>
<td>0.200</td>
</tr>
<tr>
<td>Task 2</td>
<td>40</td>
<td>150</td>
<td>0.267</td>
</tr>
<tr>
<td>Task 3</td>
<td>100</td>
<td>350</td>
<td>0.286</td>
</tr>
</tbody>
</table>

3) Check the schedulability of {task 1, task 2, task 3}:

$$U_1 + U_2 + U_3 \approx 0.753 < U_{RM}(3) = 0.780$$

Q: What if $C_1 = 40$?
RM Utilization Bound

A set of $n$ tasks is schedulable under RM scheduling if (see [Liu73] for proof)

$$ U \leq U_{RM}(n) = n(2^{1/n} - 1) $$

Example

<table>
<thead>
<tr>
<th>Task</th>
<th>$C_i$ (Execution Time)</th>
<th>$p_i$ (Period)</th>
<th>$U_i$ (Utilization)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task 1</td>
<td>40</td>
<td>100</td>
<td>0.400</td>
</tr>
<tr>
<td>Task 2</td>
<td>40</td>
<td>150</td>
<td>0.267</td>
</tr>
<tr>
<td>Task 3</td>
<td>100</td>
<td>350</td>
<td>0.286</td>
</tr>
</tbody>
</table>

Q: What if $C_1=40$?

$U_1 + U_2 + U_3 \approx 0.953 > U_{RM}(3) = 0.780$

Q: Are the tasks unschedulable?
RM Utilization Bound

A set of \( n \) tasks is schedulable under RM scheduling if (see [Liu73] for proof)

\[
U \leq U_{RM}(n) = n(2^{1/n} - 1)
\]

Example

<table>
<thead>
<tr>
<th>Task</th>
<th>( C_i ) (Execution Time)</th>
<th>( p_i ) (Period)</th>
<th>( U_i ) (Utilization)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task 1</td>
<td>40</td>
<td>100</td>
<td>0.400</td>
</tr>
<tr>
<td>Task 2</td>
<td>40</td>
<td>150</td>
<td>0.267</td>
</tr>
<tr>
<td>Task 3</td>
<td>100</td>
<td>350</td>
<td>0.286</td>
</tr>
</tbody>
</table>

Q: What if \( C_1 = 40 \)?

\[
U_1 + U_2 + U_3 \approx 0.953 > U_{RM}(3) = 0.780
\]

Q: Are the tasks unschedulable? A: Not necessarily. Need to do the exact test!

---

RM Utilization Bound

A set of \( n \) tasks is schedulable under RM scheduling if (see [Liu73] for proof)

\[
U \leq U_{RM}(n) = n(2^{1/n} - 1)
\]

Example

<table>
<thead>
<tr>
<th>Task</th>
<th>( C_i ) (Execution Time)</th>
<th>( p_i ) (Period)</th>
<th>( U_i ) (Utilization)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task 1</td>
<td>40</td>
<td>100</td>
<td>0.400</td>
</tr>
<tr>
<td>Task 2</td>
<td>40</td>
<td>150</td>
<td>0.267</td>
</tr>
<tr>
<td>Task 3</td>
<td>100</td>
<td>350</td>
<td>0.286</td>
</tr>
</tbody>
</table>

Q: What is the worst-case response time of Task 3?

\[
x_k^{i+1} = C_i + \sum_{j=1}^{k-1} \frac{r_j}{p_j} C_j \quad , \quad x_0^i = \sum_{j=1}^{i} C_j
\]
RM Utilization Bound

Utilization bound test is a **sufficient** condition
- If $U \leq U_{RM}(n)$, the task set is guaranteed to be schedulable by RM.
- $U > U_{RM}(n)$ does not necessarily mean the task set is unschedulable
  - Need to perform an exact test

UB for any $n$

$$U_{RM} = \lim_{n \to \infty} U_{RM}(n) = \ln 2 \approx 0.693$$

**Q: What does this mean?**
EDF Utilization Bound

A set of tasks is schedulable under EDF scheduling if and only if

\[ U \leq U_{EDF} = 1 \]

- Sufficient and necessary condition
- Does not depend on \# of tasks

<table>
<thead>
<tr>
<th>C (Execution Time)</th>
<th>p (Period)</th>
<th>U (Utilization)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task 1</td>
<td>40</td>
<td>100</td>
</tr>
<tr>
<td>Task 2</td>
<td>40</td>
<td>150</td>
</tr>
<tr>
<td>Task 3</td>
<td>100</td>
<td>350</td>
</tr>
</tbody>
</table>

\[ U_1 + U_2 + U_3 \approx 0.953 < U_{EDF} \]

RM vs EDF

EDF’s utilization bound is 1 while RM’s is less than 1
- RM may not fully utilize the CPU

Why do we need RM?
RM vs EDF

EDF’s utilization bound is 1 while RM’s is less than 1
- RM may not fully utilize the CPU

Why do we need RM?
- Simpler implementation
  - Priorities do not change
  - Some tasks may not have deadlines
- EDF is unpredictable
  - Domino effect during overloaded situation
  - A low critical task which overruns but has an earlier deadline can delay a high critical task.
- FAA (Federal Aviation Administration) and EASA (European Aviation Safety Agency) forbid the use of EDF
- However, EDF is desirable for budget-enforcing real-time scheduler

Priority Inversion

So far, tasks are assumed to be independent

What if tasks share data?
- Synchronization!

```
semaphore->P();
// critical section goes here
semaphore->V();
```

- But it can be a source of priority inversion
Priority Inversion

When a high priority task is delayed by a low priority task

Assume these two tasks share a critical section.

High Priority

Low Priority

Time

semaphore->P();
// critical section goes here
semaphore->V();
Priority Inversion

When a high priority task is delayed by a low priority task

```
// critical section goes here
semaphore->V();
```

```
// critical section goes here
semaphore->V();
```
Unbounded Priority Inversion

High Priority

Medium Priority

Medium Priority

Low Priority

Time

Normal Execution
Critical Section

79

Unbounded Priority Inversion

High Priority

Medium Priority

Medium Priority

Low Priority

Time

Normal Execution
Critical Section

80
Unbounded Priority Inversion

- High Priority
- Medium Priority
- Medium Priority
- Low Priority

Time

Normal Execution
Critical Section
Unbounded Priority Inversion

It actually happened on Mars!

What really happened on Mars?

The Mars Pathfinder mission was widely predicted as “ réussi” in the early days after its July 4th, 1997 landing on the Martian surface. “Success” included its automated driving, “by flying over the Martian surface controlled by software, demonstrating the 6-wheeler rover, and proving out demonstrating transmission data back to Earth, including the precision point that even a small boulder on the Mars Pathfinder rover did not show up in the radar image of the landing site.”

The rover was equipped with a 900 m3/min spacecraft gas generator, a laser detection (LIDAR) package, and a small robotic arm that deployed the radar to the surface. The rover also contained a weather station that was able to collect and transmit real-time weather data to Earth.

However, a few days into the mission, not long after Pathfinder started gathering meteorological data, the spacecraft began experiencing total system resets, each resulting in loss of data. The press reported these failures.
Unbounded Priority Inversion

It actually happened on Mars!

**What really happened on Mars?**

From: Mike Jones  
Date: Friday, December 31, 1996 11:42 AM  
Subject: What really happened on Mars?

THE PROBLEM

It actually happened on Mars! NASA Mars Pathfinder (1997)

It actually happened on Mars!

NASA Mars Pathfinder (1997)

The Mars Pathfinder mission was widely publicized as "Titanic" in the early days after its July 4, 1997 landing on the Martian surface. Everyone wanted to see roving, exploring the Martian terrain, and gathering and transmitting meteorological data back to Earth. The primary goal was to develop a sustainable human mission to Mars, including the pressurized rover that was sent to the moon in the late 1990s. (Note: the pressurized rover is not the same as the rover that was actually sent to Mars.)

The Pathfinder rover was a small mobile surface platform that was designed to gather meteorological data and other information from the Martian surface. The rover was equipped with a number of scientific instruments, including a camera, a spectrometer, and a seismometer.

However, the rover遇到了 some unexpected problems. During its first few days on the surface, the rover encountered a number of technical difficulties, including a malfunction in its communication system.

Unbounded Priority Inversion

It actually happened on Mars!

NASA Mars Pathfinder (1997)

L is executing, accessing the bus.

M can't access the bus. It is blocked by L preempts. M is further blocked.

Watchdog timer notices that L has not executed for some time. Hence, it resets the system!

---

voluminous data back to Earth, including the 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. The press reported these failures.

---

voluminous data back to Earth, including the 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. The press reported these failures.

---

voluminous data back to Earth, including the 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. The press reported these failures.

---

voluminous data back to Earth, including the 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. The press reported these failures.

---

voluminous data back to Earth, including the 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. The press reported these failures.

---

voluminous data back to Earth, including the 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. The press reported these failures.
Unbounded Priority Inversion

It actually happened on Mars!

How was the problem corrected?

“Priority Inheritance Protocol (PIP)”
VxWorks had PIP, but it had been turned off for the mutex!


Priority Inheritance Protocol

Low priority task inherits the highest priority of all the blocked tasks
- This keeps medium tasks from delaying the low priority task that is in a critical section
Priority Inheritance Protocol

Low priority task inherits the highest priority of all the blocked tasks
- This keeps medium tasks from delaying the low priority task that is in a critical section

Task 1 (High Priority)  Task 2 (Medium Priority)  Task 3 (Low Priority)

Task 3 inherits Priority(Task 1)

Task 2 can't preempt Task 3!

Priority Inheritance Protocol

Low priority task inherits the highest priority of all the blocked tasks
- This keeps medium tasks from delaying the low priority task that is in a critical section

Task 1 (High Priority)  Task 2 (Medium Priority)  Task 3 (Low Priority)

Task 3 inherits Priority(Task 1)
**Priority Inheritance Protocol**

Low priority task inherits the highest priority of all the blocked tasks
- This keeps medium tasks from delaying the low priority task that is in a critical section

![Diagram of Priority Inheritance Protocol]

Task 1 (High Priority)

Task 2 (Medium Priority)

Task 3 (Low Priority)

Task 3 inherits Priority(Task 1) Returns to the normal priority

Time

93
Priority Inheritance Protocol

A job \( J \) can be blocked for at most \( \min(n, m) \) times where
- \( n \) = number of lower priority jobs that could block \( J \)
- \( m \) = number of distinct semaphores that can be used to block \( J \)

But chained blocking and deadlock can happen under PIP
- Solution: Priority Ceiling Protocol (PCP)

https://www.youtube.com/watch?feature=oembed&v=Y6v98S1BHek
Priority Ceiling Protocol

**Priority ceiling of a semaphore**
- The priority of the highest priority task that may use the semaphore

**Key Idea**
- A job J is allowed to enter a critical section only if its priority is higher than all priority ceilings of the semaphores currently locked by jobs other than J
  - Thus, it can never be blocked by lower priority jobs until its completion!
- When a job gets a semaphore, PCP guarantees that this job will get all the semaphores that it ever needs.
  - Hence, PCP prevents chained blocking and deadlock.

For more information, see