What is a Real-Time System?
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)

Correctness depends on both functional and **temporal** aspects

General-Purpose Systems

Optimizing **average performance**

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
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 of Periodic Task Model]

(Worst-case) Execution 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 with job arrivals and deadlines]

**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

![Diagram showing periodic task model with job arrivals and deadlines]

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

<table>
<thead>
<tr>
<th>Seq.</th>
<th>Function</th>
<th>CPU</th>
<th>Period</th>
<th>Deadline</th>
<th>Importance</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Aircraft flight data</td>
<td>8 ms.</td>
<td>65 ms.</td>
<td>critical</td>
<td></td>
</tr>
<tr>
<td>3.2</td>
<td>Steering</td>
<td>6</td>
<td>80</td>
<td>critical</td>
<td></td>
</tr>
<tr>
<td>3.3</td>
<td>Radar control</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.3.1</td>
<td>Radar beacon</td>
<td>2</td>
<td>60</td>
<td>critical</td>
<td></td>
</tr>
<tr>
<td>3.3.2</td>
<td>Radar tracking</td>
<td>2</td>
<td>40</td>
<td>critical</td>
<td></td>
</tr>
<tr>
<td>3.3.3</td>
<td>Radar beacon or</td>
<td>2</td>
<td>200</td>
<td>critical</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Radar tracking or</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>missile tracking</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.4</td>
<td>Designate target</td>
<td>1</td>
<td>40</td>
<td>critical</td>
<td></td>
</tr>
<tr>
<td>3.5</td>
<td>Confirm designation</td>
<td>1</td>
<td>200</td>
<td>critical</td>
<td></td>
</tr>
<tr>
<td>3.6</td>
<td>Target tracking</td>
<td>2</td>
<td>40</td>
<td>critical</td>
<td></td>
</tr>
<tr>
<td>3.6.2</td>
<td>Target acquisition</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.6.3</td>
<td>Target tracking</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.6.4</td>
<td>Target acquisition</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.7</td>
<td>Weapon control</td>
<td>1</td>
<td>200</td>
<td>essential</td>
<td></td>
</tr>
<tr>
<td>3.7.1</td>
<td>Weapon selection processing</td>
<td>2</td>
<td>400</td>
<td>essential</td>
<td></td>
</tr>
<tr>
<td>3.7.2</td>
<td>AutoCADIP logics</td>
<td>1</td>
<td>200</td>
<td>critical</td>
<td></td>
</tr>
<tr>
<td>3.7.3</td>
<td>Weapon trajectory</td>
<td>5</td>
<td>400</td>
<td>essential</td>
<td></td>
</tr>
<tr>
<td>3.8</td>
<td>Weapon release</td>
<td>1</td>
<td>10</td>
<td>critical</td>
<td></td>
</tr>
<tr>
<td>3.9</td>
<td>HMD display</td>
<td>6</td>
<td>52</td>
<td>essential</td>
<td></td>
</tr>
<tr>
<td>3.10</td>
<td>MPO:HMD display</td>
<td>6</td>
<td>52</td>
<td>essential</td>
<td></td>
</tr>
<tr>
<td>3.11</td>
<td>MPO display</td>
<td>6</td>
<td>52</td>
<td>essential</td>
<td></td>
</tr>
<tr>
<td>3.12</td>
<td>MPO button response</td>
<td>1</td>
<td>200</td>
<td>background</td>
<td></td>
</tr>
</tbody>
</table>

Source: Generic Avionics Software Specification

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?
Priority and Criticality

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)

- Task 1: Critical task (CPSC422 Assignment)
- Task 2: Non-critical task (Instagram)

**Deadline miss!**

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

- Task 1: Critical task (CPSC422 Assignment)
- Task 2: Non-critical task (Instagram)

**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: $C_i$
- Deadline: $D_i$
- Time
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 a shorter period.
  - Optimal fixed-priority scheduling.

- Earliest Deadline First (EDF)
  - Assign higher priority to jobs that have an 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?

\[ \tau_1 \] and \[ \tau_2 \] are both tasks with different periods and execution times. The task with the higher priority is the one with the shorter period. In this case, \[ \tau_1 \] has a shorter period than \[ \tau_2 \]. Therefore, \[ \tau_1 \] has a higher priority.

\[ \tau_1 \]

\[ \tau_2 \]
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 **schedulable**
Earliest Deadline First (EDF)

\[ \begin{align*}
\tau_1 & := (p_1 = 5, C_1 = 2) \\
\tau_2 & := (p_2 = 9, C_2 = 4)
\end{align*} \]

(assume deadline = period)

Which one has a higher priority?

\[ \begin{align*}
\tau_1 & \text{ Release at 2, runs till 5} \\
\tau_2 & \text{ Release at 4, runs till 9}
\end{align*} \]
Earliest Deadline First (EDF)

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

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)
Earliest Deadline First (EDF)

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

[Diagram showing release times for \( \tau_1 \) and \( \tau_2 \)]

41

Earliest Deadline First (EDF)

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

[Diagram showing release times for \( \tau_1 \) and \( \tau_2 \)]

42
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 worst-case response time is not greater than its deadline

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

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

where \( r_i^0 = \sum_{j=1}^{i} C_j \)

• Iterative method
• Tasks are ordered according to their priority; \( \tau_1 \) has the highest priority

• If \( r_i^{k+1} > D_i \) \quad \rightarrow \text{Unschedulable}

• If \( r_i^{k+1} = r_i^k \leq D_i \) \quad \text{for some } k \quad \rightarrow \text{Schedulable}

• Test task-by-task. If any task fails the exact test, the task set is unschedulable

---

Exact Test

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

Released at time 0

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

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

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

(assume deadline = period)

\[ r_0^3 = 4 + 4 + 10 = 18 \]
Exact Test

\[ r_{i}^{k+1} = C_{i} + \sum_{j=1}^{i-1} \left\lfloor \frac{r_{j}^{k}}{p_{j}} \right\rfloor C_{j}, \quad r_{i}^{0} = \sum_{j=1}^{i} C_{j} \]

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

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

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

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

\[ r_{3}^{1} = 10 + \sum_{j=1}^{2} \left\lfloor \frac{r_{j}^{0}}{p_{j}} \right\rfloor C_{j} = 10 + \left\lfloor \frac{18}{10} \right\rfloor 4 + \left\lfloor \frac{18}{15} \right\rfloor 4 = 10 + 2 \cdot 4 + 2 \cdot 4 = 10 + 8 + 8 = 26 \]
Exact Test

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

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

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

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

New job of Task 1 arrives before Task 3's job finishes. -> Additional preemption

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

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

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

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

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

\[ r_2^2 = 10 + \sum_{j=1}^{2} \left[ \frac{r_j^1}{p_j} \right] C_j = 10 + \left[ \frac{26}{10} \right] 1 + \left[ \frac{26}{15} \right] 4 = 10 + 3 \cdot 4 + 2 \cdot 4 = 10 + 12 + 8 = 30 \]

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

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

\[ \begin{align*}
\tau_1 &:= (p_1 = 10, c_1 = 4) \\
\tau_2 &:= (p_2 = 15, c_2 = 4) \\
\tau_3 &:= (p_3 = 35, c_3 = 10)
\end{align*} \]

Worst-case Response Time (=30) < Deadline (=35)

Utilization Bound Test

Task Utilization

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

Processor Utilization \((n=\text{number of tasks})\)

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

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>$C_i$ (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>
</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>

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>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>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>

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

$$U_1 + U_2 \approx 0.467 < U_{RM}(2) = 0.828$$
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)

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

**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 \([\text{Liu73}]\) 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>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>

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 \([\text{Liu73}]\) 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>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>

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

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

Priority Inversion

So far, tasks are assumed to be independent
What if tasks share data?
- Synchronization!

```c
semaphore->P();
// critical section goes here
semaphore->V();
```
- But it can be a source of priority inversion

A few definitions
- Synchronization
- Mutual exclusion
- Critical section
- Priority inversion
- Scheduler

How to use semaphores
- Binary semaphores can be used for mutual exclusion
- Scheduling semantics
- Controlling access to a finite resource
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.

High Priority

Low Priority

Time

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

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

Time

Normal Execution
Critical Section

P()
Unbounded Priority Inversion

High Priority

Medium Priority

Medium Priority

Low Priority

Normal Execution

Critical Section

Time

Unbounded Priority Inversion

High Priority

Medium Priority

Medium Priority

Low Priority

Normal Execution

Critical Section

Time
Unbounded Priority Inversion

High Priority

Medium Priority

Medium Priority

Low Priority

Time

Normal Execution
Critical Section

Unbounded Priority Inversion

High Priority

Medium Priority

Medium Priority

Low Priority

V()

V()
Unbounded Priority Inversion

It actually happened on Mars!

What really happened on Mars?

NASA Mars Pathfinder (1997)

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!

THE PROBLEM

The Mars Pathfinder mission was widely predicted as “風險” in the early days after its July 4, 1997 landing on the Martian surface. Scientists included in its equipment an extremely sensitive, high-tech, but fragile Mark Zbarrel sensor networked to its camera, deploying the Pathfinder rover, and gathering and transmitting astronomical data back to Earth, including the panoramic images that were sent to the Web. But within minutes, the rover controller detected a failure—a failure that eventually caused the entire mission to be lost.

It was just after a standard gathering meteorological data start-up procedure that was sent to the rover. The money shot—data that was sent to the Web—was lost forever, as the rover controller detected a failure—a failure that eventually caused the entire mission to be lost.

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.

1) is executing, accessing the bus.
2) can’t access the bus. It is blocked by .
3) preempts. is further blocked.
4) Watchdog timer notices has not executed for some time. Hence, it resets the system!
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

Task 1 (High Priority)

Task 2 (Medium Priority)

Task 3 (Low Priority)
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)

Time

Normal Execution
Critical Section

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


---

Interested in Research?

- Self-driving cars
  - CPSC 335
- Real-time + machine learning
- Internet of things
- Real-time scheduling

For research opportunities (e.g., senior project), feel free to contact

- Dr. Man-Ki Yoon (man-ki.yoon@yale.edu)
- Dr. Jung-Eun Kim (jung-eun.kim@yale.edu)
- Prof. Zhong Shao (zhong.shao@yale.edu)