CS 422/522 Design & Implementation of Operating Systems

Lecture 22: Real-Time Systems

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

Part of the slides are based on UIUC CS 431 Lecture Notes

1

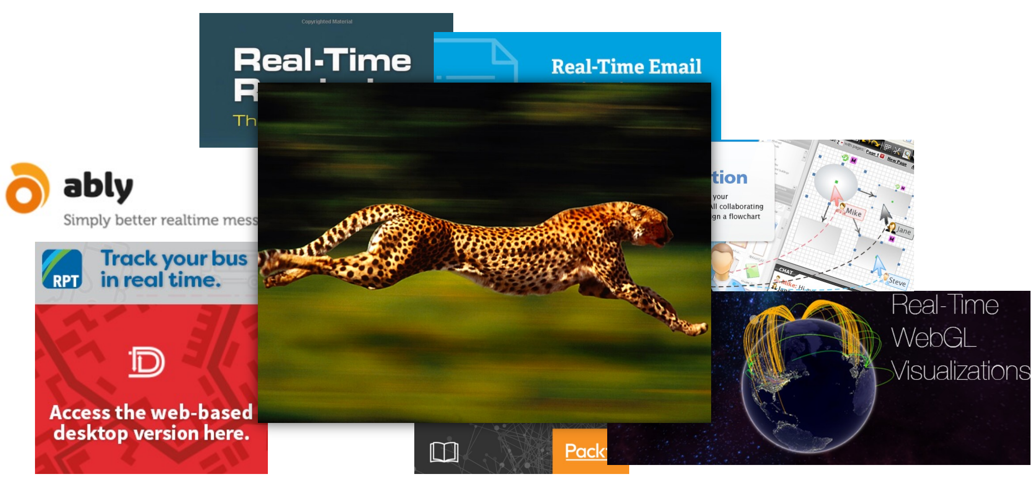
What is a Real-Time System?

2

What is a Real-Time System?

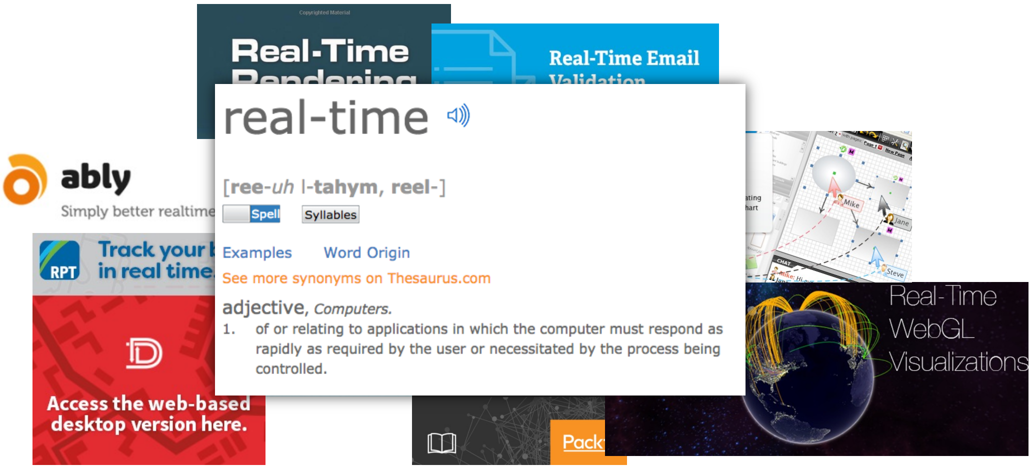
3

What is a Real-Time System?



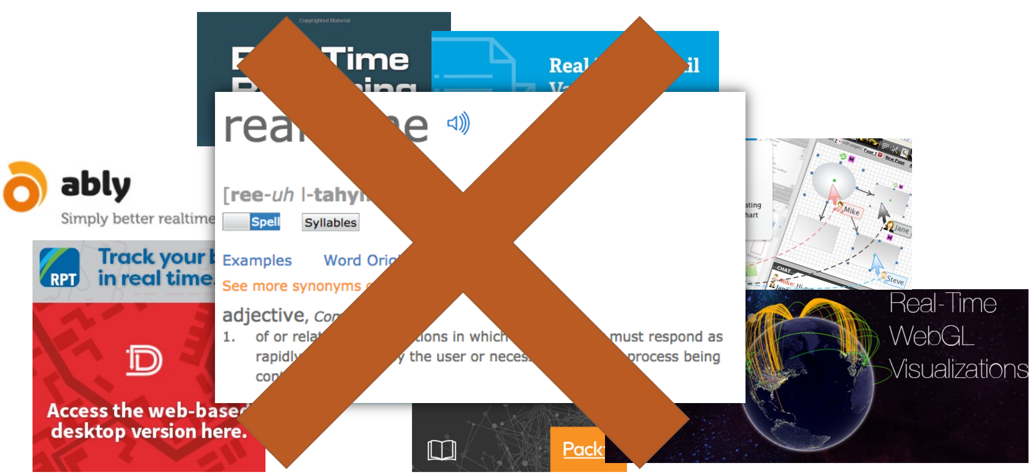
4

What is a Real-Time System?



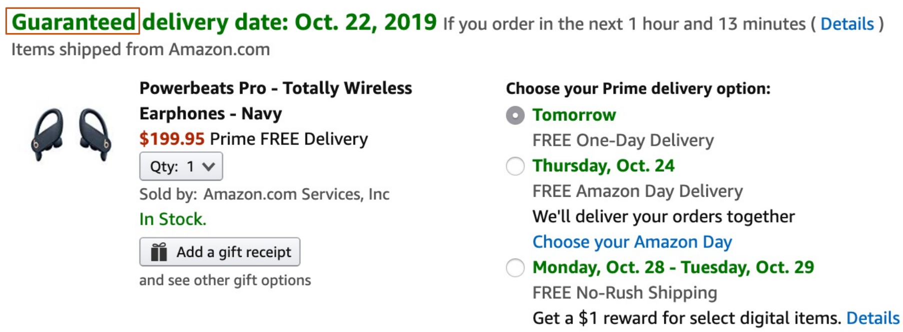
5

What is a Real-Time System?



6

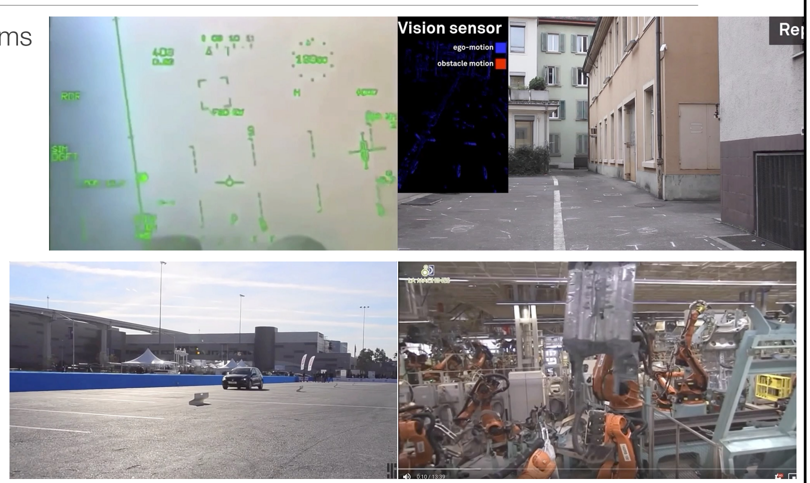
What is a Real-Time System?



7

Example of Real-Time Systems

- Avionics and automotive systems
- Radar systems
- Factory process control
- Robotics
- Multi-media systems
- …



8

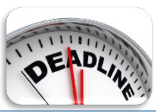
Real-Time Systems vs General-Purpose Systems

Real-Time Systems


General-Purpose Systems

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

Optimizing average performance

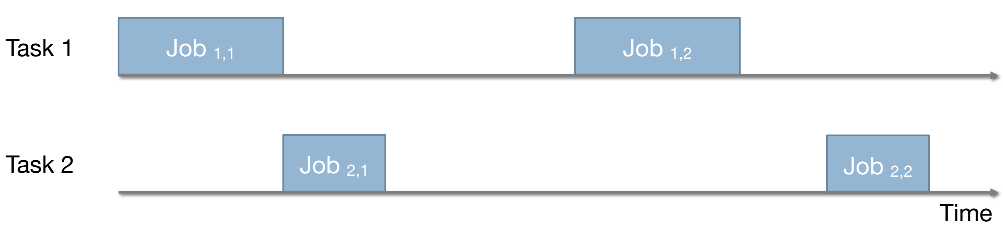


Correctness depends on
both functional and temporal aspects

9

Tasks and Jobs

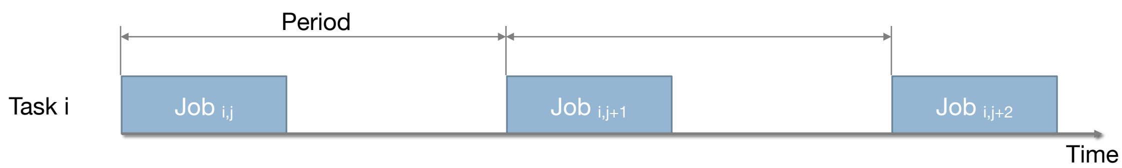
- **Task**: A sequence of the same type of jobs (e.g., process or thread)
- **Job**: A unit of computation, e.g.,
  - Reading sensor values
  - Computing control commands
- Sometimes task and job are used interchangeably



10

Periodic Task Model

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

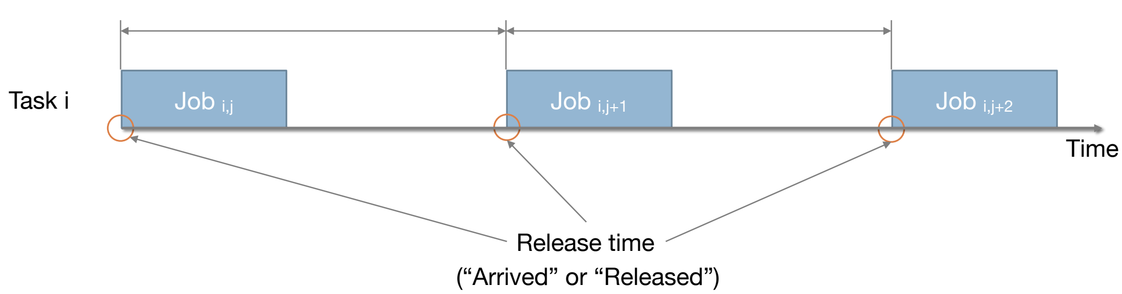


“Sporadic” task: inter-arrival time is not fixed, but still lower-bounded.

11

Periodic Task Model

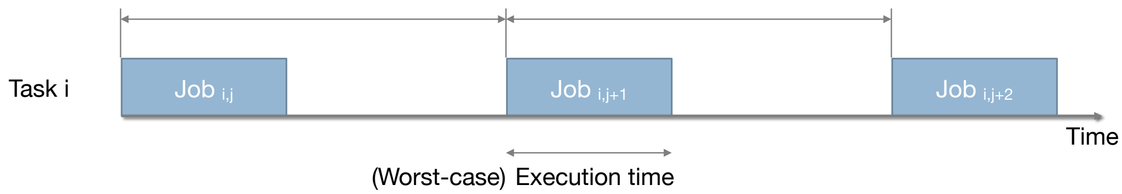
A task is said to be ***periodic*** if its inter-arrival time (i.e., period) is a constant



12

Periodic Task Model

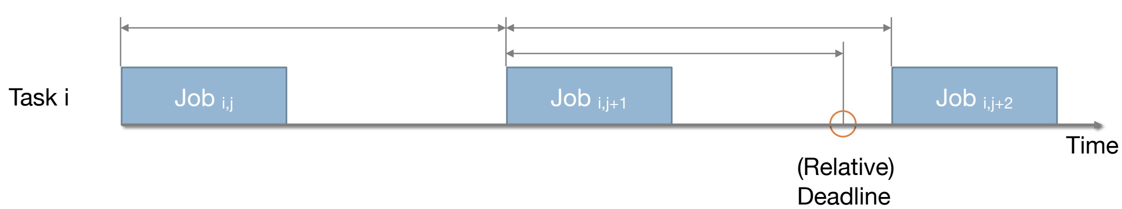
A task is said to be ***periodic*** if its inter-arrival time (i.e., period) is a constant



13

Periodic Task Model

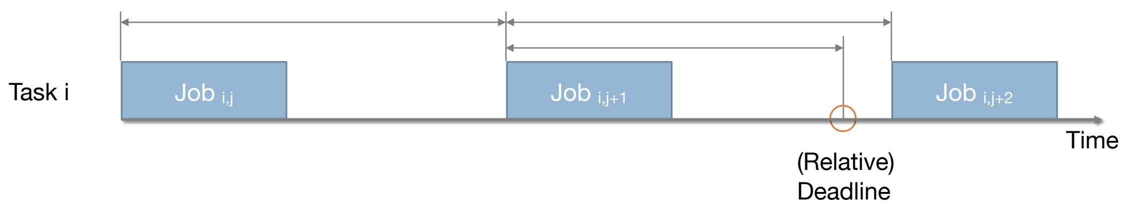
A task is said to be ***periodic*** if its inter-arrival time (i.e., period) is a constant



14

Periodic Task Model

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

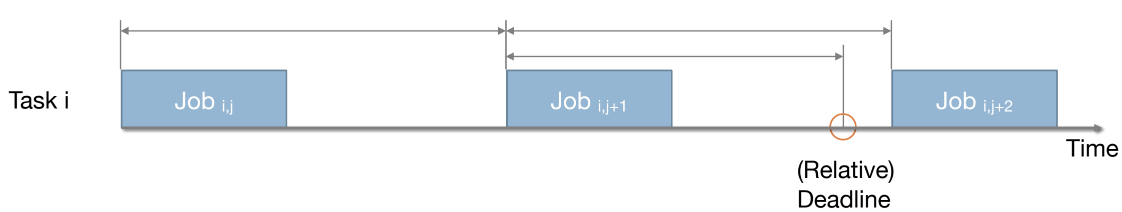


Hard deadline vs **Soft** deadline

15

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

16

Periodic Task Model

Sec.	Function	CPU	Period	Deadline	Importance
	Navigation				
3.1	Aircraft flight data	8 ms.	55 ms.		critical
3.2	Steering	6	80		critical
	Radar Control				
3.3	Radar search or	2	80		critical
	Radar tracking	2	40		critical
	Initiate tracking	2		200	essential
	Targeting				
3.4	Designate target	1		40	critical
	Confirm designation	1		200	critical
3.5	Target tracking	4	40		critical
	Target sweetening	2		40	critical
	Weapon control				
3.6	Input for weapon selection	1		200	essential
	Weapon selection processing	2		400	essential
	AUTO/CCIP toggle	1		200	critical
3.7	Weapon trajectory	7	100		critical
	Reinitiate trajectory	6		400	essential
3.8	Weapon release	1	10	5[4]	critical
	Controls and Displays				
3.9	HUD display (assuming AUTO-delivery)	6	52		essential
3.10	MPD HUD display	6	52		essential
3.11	MPD tactical display	8	52		essential
	MPD button response	1		200	background
	Change display mode	1		200	background

Source: Generic Avionics Software Specification

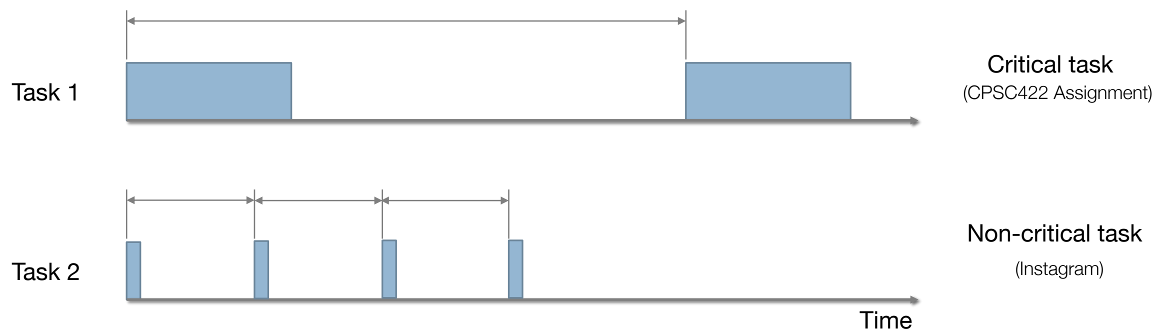
17

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?

18

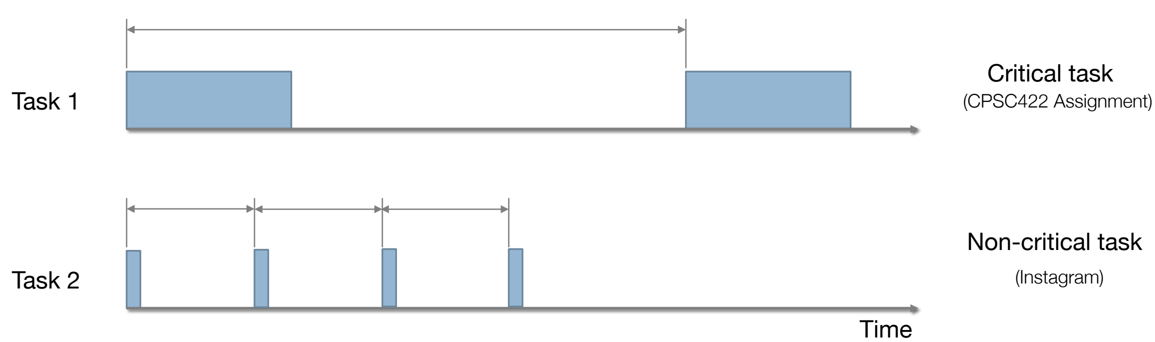
Priority and Criticality



Should we give a higher priority to Task 1? Or Task 2?

19

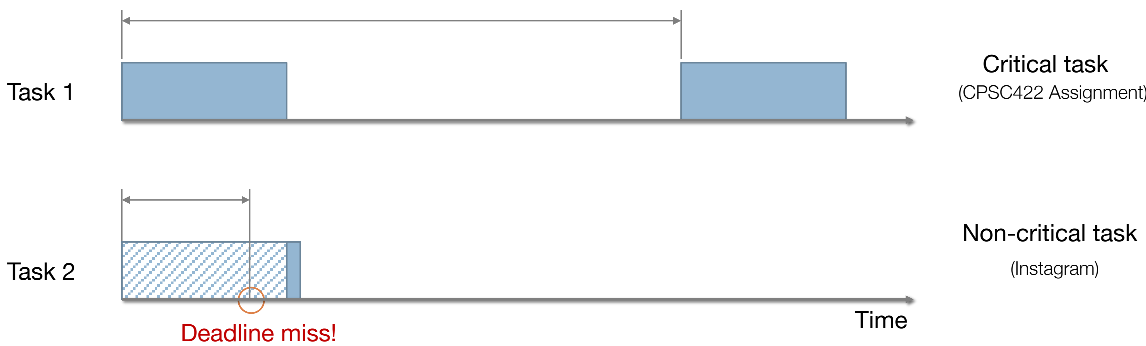
Priority and Criticality



Let’s say we give a higher priority to Task 1. What happens?

20

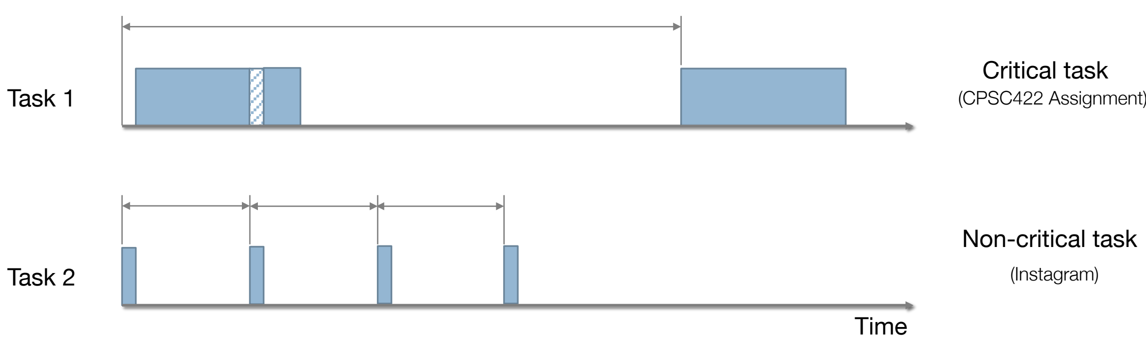
Priority and Criticality



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

21

Priority and Criticality



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

Both tasks are schedulable!

22

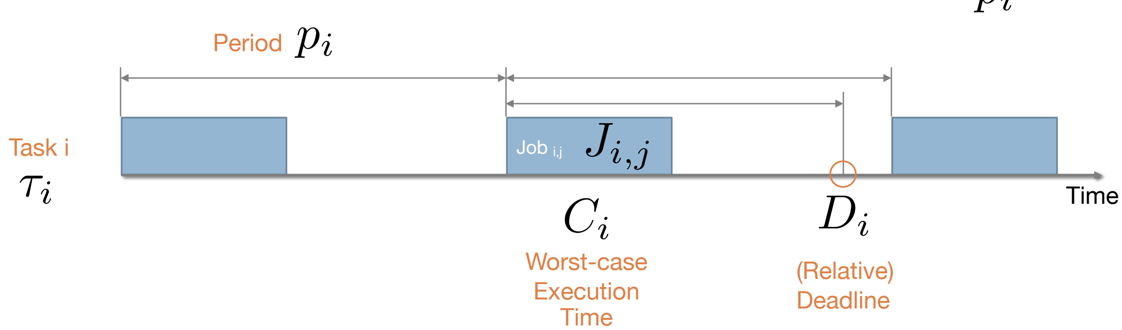
Priority and Criticality

- 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.

23

Notations

Task Utilization $U_i = \frac{C_i}{p_i}$



24

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

25

Real-Time Scheduling Algorithms

- **Rate-Monotonic (RM)**
  - Assign higher priority to ***tasks*** that hav
  - Optimal fixed-priority scheduling
- **Earliest Deadline First (EDF)**
  - Assign higher priority to ***jobs*** that have
  - Optimal dynamic-priority scheduling

What does it mean by '**optimal**' scheduling?

26

Real-Time Scheduling Algorithms

- **Rate-Monotonic (RM)**
  - Assign higher priority to ***tasks*** that hav
  - Optimal fixed-priority scheduling
- **Earliest Deadline First (EDF)**
  - Assign higher priority to ***jobs*** that have
  - Optimal dynamic-priority scheduling

What does it mean by '**optimal**' scheduling?

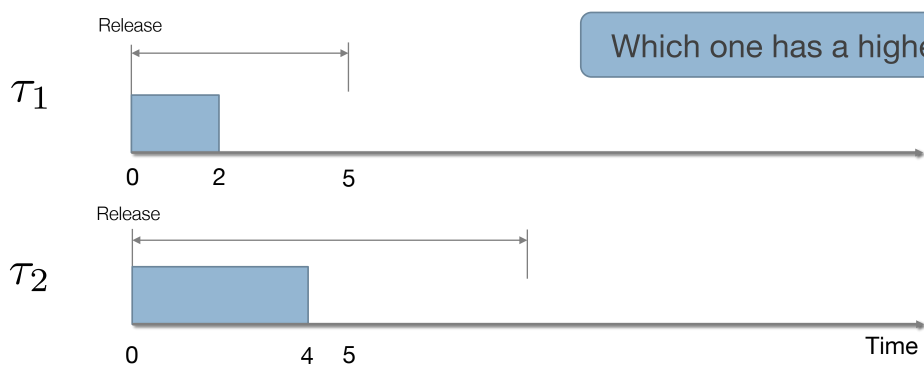
If a task set is not schedulable by the optimal scheduling algorithm, no other scheduling algorithms can schedule the task set.

27

Rate-Monotonic (RM)

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

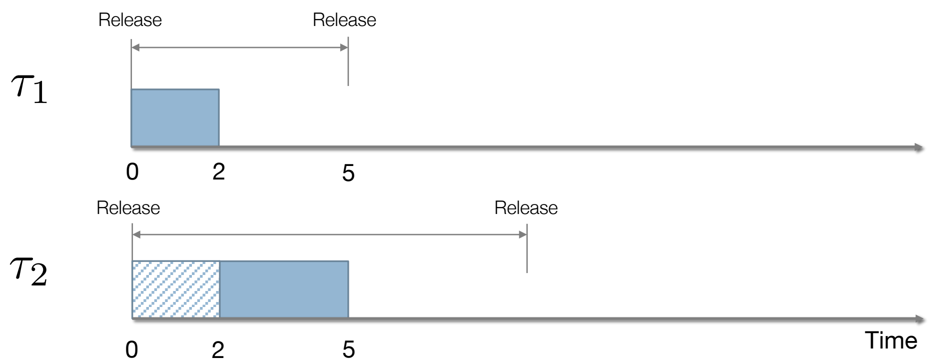
Which one has a higher priority?



28

Rate-Monotonic (RM)

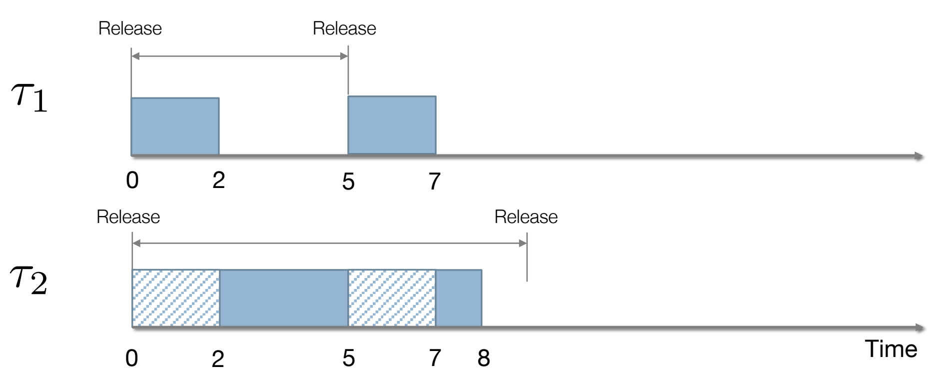
$$\tau_1 := (p_1 = 5, C_1 = 2)$$
$$\tau_2 := (p_2 = 9, C_2 = 4)$$



29

Rate-Monotonic (RM)

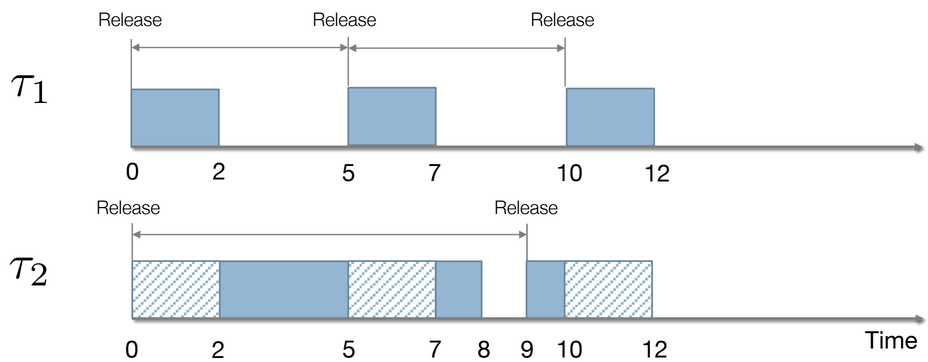
$$\tau_1 := (p_1 = 5, C_1 = 2)$$
$$\tau_2 := (p_2 = 9, C_2 = 4)$$



30

Rate-Monotonic (RM)

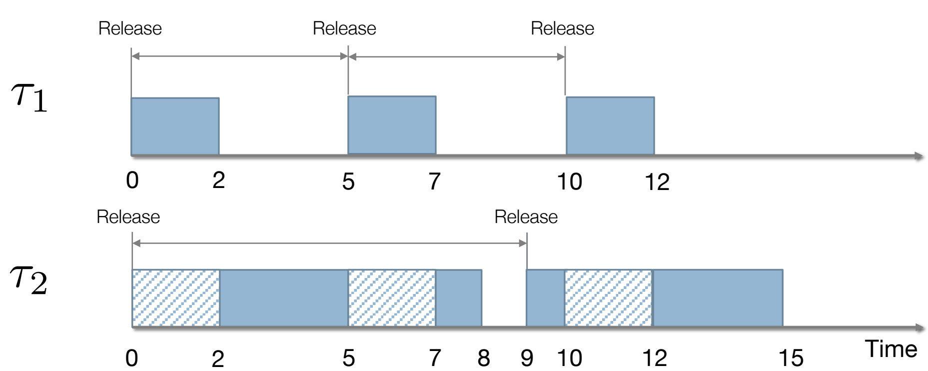
$\tau_1 := (p_1 = 5, C_1 = 2)$
$\tau_2 := (p_2 = 9, C_2 = 4)$



31

Rate-Monotonic (RM)

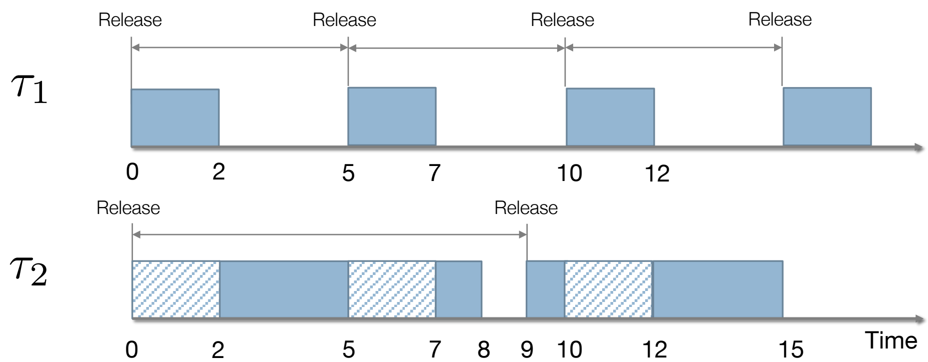
$\tau_1 := (p_1 = 5, C_1 = 2)$
$\tau_2 := (p_2 = 9, C_2 = 4)$



32

Rate-Monotonic (RM)

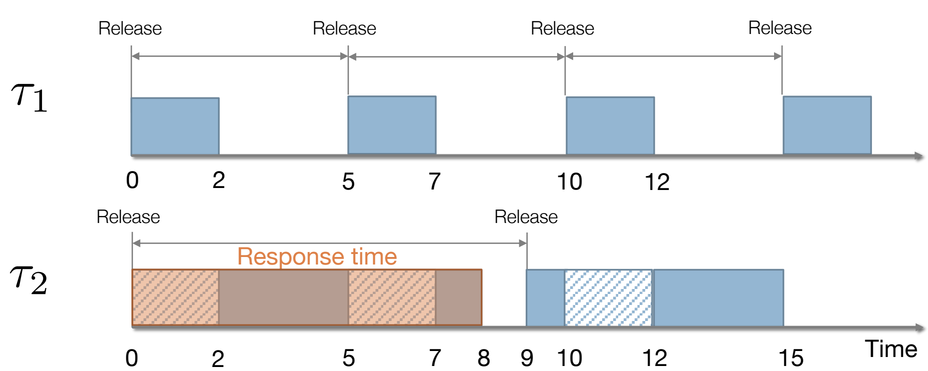
$$\tau_1 := (p_1 = 5, C_1 = 2)$$
$$\tau_2 := (p_2 = 9, C_2 = 4)$$



33

Rate-Monotonic (RM)

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



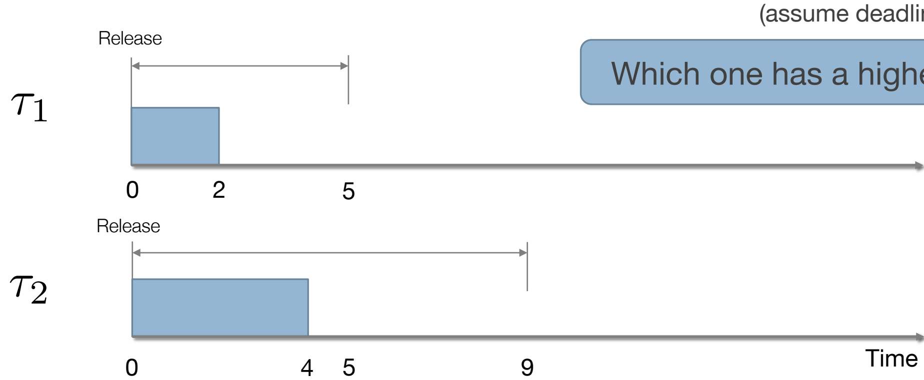
If response time <= deadline, the job is ***schedulable***

34

Earliest Deadline First (EDF)

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

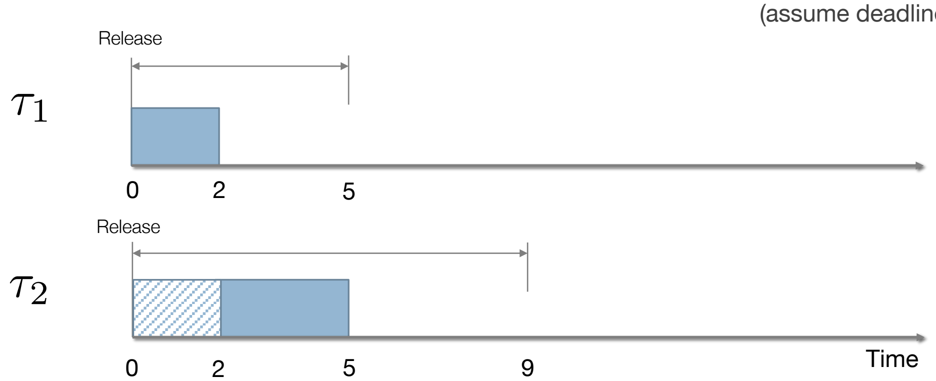
Which one has a higher priority?



35

Earliest Deadline First (EDF)

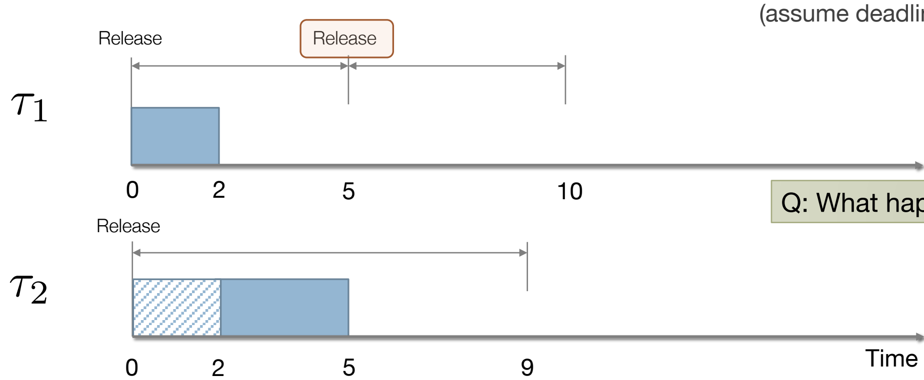
$\tau_1 := (p_1 = 5, C_1 = 2)$
$\tau_2 := (p_2 = 9, C_2 = 4)$
(assume deadline = period)



36

Earliest Deadline First (EDF)

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

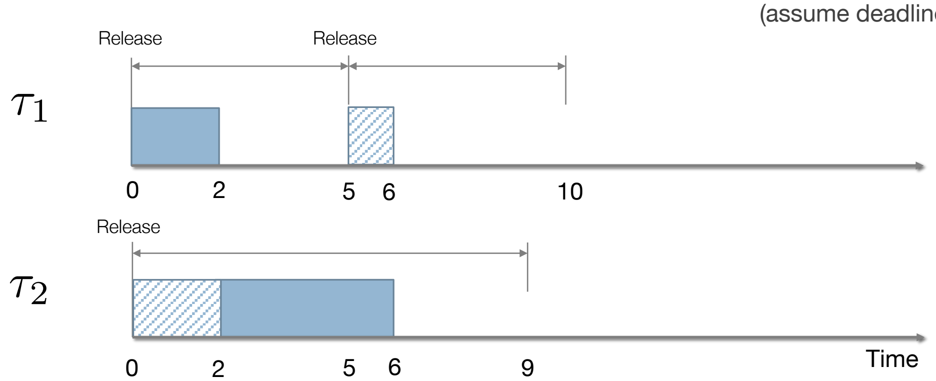


Q: What happens next?

37

Earliest Deadline First (EDF)

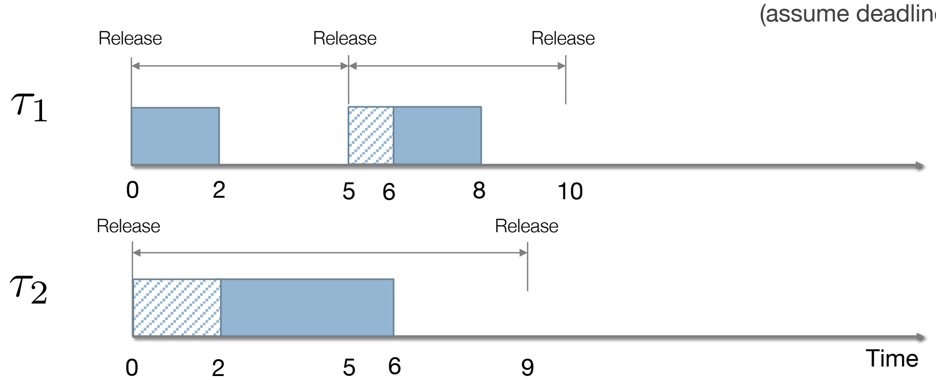
$\tau_1 := (p_1 = 5, C_1 = 2)$
$\tau_2 := (p_2 = 9, C_2 = 4)$
(assume deadline = period)



38

Earliest Deadline First (EDF)

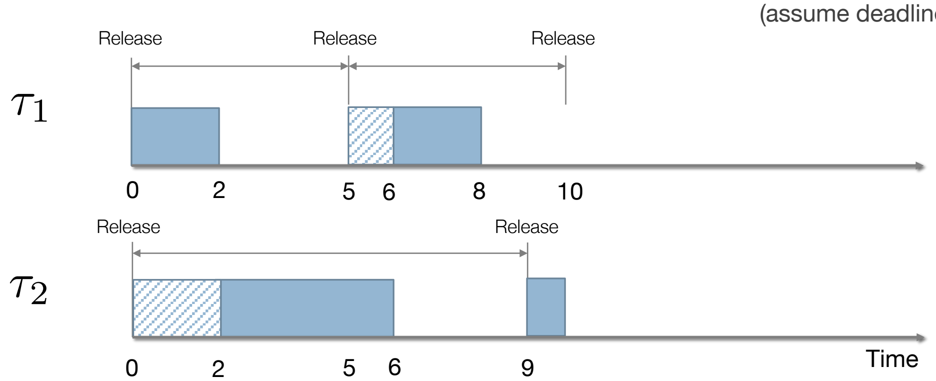
$\tau_1 := (p_1 = 5, C_1 = 2)$
$\tau_2 := (p_2 = 9, C_2 = 4)$
(assume deadline = period)



39

Earliest Deadline First (EDF)

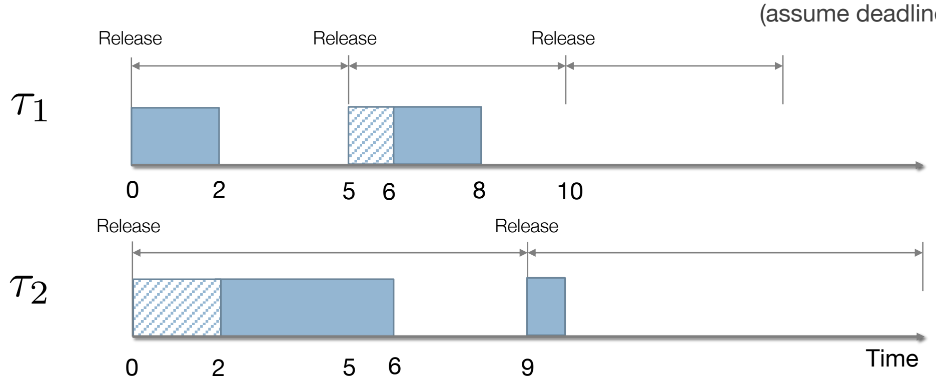
$\tau_1 := (p_1 = 5, C_1 = 2)$
$\tau_2 := (p_2 = 9, C_2 = 4)$
(assume deadline = period)



40

Earliest Deadline First (EDF)

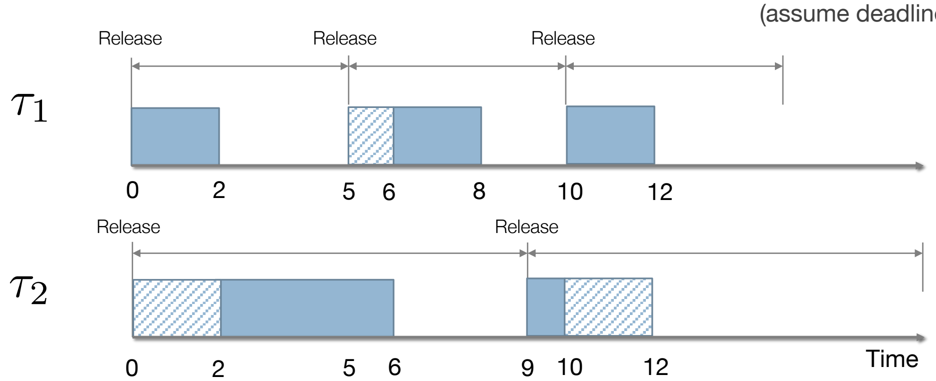
$\tau_1 := (p_1 = 5, C_1 = 2)$
$\tau_2 := (p_2 = 9, C_2 = 4)$
(assume deadline = period)



41

Earliest Deadline First (EDF)

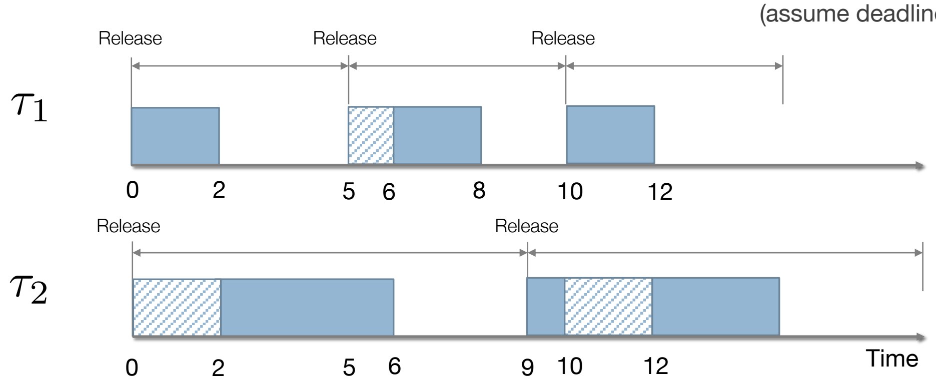
$\tau_1 := (p_1 = 5, C_1 = 2)$
$\tau_2 := (p_2 = 9, C_2 = 4)$
(assume deadline = period)



42

Earliest Deadline First (EDF)

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



43

Schedulability Analysis

- How can we know if a set of periodic tasks is schedulable?

44

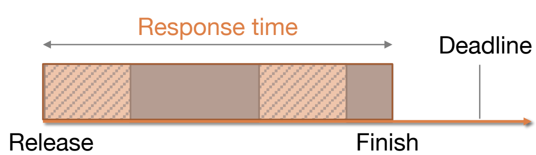
Schedulability Analysis

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

45

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

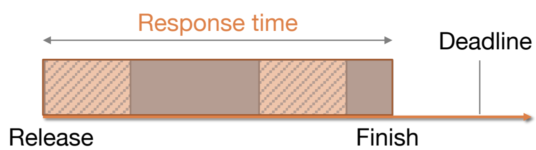


- When is the worst-case?

46

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])

[Liu73] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. Journal of the ACM, 20(1):46–61, 1973.

47

Exact Test

$$r_i^{k+1} = C_i + \sum_{j=1}^{i-1} \left\lceil \frac{r_i^k}{p_j} \right\rceil C_j \qquad \text{where } r_i^0 = \sum_{j=1}^{i} C_j$$

- Iterative method
- Tasks are ordered according to their priority; τ_1 has the highest priority
- If $r_i^{k+1} > D_i$ -> **Unschedulable**
- If $r_i^{k+1} = r_i^k \leq D_i$ for some k -> ***Schedulable***
- Test task-by-task. If any task fails the exact test, the task set is unschedulable

48

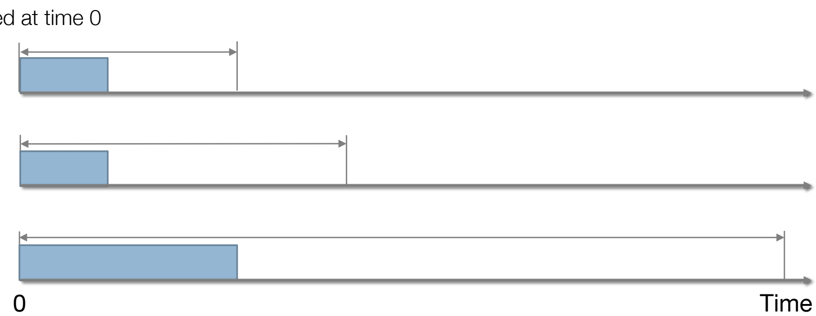
Exact Test

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

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

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

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



(assume deadline = period)

49

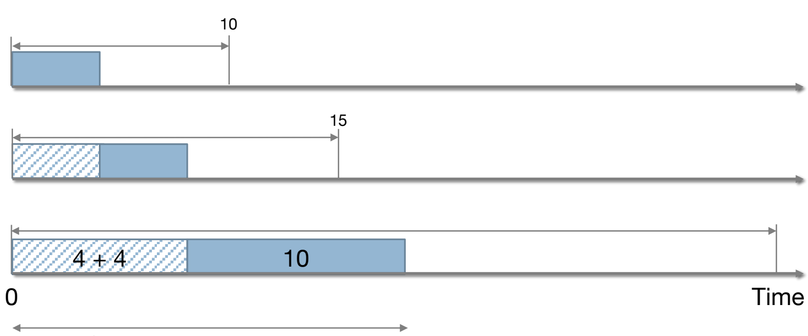
Exact Test

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

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

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

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



$r_3^0 = 4 + 4 + 10 = 18$

50

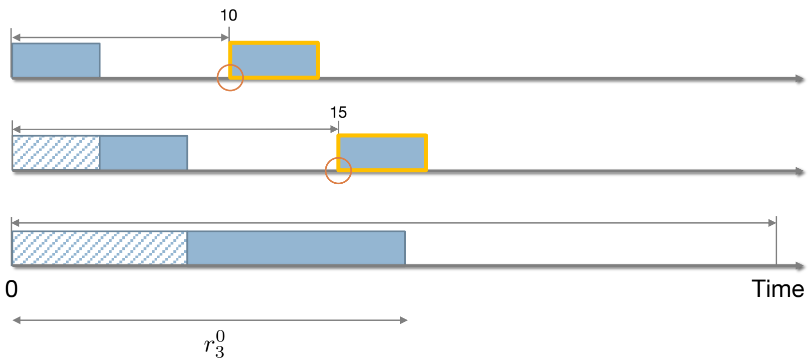
Exact Test

$$r_i^{k+1} = C_i + \sum_{j=1}^{i-1} \left\lceil \frac{r_i^k}{p_j} \right\rceil C_j \quad , \; r_i^0 = \sum_{j=1}^{i} C_j$$

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

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

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



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

51

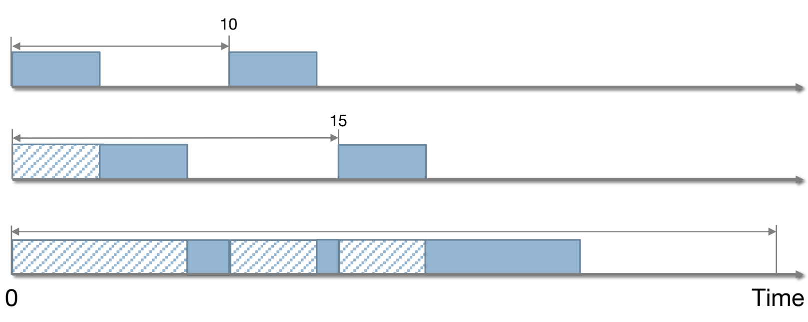
Exact Test

$$r_i^{k+1} = C_i + \sum_{j=1}^{i-1} \left\lceil \frac{r_i^k}{p_j} \right\rceil C_j \quad , \; r_i^0 = \sum_{j=1}^{i} C_j$$

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

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

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



$$r_3^1 = 10 + \sum_{j=1}^{2} \boxed{\left\lceil \frac{r_3^0}{p_j} \right\rceil} C_j = 10 + \left\lceil \frac{18}{10} \right\rceil 4 + \left\lceil \frac{18}{15} \right\rceil 4 = 10 + 2 \cdot 4 + 2 \cdot 4 = 10 + 8 + 8 = 26$$

52

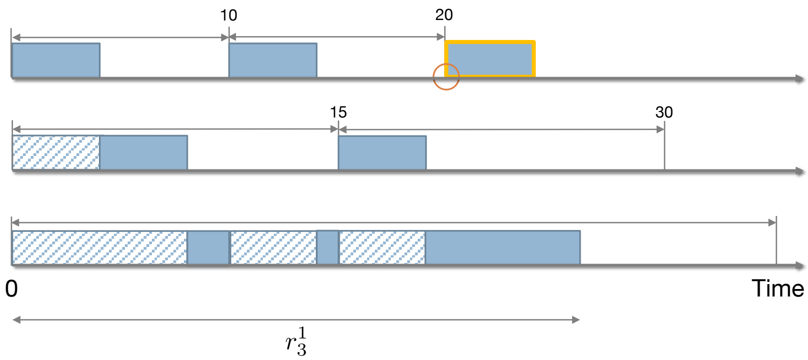
Exact Test

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

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

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

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



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

53

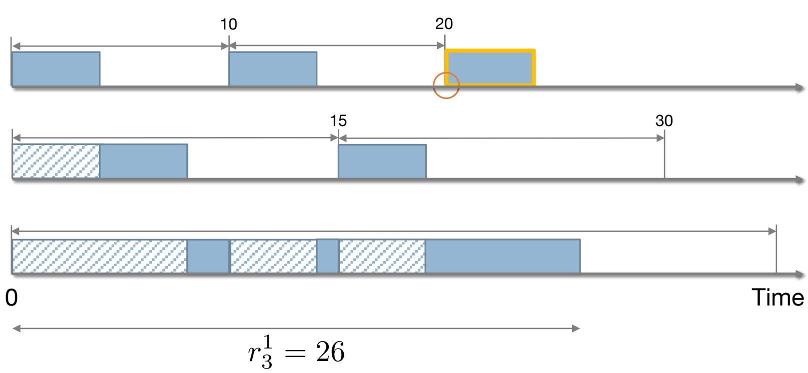
Exact Test

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

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

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

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



Q: Compute r_3^2

54

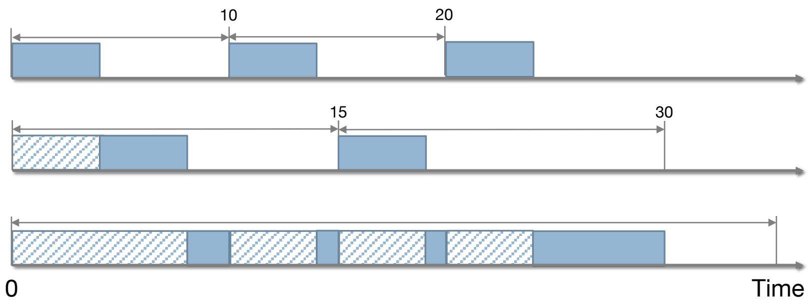
Exact Test

$$r_i^{k+1} = C_i + \sum_{j=1}^{i-1} \left\lceil \frac{r_i^k}{p_j} \right\rceil C_j \quad , \; r_i^0 = \sum_{j=1}^{i} C_j$$

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

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

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



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

55

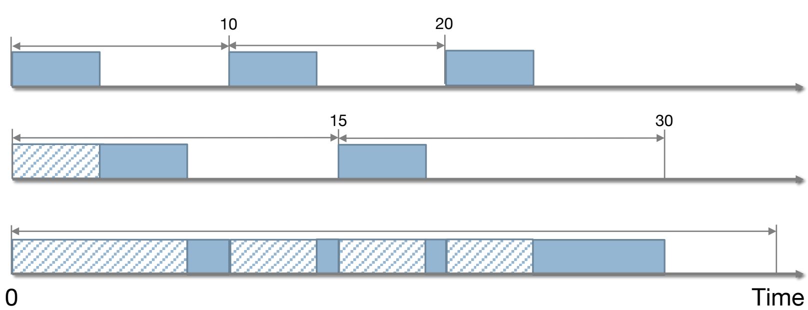
Exact Test

$$r_i^{k+1} = C_i + \sum_{j=1}^{i-1} \left\lceil \frac{r_i^k}{p_j} \right\rceil C_j \quad , \; r_i^0 = \sum_{j=1}^{i} C_j$$

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

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

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



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

$$r_3^3 = 10 + \sum_{j=1}^{2} \left\lceil \frac{r_3^2}{p_j} \right\rceil C_j = 10 + \left\lceil \frac{30}{10} \right\rceil 4 + \left\lceil \frac{30}{15} \right\rceil 4 = 10 + 3 \cdot 4 + 2 \cdot 4 = 10 + 12 + 8 = 30$$

56

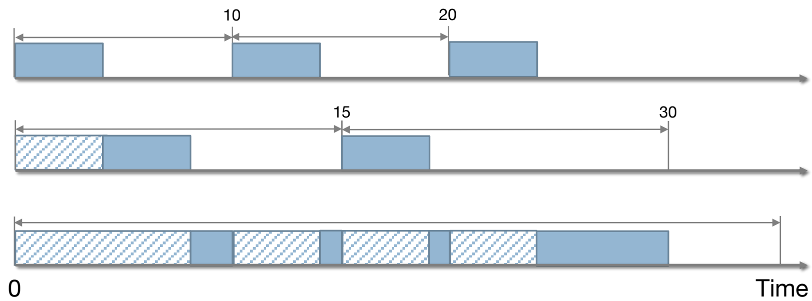
Exact Test

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

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

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

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



$$r_3^2 = 10 + \sum_{j=1}^{2} \left\lceil \frac{r_3^1}{p_j} \right\rceil C_j = 10 + \left\lceil \frac{26}{10} \right\rceil 4 + \left\lceil \frac{26}{15} \right\rceil 4 = 10 + 3 \cdot 4 + 2 \cdot 4 = 10 + 12 + 8 = \boxed{30}$$

$$r_3^3 = 10 + \sum_{j=1}^{2} \left\lceil \frac{r_3^2}{p_j} \right\rceil C_j = 10 + \left\lceil \frac{30}{10} \right\rceil 4 + \left\lceil \frac{30}{15} \right\rceil 4 = 10 + 3 \cdot 4 + 2 \cdot 4 = 10 + 12 + 8 = \boxed{30}$$

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

57

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 $U \leq U_b$

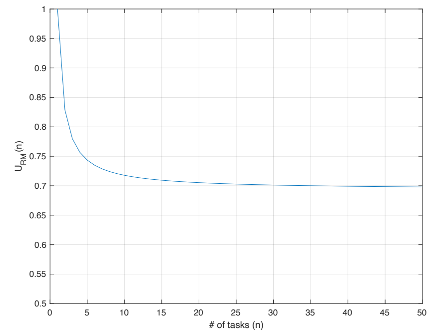
U_b depends on the scheduling algorithm, # of tasks, availability on timing information, …

58

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

	C_i (Execution Time)	p_i (Period)	U_i (Utilization)
Task 1	20	100	?
Task 2	40	150	?
Task 3	100	350	?

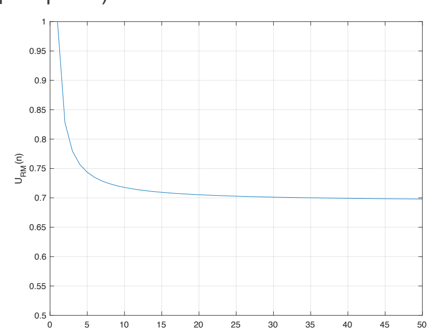
[Liu73] C. L. Liu and J. W. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. Journal of the ACM, 20(1):46–61, 1973.

59

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

	C_i (Execution Time)	p_i (Period)	U_i (Utilization)
Task 1	20	100	0.200
Task 2	40	150	0.267
Task 3	100	350	0.286

1) Check the schedulability of {task 1}:

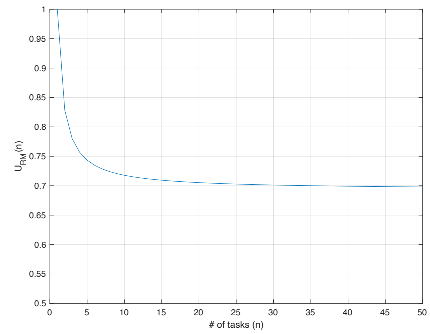
$$U_1 = 0.2 < U_{RM}(1) = 1$$

60

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

	C_i (Execution Time)	p_i (Period)	U_i (Utilization)
Task 1	20	100	0.200
Task 2	40	150	0.267
Task 3	100	350	0.286

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

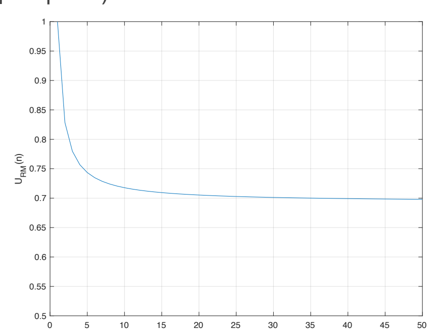
$$U_1 + U_2 \approx \qquad U_{RM}(\ ) =$$

61

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

	C_i (Execution Time)	p_i (Period)	U_i (Utilization)
Task 1	20	100	0.200
Task 2	40	150	0.267
Task 3	100	350	0.286

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

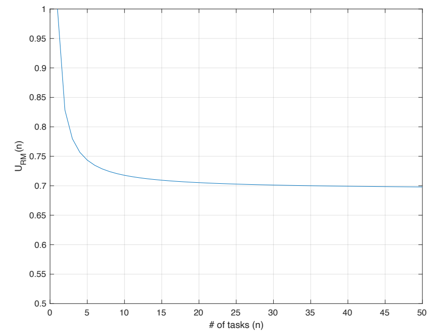
$$U_1 + U_2 \approx 0.467 < U_{RM}(2) = 0.828$$

62

RM Utilization Bound

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

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



Example

	C_i (Execution Time)	p_i (Period)	U_i (Utilization)
Task 1	20	100	0.200
Task 2	40	150	0.267
Task 3	100	350	0.286

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

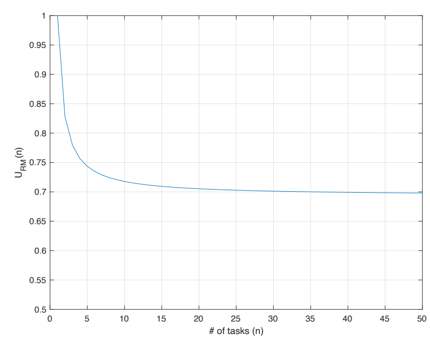
$$U_1 + U_2 + U_3 \approx 0.753 < U_{RM}(3) = 0.780$$

63

RM Utilization Bound

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

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



Example

	C_i (Execution Time)	p_i (Period)	U_i (Utilization)
Task 1	20	100	0.200
Task 2	40	150	0.267
Task 3	100	350	0.286

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

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

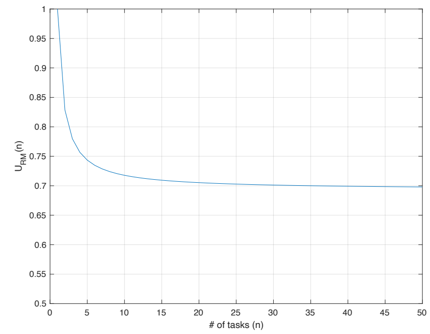
Q: What if C_1=40?

64

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

	Ci (Execution Time)	pi (Period)	Ui (Utilization)
Task 1	40	100	0.400
Task 2	40	150	0.267
Task 3	100	350	0.286

Q: What if C_1=40?

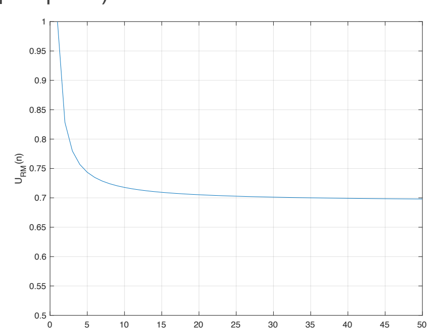
$$U_1 + U_2 + U_3 \approx 0.953 > U_{RM}(3) = 0.780$$

65

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

	Ci (Execution Time)	pi (Period)	Ui (Utilization)
Task 1	40	100	0.400
Task 2	40	150	0.267
Task 3	100	350	0.286

Q: What if C_1=40?

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

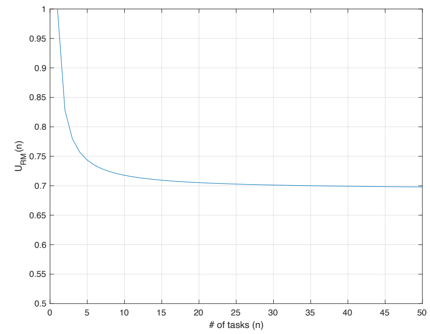
Q: Are the tasks unschedulable?

66

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

	C_i (Execution Time)	p_i (Period)	U_i (Utilization)
Task 1	40	100	0.400
Task 2	40	150	0.267
Task 3	100	350	0.286

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?

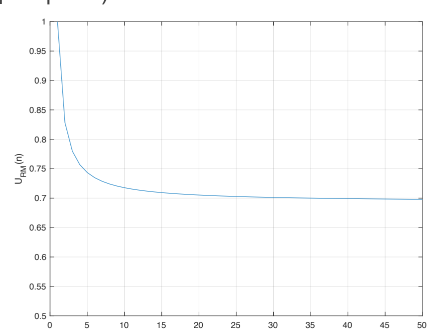
A: Not necessarily. Need to do the exact test!

67

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

	C_i (Execution Time)	p_i (Period)	U_i (Utilization)
Task 1	40	100	0.400
Task 2	40	150	0.267
Task 3	100	350	0.286

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

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

68

RM Utilization Bound

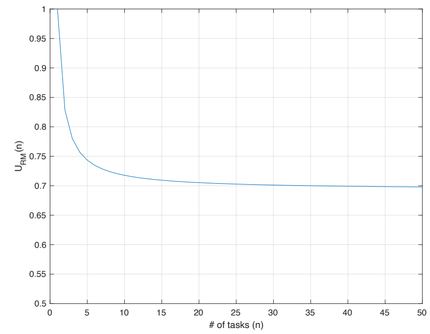
Utilization bound test is a ***sufficient*** condition

- 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 $\mathbf{n}$

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

Q: What does this mean?



69

RM Utilization Bound

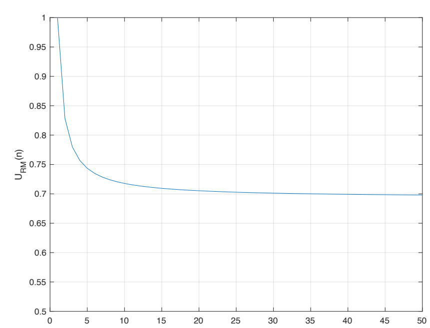
Utilization bound test is a ***sufficient*** condition

- 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 $\mathbf{n}$

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

- That is, any task set is schedulable if $U \leq U_{RM}$



70

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

	C_i (Execution Time)	p_i (Period)	U_i (Utilization)
Task 1	40	100	0.400
Task 2	40	150	0.267
Task 3	100	350	0.286

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

71

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?

72

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***

73

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***

A few definitions

- **Sychronization:**
  - using atomic operations to ensure cooperation between threads
- **Mutual exclusion:**
  - ensuring that only one thread does a particular thing at a time. One thread doing it excludes the other, and vice versa.
- **Critical section:**
  - piece of code that only one thread can execute at once. Only one thread at a time will get into the section of code.
- **Lock:** prevents someone from doing something
  - lock before entering critical section, before accessing shared data
  - unlock when leaving, after done accessing shared data
  - wait if locked

How to use semaphores

- Binary semaphores can be used for mutual exclusion:
  initial value of 1; P() is called before the critical section; and V() is called after the critical section.

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

- Scheduling constraints
  - having one thread to wait for something to happen
    - Example: Thread::Join, which must wait for a thread to terminate. By setting the initial value to 0 instead of 1, we can implement waiting on a semaphore
- Controlling access to a finite resource

Looks familiar? Lectures 6—9

74

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

Priority Inversion

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

Normal Execution

Critical Section

Assume these two tasks share a critical section

75

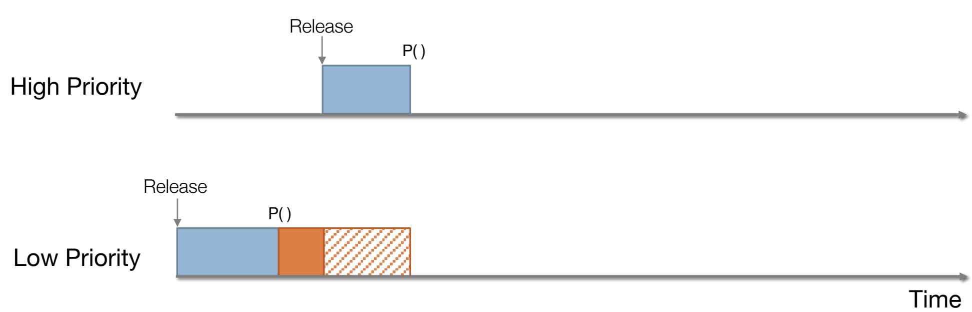
```
semaphore->P();
// critical section goes here
semaphore->V();
```

Priority Inversion

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

Normal Execution

Critical Section



76

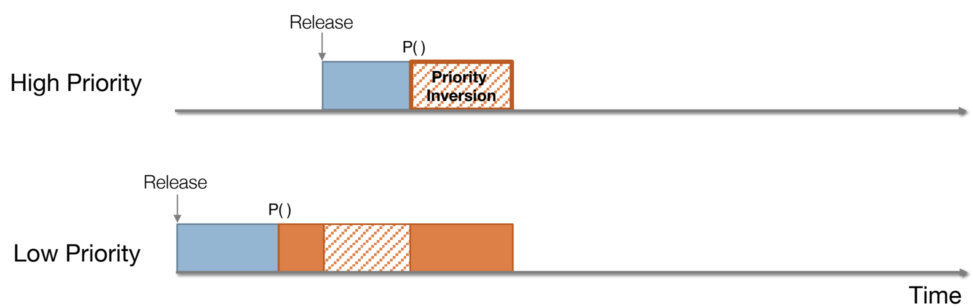
```
semaphore->P();
// critical section goes here
semaphore->V();
```

Priority Inversion

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

Normal Execution

Critical Section



77

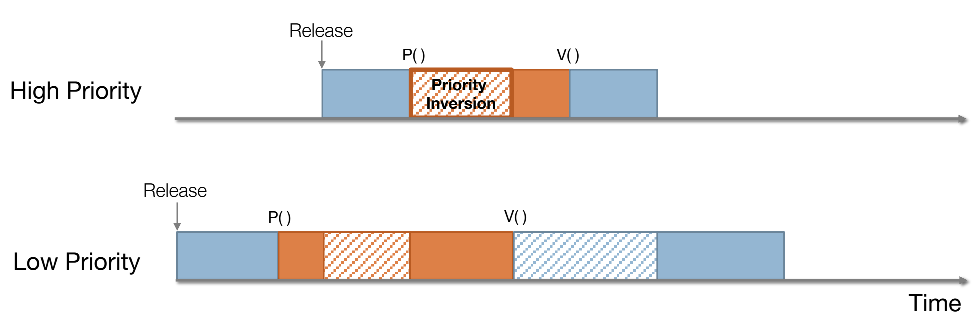
```
semaphore->P();
// critical section goes here
semaphore->V();
```

Priority Inversion

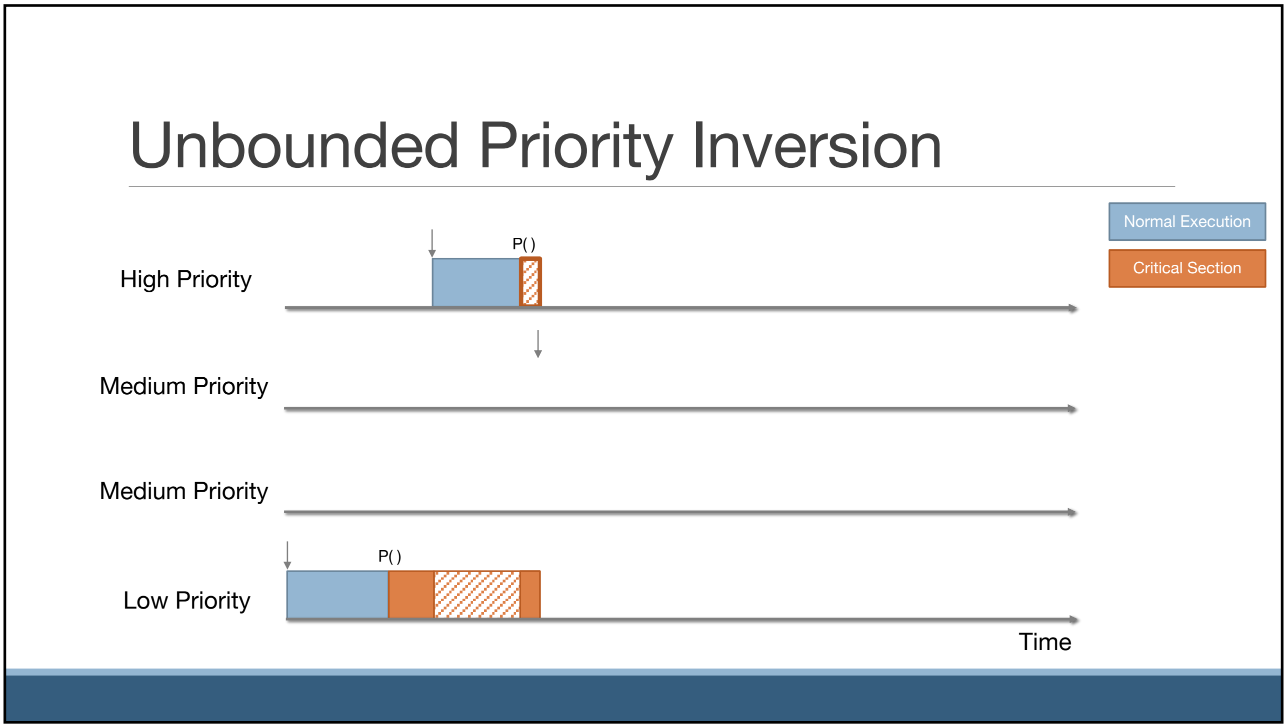
When a high priority task is delayed by a low priority task

Normal Execution

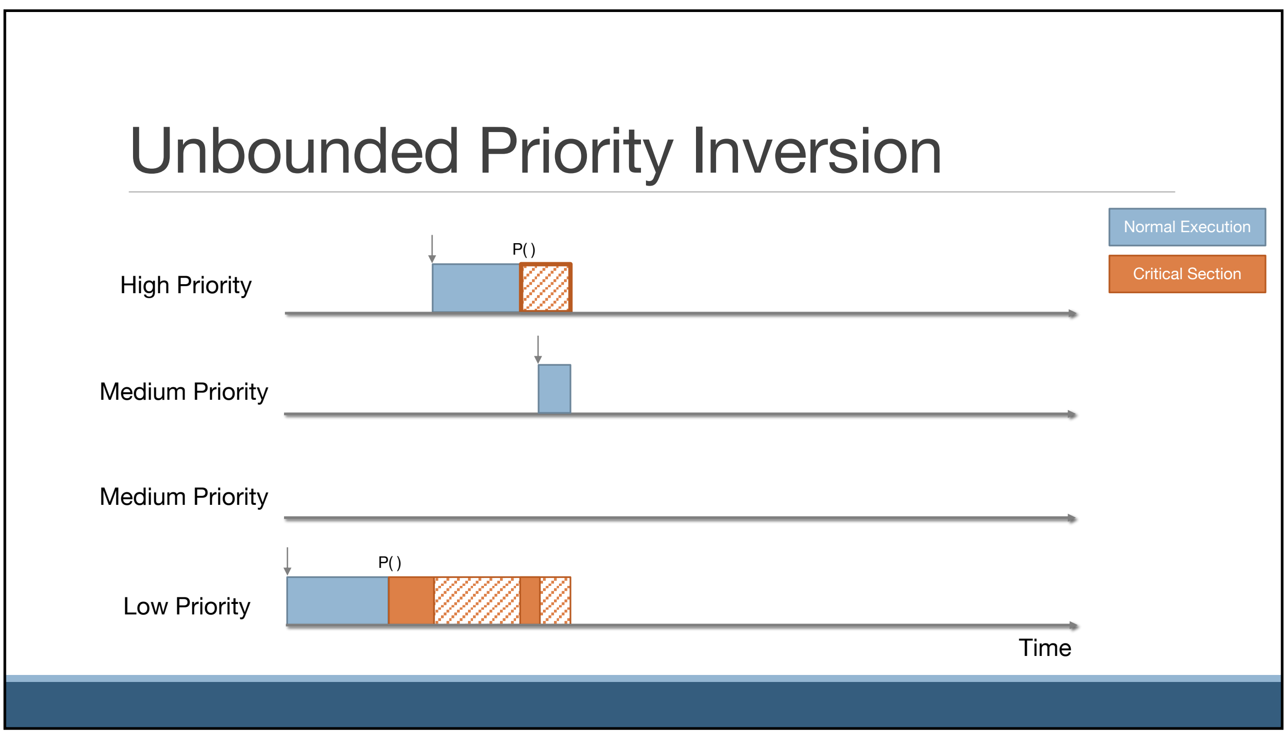
Critical Section



78

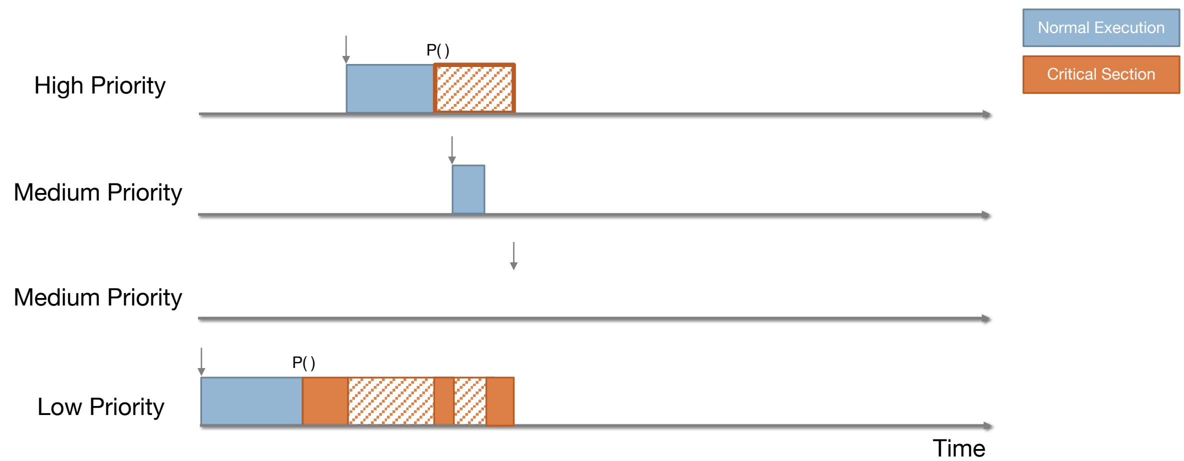

Unbounded Priority Inversion
Normal Execution
Critical Section
High Priority
P( )
Medium Priority
Medium Priority
Low Priority
P( )
Time

79


Unbounded Priority Inversion
Normal Execution
Critical Section
High Priority
P( )
Medium Priority
Medium Priority
Low Priority
P( )
Time

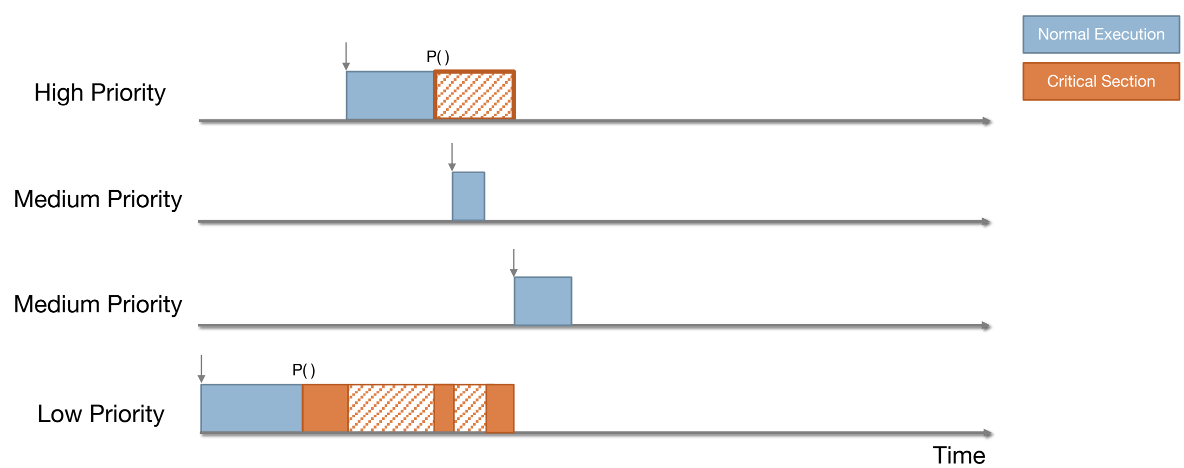
80

Unbounded Priority Inversion



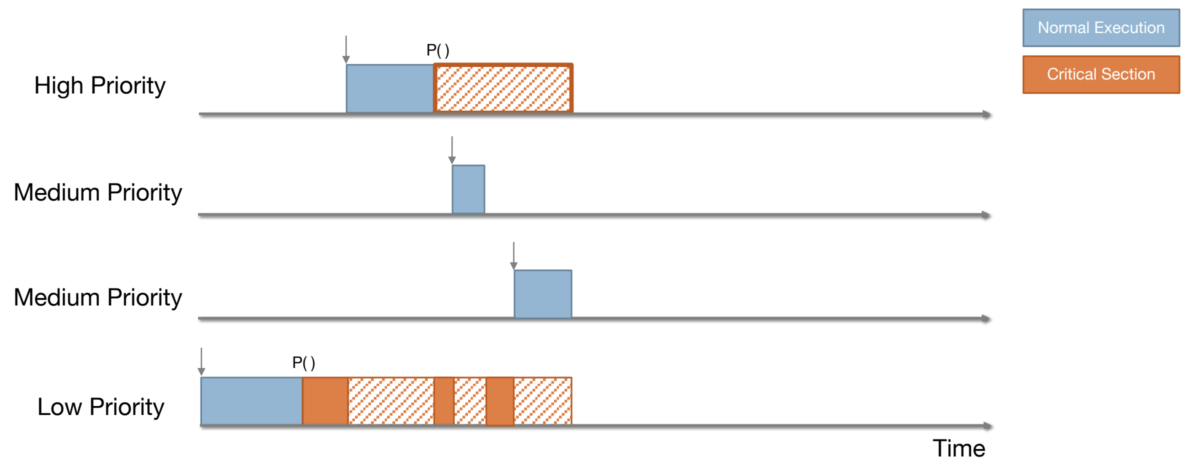
81

Unbounded Priority Inversion



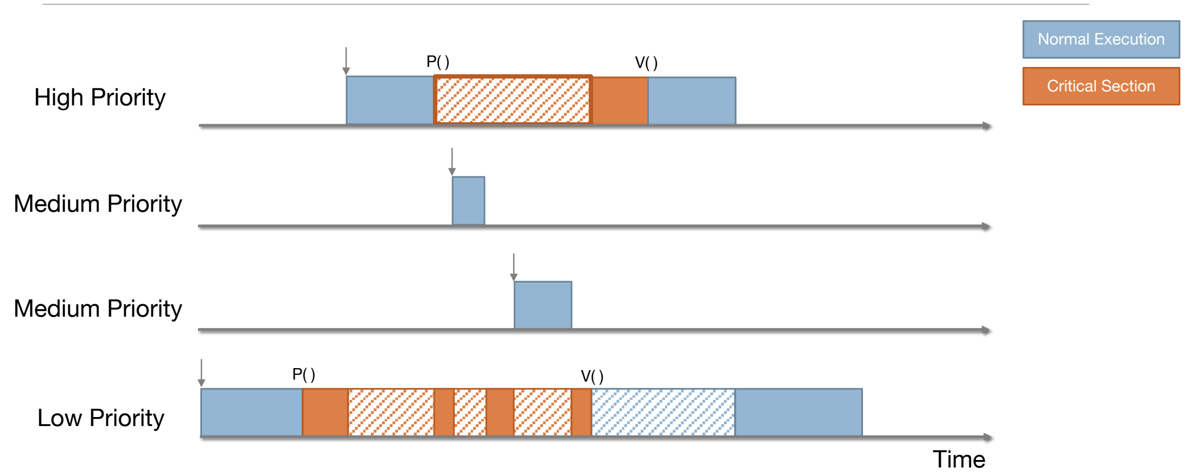
82

Unbounded Priority Inversion



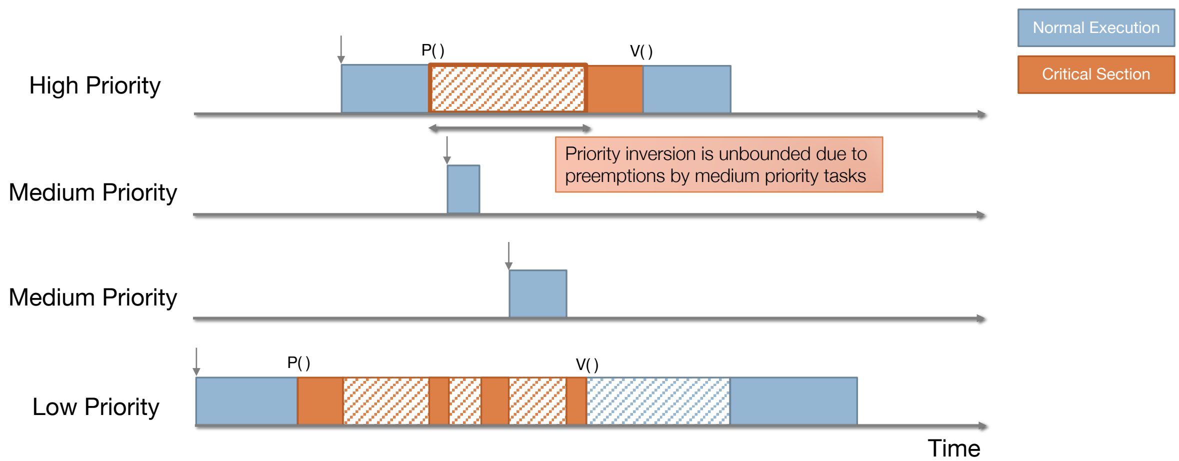
83

Unbounded Priority Inversion



84

Unbounded Priority Inversion



85

Unbounded Priority Inversion

It actually happened on Mars!

NASA Mars Pathfinder (1997)

What really happened on Mars?

From: Mike Jones <mbj@microsoft.com>
Sent: Sunday, December 07, 1997 6:47 PM
Subject: What really happened on Mars?

THE PROBLEM

The Mars Pathfinder mission was widely proclaimed as "flawless" in the early days after its July 4th, 1997 landing on the Martian surface. Successes included its unconventional "landing" -- bouncing onto the Martian surface surrounded by airbags, deploying the Sojourner rover, and gathering and transmitting 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 in terms such as "software glitches" and "the computer was trying to do too many things at once".

This we[illegible] the IEEE Real-Time Systems Symposium I heard a fascinating keynote address by David Wilner[illegible]ief Technical Officer of Wind River Systems. Wind River makes VxWorks, the real-time

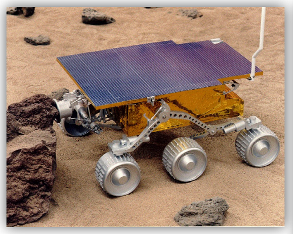
Click for more information

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

86

Unbounded Priority Inversion

It actually happened on Mars!



NASA Mars Pathfinder (1997)

What really happened on Mars?

From: Mike Jones <mbj@microsoft.com>
Sent: Sunday, December 07, 1997 6:47 PM
Subject: What really happened on Mars?

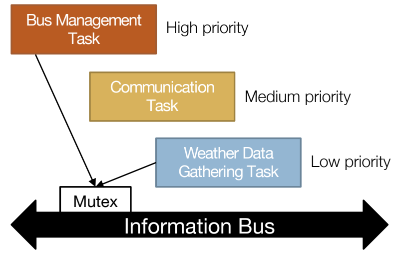
THE PROBLEM

The Mars Pathfinder mission was widely proclaimed as "flawless" in the early days after its July 4th, 1997 landing on the Martian surface. Successes included its unconventional "landing" -- bouncing onto the Martian surface surrounded by airbags, deploying the Sojourner rover, and gathering and transmitting 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 in terms such as "software glitches" and "the computer was trying to do too many things at once".

This we… the IEEE Real-Time Systems Symposium I heard a fascinating keynote address by David Wilner… Chief Technical Officer of Wind River Systems. Wind River makes VxWorks, the real-time embe… systems kernel that was used in the Mars Pathfinder mission. In his talk, he explained in detail

Click for more information

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



87

Unbounded Priority Inversion

It actually happened on Mars!

NASA Mars Pathfinder (1997)

What really happened on Mars?

From: Mike Jones <mbj@microsoft.com>
Sent: Sunday, December 07, 1997 6:47 PM
Subject: What really happened on Mars?

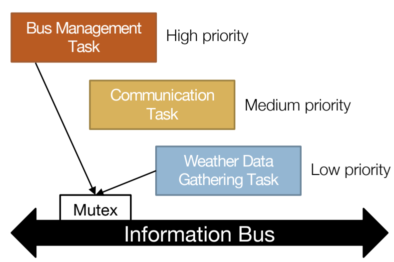
THE PROBLEM

The Mars Pathfinder mission was widely proclaimed as "flawless" in the early days after its July 4th, 1997 landing on the Martian surface. Successes included its unconventional "landing" -- bouncing onto the Martian surface surrounded by airbags, deploying the Sojourner rover, and gathering and transmitting 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 in terms such as "software glitches" and "the computer was trying to do too many things at once".

This we… the IEEE Real-Time Systems Symposium I heard a fascinating keynote address by David Wilner… Chief Technical Officer of Wind River Systems. Wind River makes VxWorks, the real-time embe… systems kernel that was used in the Mars Pathfinder mission. In his talk, he explained in detail

Click for more information

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) L is executing, accessing the bus.

2) H can't access the bus. It is blocked by L

3) M preempts. L , so H is further blocked.

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

88

Unbounded Priority Inversion

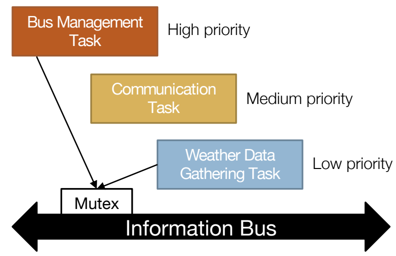
It actually happened on Mars!

How was the problem corrected?

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

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.

http://www.cs.cornell.edu/courses/cs614/1999sp/papers/pathfinder.html
https://www.cs.unc.edu/~anderson/teach/comp790/papers/mars_pathfinder_long_version.html

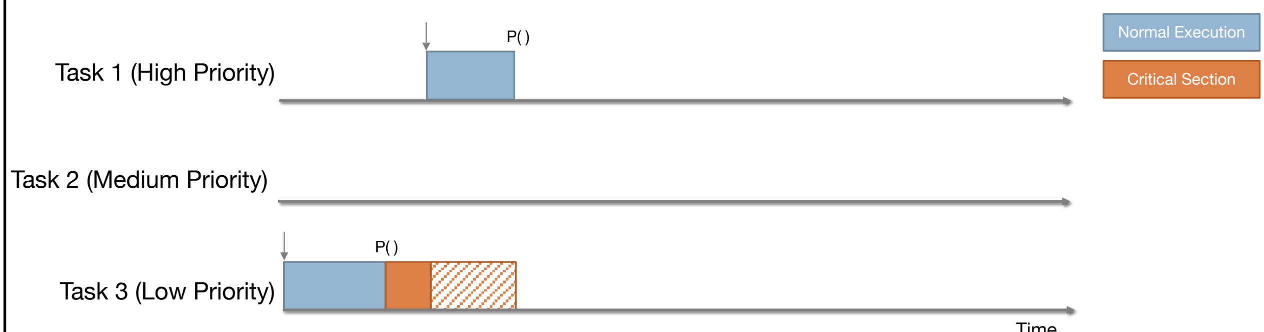


1) L is executing, accessing the bus.
2) H can't access the bus. It is blocked by L
3) M preempts. L , so H is further blocked.
4) Watchdog timer notices that H has not executed for some time. Hence, it resets the system!

89

Priority Inheritance Protocol

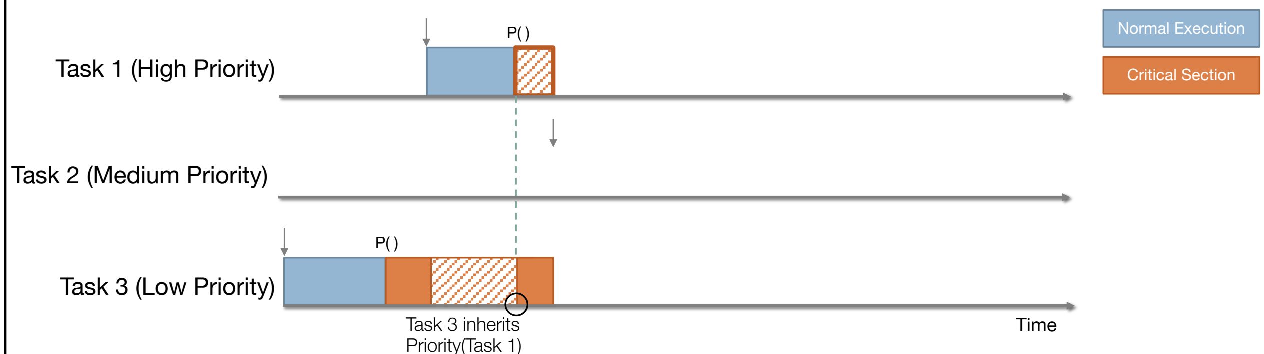
Low priority task inherits the highest priority of all the blocked tasks
- This keeps medium tasks from delaying the low priority task that is in a critical section



90

Priority Inheritance Protocol

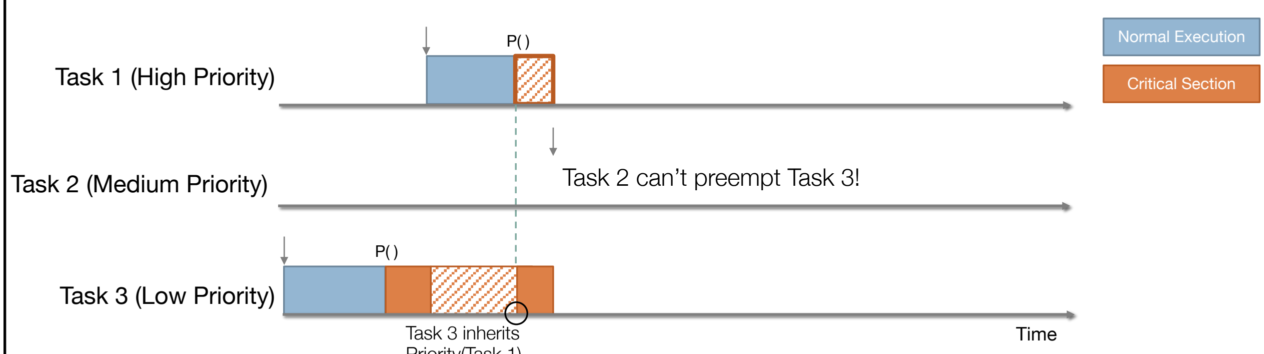
Low priority task inherits the highest priority of all the blocked tasks
- This keeps medium tasks from delaying the low priority task that is in a critical section



91

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

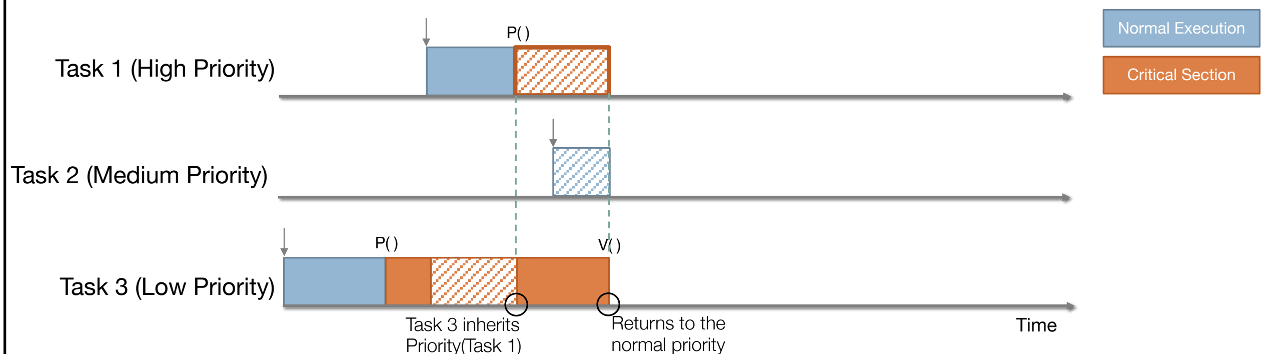


92

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

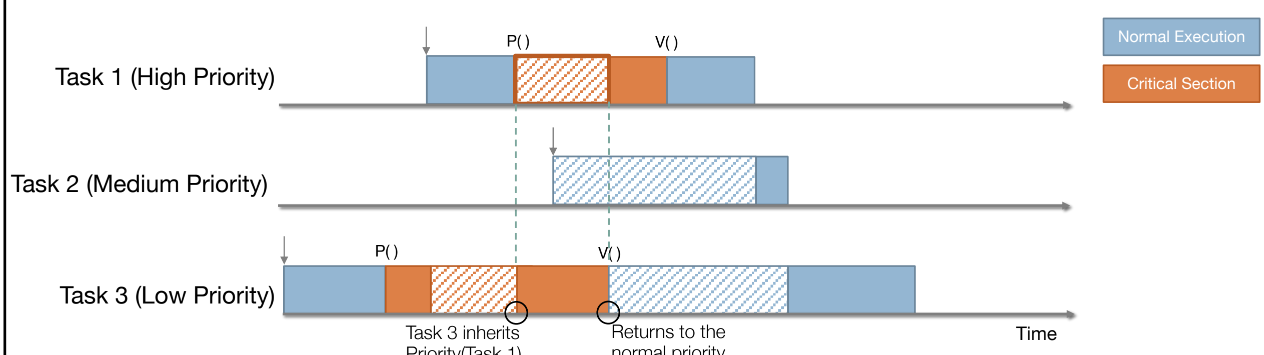


93

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



94

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)

95

Priority Inheritance Protocol

A job ***J***
- **n** = nu
- **m** = n

But cha
- Soluti

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

96

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

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.

97