Periodic task scheduling

Static priority scheduling
Rate monotonic priority assignment
Derivation of the RM utilization bound

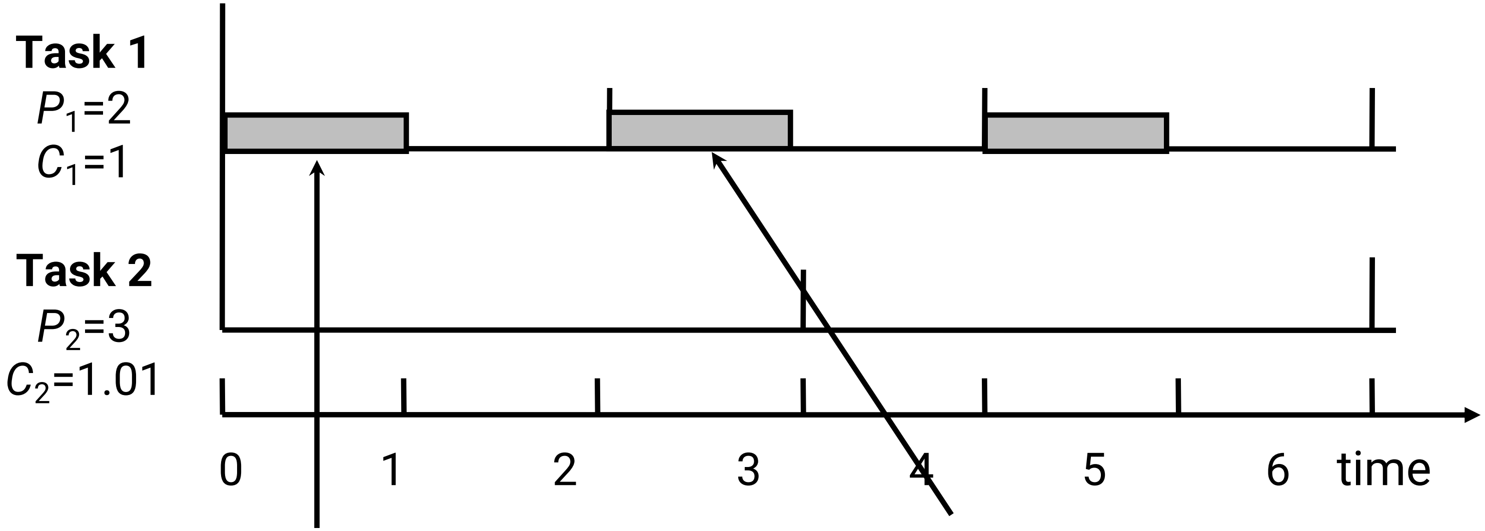
Impact of GRMS

- GRMS: Generalized Rate Monotonic Scheduling
- Cited in the Selected Accomplishment section of the National Research Council’s report on A Broader Agenda for Computer Science and Engineering in 1992.
- “Through the development of Rate Monotonic Scheduling [theory], we now have a system that will allow [Space Station] Freedom's computers to budget their time, to choose between a variety of tasks, and decide not only which one to do first but how much time to spend in the process.” [Deputy Administrator of NASA, Aaron Cohen]
- “The navigation payload software for the next block of Global Positioning System upgrade recently completed testing. ... This design would have been difficult or impossible prior to the development of rate monotonic theory.” [L. Doyle, and J. Elzey, “Successful Use of Rate Monotonic Theory on A Formidable Real-Time System”]

Review

- Terminology
  - Definitions of tasks, task invocations, release/arrival time, absolute deadline, relative deadline, period, start time, finish time, …
  - Preemptive versus non-preemptive scheduling
  - Priority-based scheduling
  - Static versus dynamic priorities
- Utilization (*U*) and schedulability
  - Main problem: Find *Bound* for scheduling policy such that
    - *U* < *Bound* → All deadlines met!
- Optimality of EDF scheduling
  - $Bound_{EDF}$ = 100%

A quick refresher



Task 1
P_1=2
C_1=1

Task 2
P_2=3
C_2=1.01

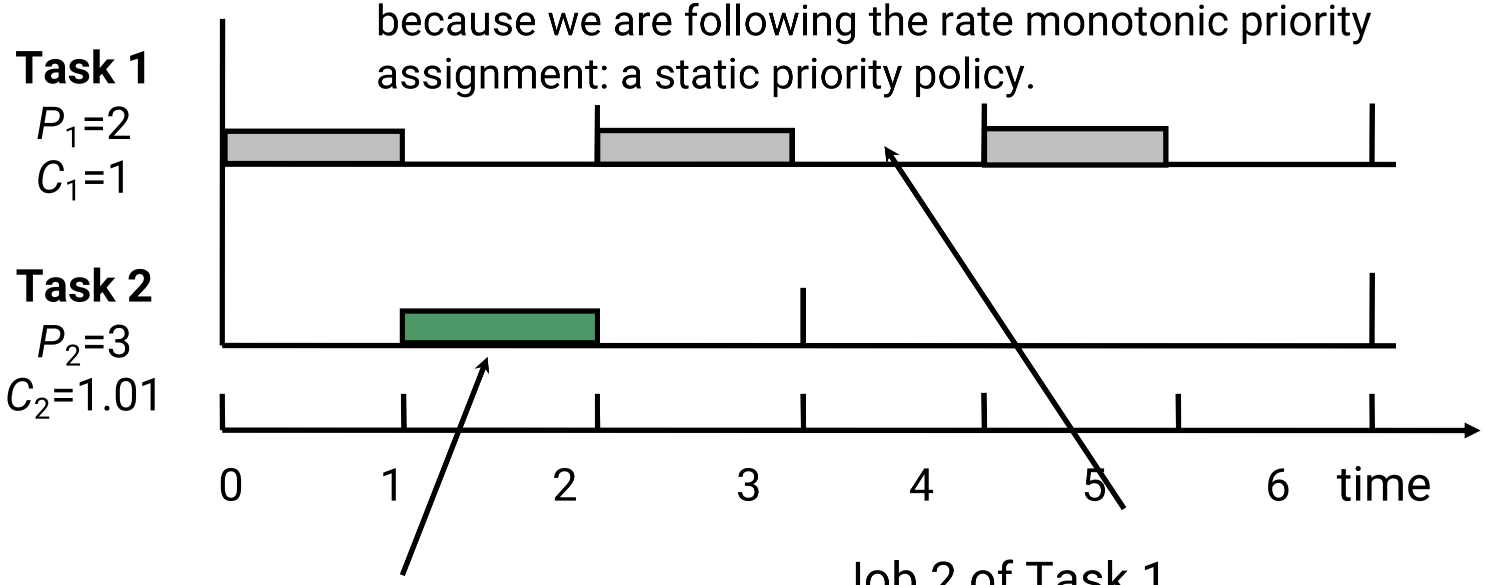
0 1 2 3 4 5 6 time

Job 1 of Task 1
Release time = 0
Relative deadline = 2
Absolute deadline = 0+2 = 2

Job 2 of Task 1
Release time = 2
Relative deadline = 2
Absolute deadline = 2+2 = 4

The release time of the first job of a task is also known as the **phase** of the task. The phase of Task 1 is 0.

A quick refresher



Job 2 of Task 1 preempts Job 1 of Task 2 at time=2 because we are following the rate monotonic priority assignment: a static priority policy.

Job 1 of Task 2
Release time = 0
Start time = 1
Relative deadline = 3
Absolute deadline = 0+3 = 3

Job 2 of Task 1
Release time = 2
Relative deadline = 2
Absolute deadline = 2+2 = 4

Schedulability analysis of periodic tasks

- Main problem
  - Given a set of periodic tasks, can they meet their deadlines?
  - Depends on scheduling policy
- Solution approaches
  - Utilization bounds (simplest)
  - Exact analysis (NP-Hard)
  - Approximation schemes with provable bounds on approximation error
  - (Meta)Heuristics
- Two most important scheduling policies
  - Earliest deadline first (dynamic)
  - Rate monotonic (static)

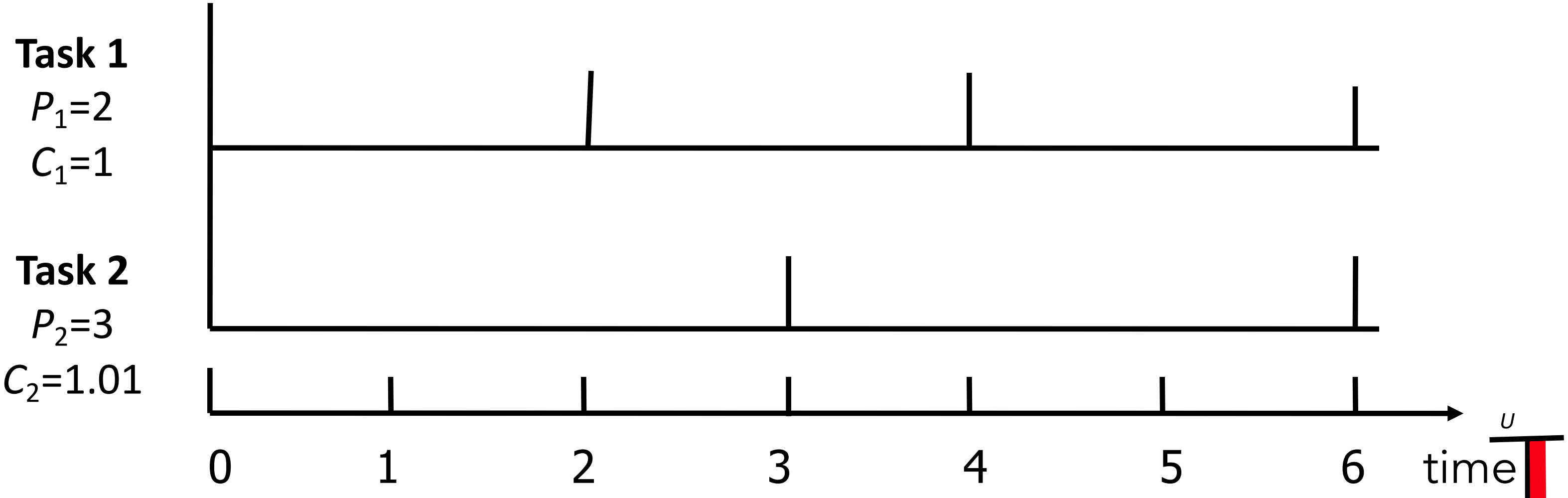
Schedulability analysis of periodic tasks

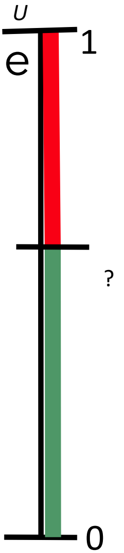
- Main problem
  - Given a set of periodic tasks, can they meet their deadlines?
  - Depends on scheduling policy
- Solution approaches
  - Utilization bounds (simplest)
  - Exact analysis (NP-Hard)
  - Approximation schemes with provable bounds on approximation error
  - (Meta)Heuristics
- Two most important scheduling policies
  - Earliest deadline first (Dynamic)
  - Rate monotonic (static)

Utilization bounds

- Intuitively,
  - The lower the processor utilization, *U*, the easier it is to meet deadlines.
  - The higher the processor utilization, *U*, the more difficult it is to meet deadlines.
- **Question:** Is there a threshold U_{bound} such that
  - When $U < U_{bound}$ deadlines are met
  - When $U > U_{bound}$ deadlines are missed

Example (Rate monotonic scheduling)

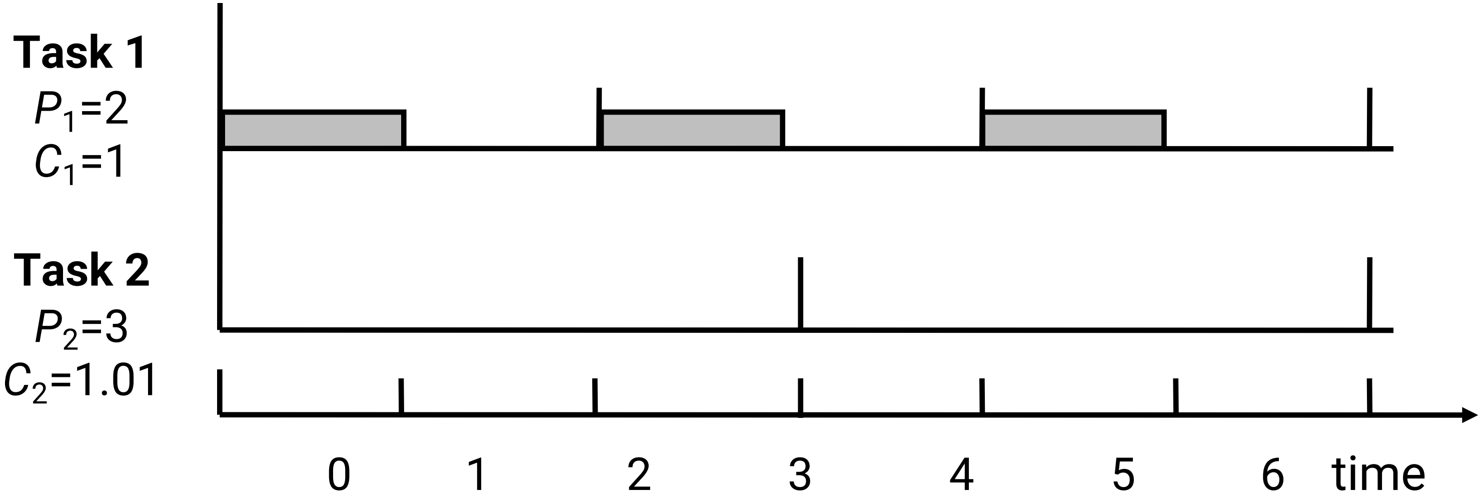




$$U = \frac{C_1}{P_1} + \frac{C_2}{P_2} = \frac{1}{2} + \frac{1.01}{3} \approx 0.833$$

- Question: Is there a threshold U_{bound} such that
  - When $U < U_{bound}$ deadlines are met
  - When $U > U_{bound}$ deadlines are missed

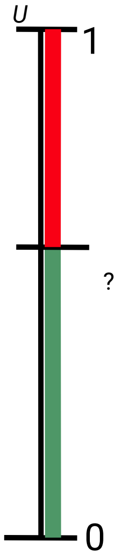
Example (Rate monotonic scheduling)



Task 1
P_1=2
C_1=1

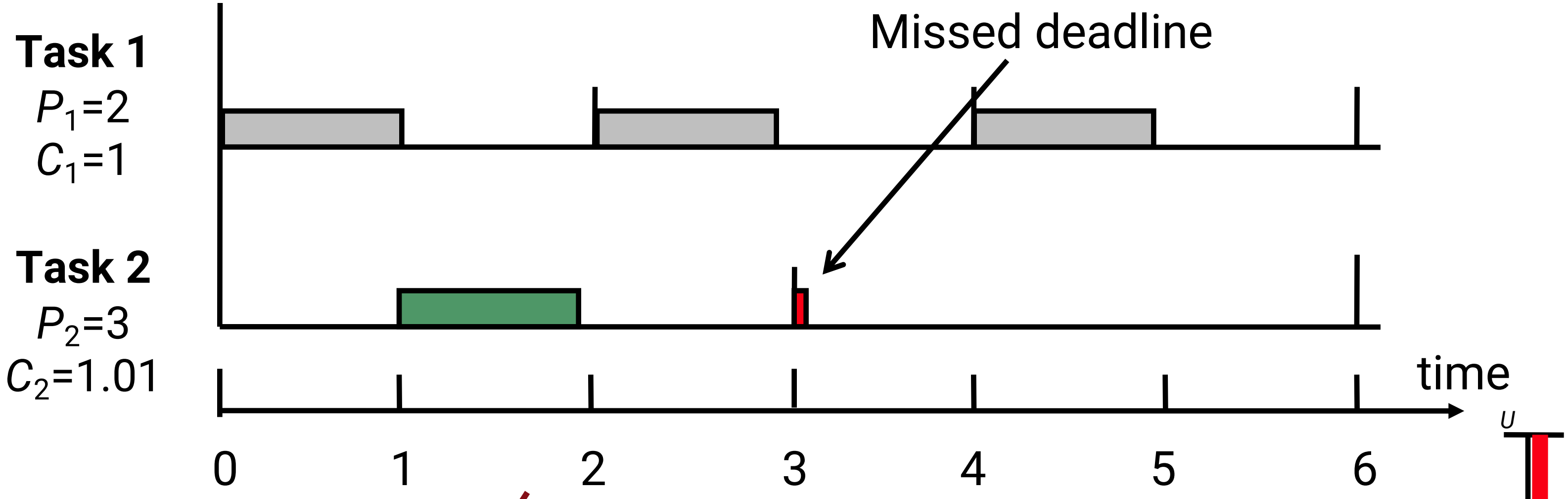
Task 2
P_2=3
C_2=1.01

$$U = \frac{C_1}{P_1} + \frac{C_2}{P_2} = \frac{1}{2} + \frac{1.01}{3} \approx 0.833$$



- Question: Is there a threshold U_{bound} such that
  - When $U < U_{bound}$ deadlines are met
  - When $U > U_{bound}$ deadlines are missed?

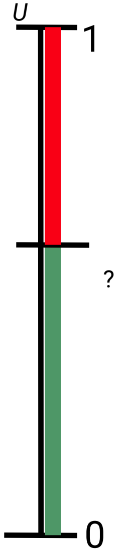
Example (Rate monotonic scheduling)



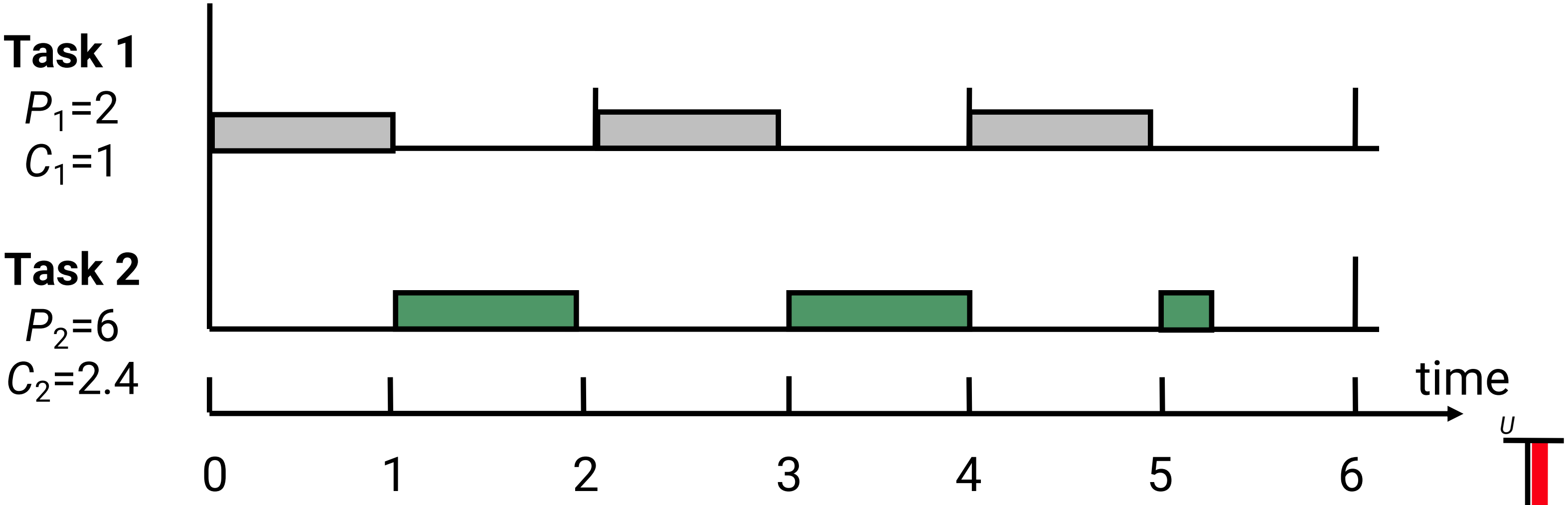
$$U = \frac{C_1}{P_1} + \frac{C_2}{P_2} = \frac{1}{2} + \frac{1.01}{3} \approx \boxed{0.833}$$

Unschedulable

- Question: Is there a threshold U_{bound} such that
  - When $U < U_{bound}$ deadlines are met
  - When $U > U_{bound}$ deadlines are missed?

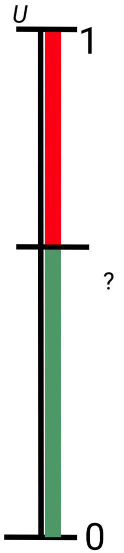


Another example (Rate monotonic scheduling)



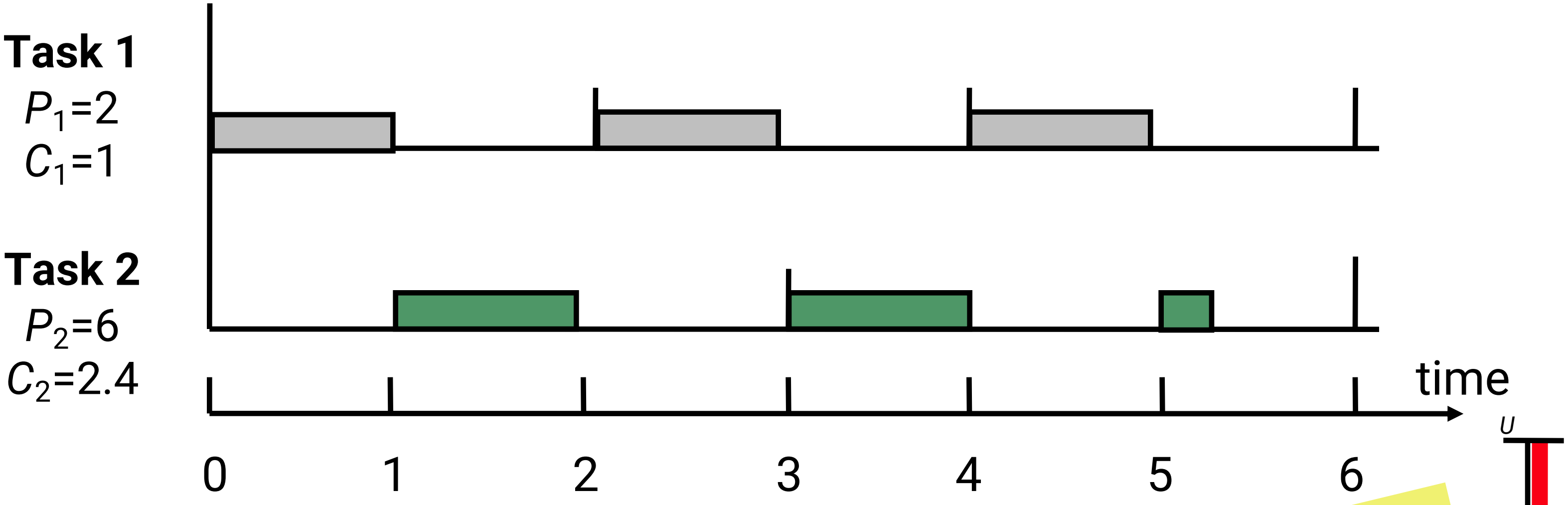
$$U = \frac{C_1}{P_1} + \frac{C_2}{P_2} = \frac{1}{2} + \frac{2.4}{6} = \boxed{0.9}$$

Schedulable



- Question: Is there a threshold U_{bound} such that
  - When $U < U_{bound}$ deadlines are met
  - When $U > U_{bound}$ deadlines are missed

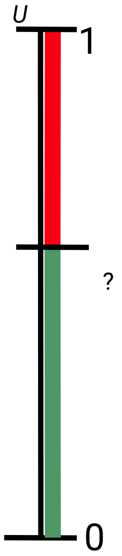
Another example (Rate monotonic scheduling)



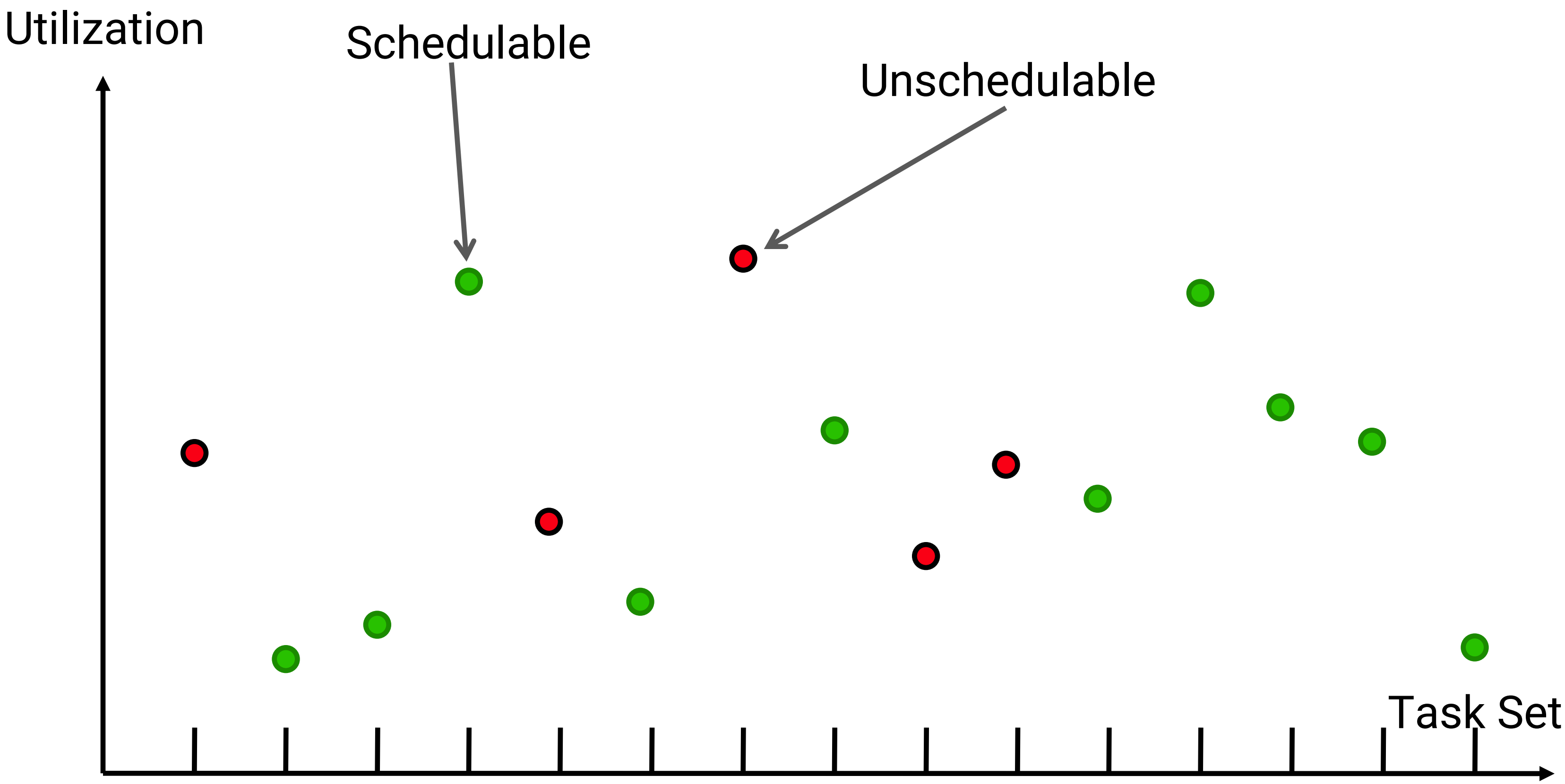
$$U = \frac{C_1}{P_1} + \frac{C_2}{P_2} = \frac{1}{} + \frac{2.4}{}$$

Schedulability depends on the task set.
No clean relationship between utilization and schedulability.

- Question: Is there … such that
  - When $U <$ … deadlines are met
  - When $U > U_{bound}$ deadlines are missed

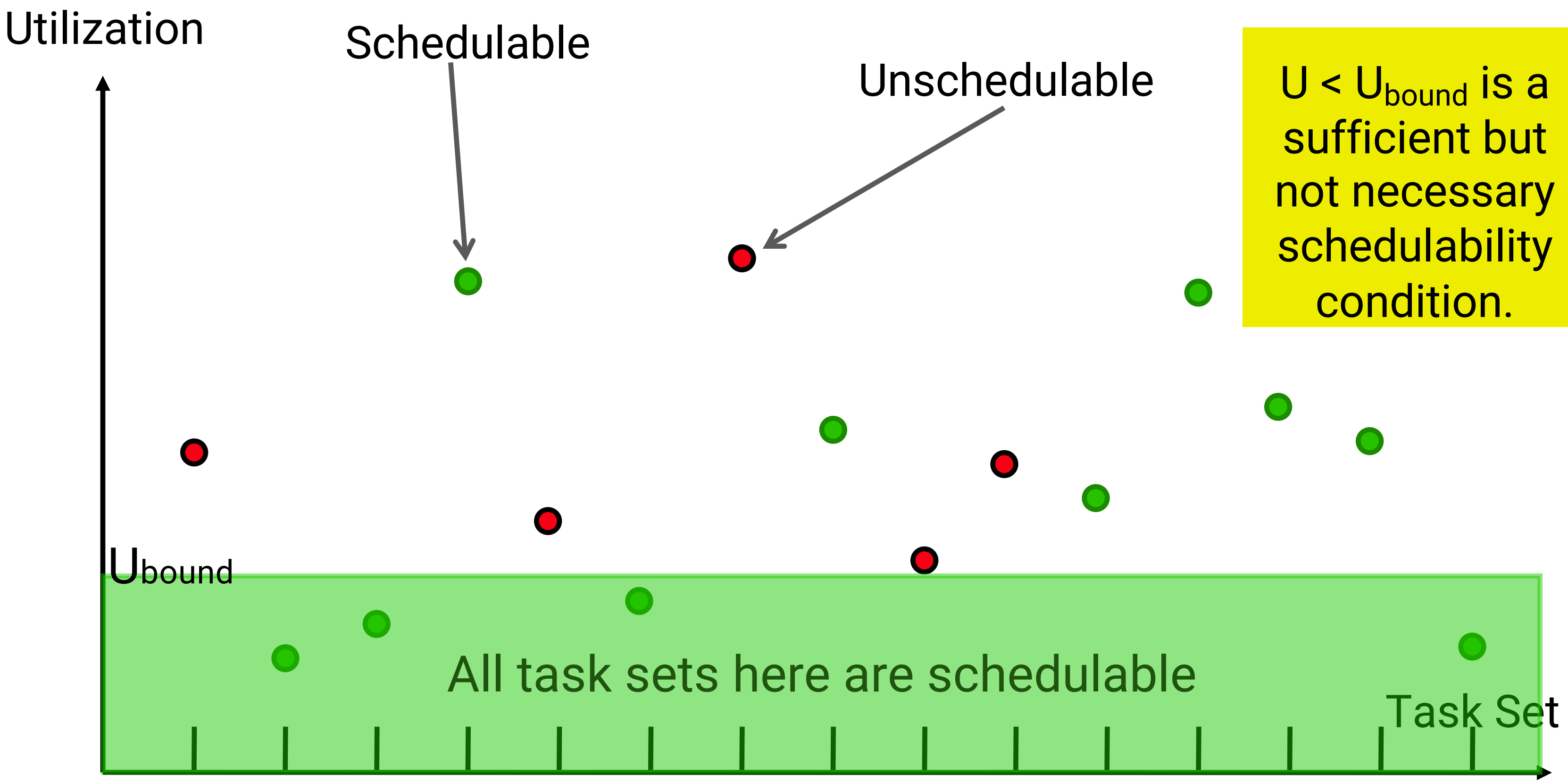


Visualizing schedulability



- Question: is there a threshold U_{bound} such that
  - ~~When $U < U_{bound}$ deadlines are met~~
  - ~~When $U > U_{bound}$ deadlines are missed~~

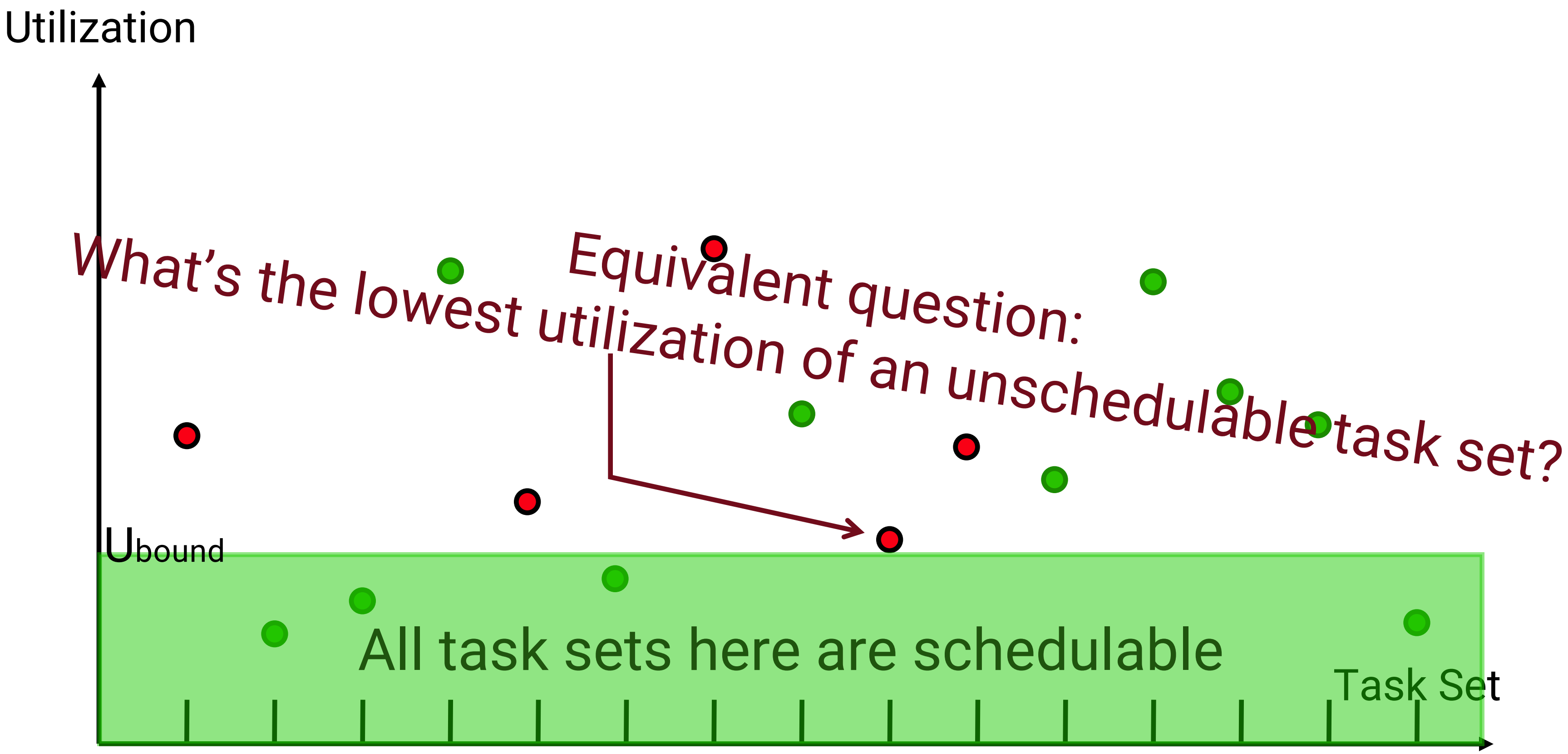
Visualizing schedulability



$U < U_{bound}$ is a sufficient but not necessary schedulability condition.

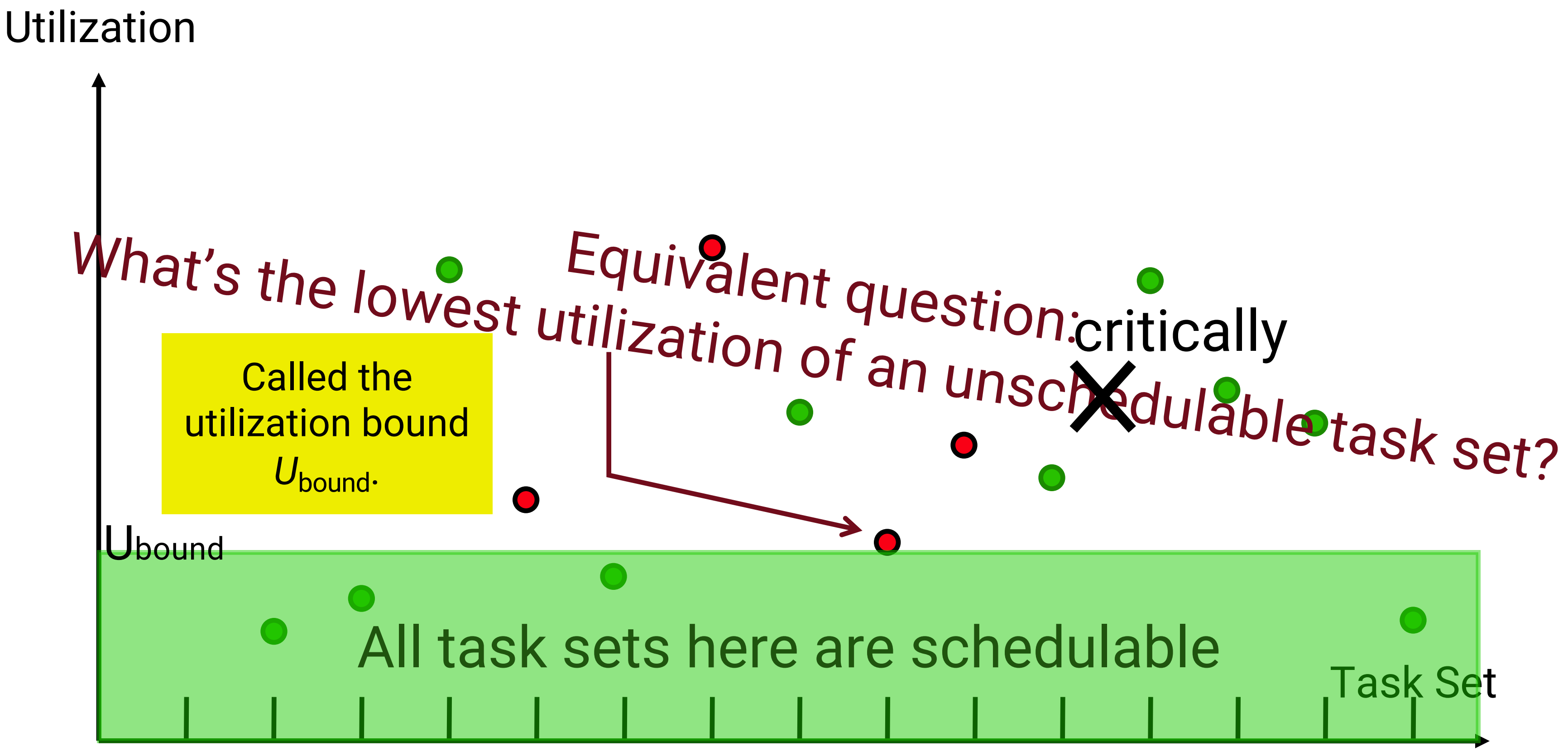
- Modified question: Is there a threshold U_{bound} such that
  - When $U < U_{bound}$ deadlines are met (false negatives are OK, but false positives are NOT OK)
  - When $U > U_{bound}$ deadlines might or might not be missed

Visualizing schedulability


Utilization
What's the lowest utilization of an unschedulable task set?
Equivalent question:
Ubound
All task sets here are schedulable
Task Set

- Modified question: Is there a threshold U_{bound} such that
  - When $U < U_{bound}$ deadlines are met
  - When $U > U_{bound}$ deadlines might or might not be missed

Visualizing schedulability



Equivalent question:

What's the lowest utilization of an unschedulable task set?

critically

Called the utilization bound U_{bound}.

- Modified question: Is there a threshold U_{bound} such that
  - When $U < U_{bound}$ deadlines are met
  - When $U > U_{bound}$ deadlines might or might not be missed

Utilization bound: The Plan

1. Consider a fixed priority assignment S
2. Consider a task set Γ with utilization $U = \sum_{i=1}^{n} c_i/P_i$ that is **feasible** under S
3. **Inflate** execution times of the tasks until Γ **fully utilizes** the processor under S
   - **Meaning:** If execution times of inflated Γ are further increased by any $\epsilon > 0$, then the inflated Γ will become infeasible under S
   - Inflated Γ is said to **fully utilize** the processor under S
     - Also called **critically schedulable** under S

Utilization bound: The Plan

4. Now given the set

$$\mathscr{U} = \left\{ U(\Gamma) : \Gamma \text{ critically schedulable under } S \right\}$$

Q: What utilization value U is such that all task sets whose utilization is $\leq U$ are feasible under S?

$$U_{\text{lub}}(S) = \inf \mathscr{U}$$

least **u**pper **b**ound

Intuition: Any taskset that is generated from $\mathscr{U}$ by reducing execution times and holding the periods fixed is feasible (less workload → less interference)

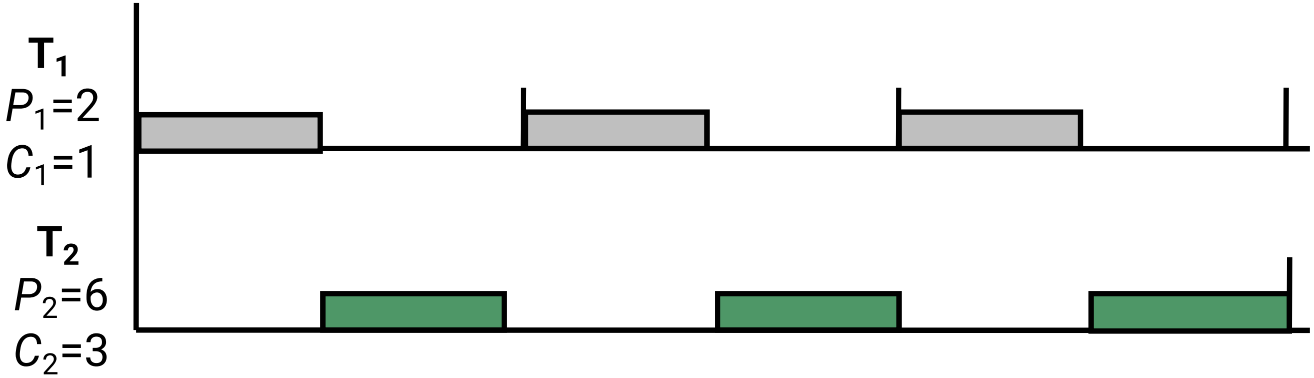
Utilization bound: The Plan

$$\mathscr{U} = \left\{ U(\Gamma) : \Gamma \text{ critically schedulable under } S \right\}$$

$$U_{\text{lub}}(S) = \inf \mathscr{U}$$

Why not take the sup (max) of the critically schedulable set instead of inf (min)?

Consider two tasks T_1 = (1, 2) and T_2 = (3, 6) ➔ U = 1 (critically schedulable and achieves the sup)

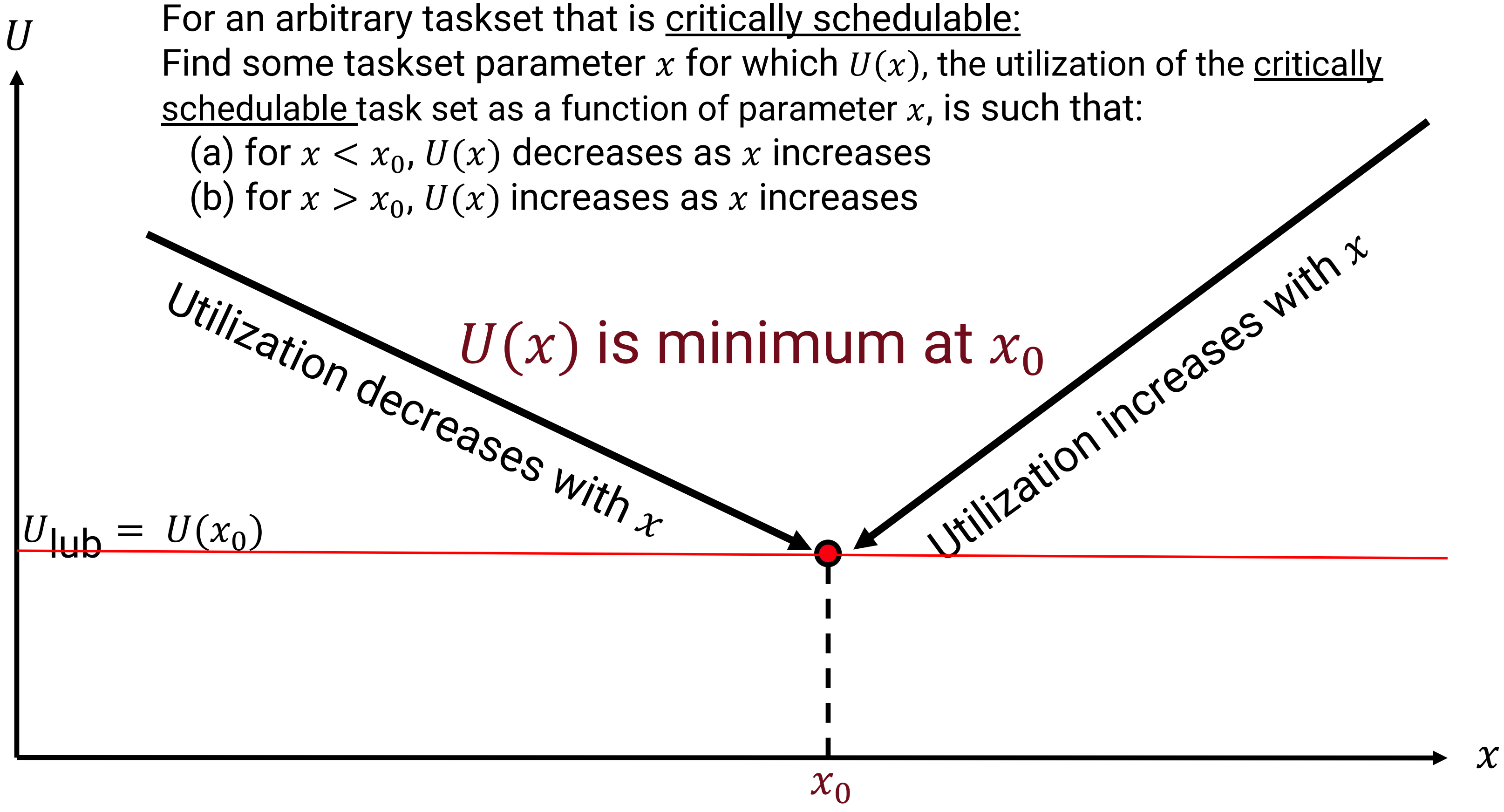


But we saw that taskset T_1 = (1, 2) and T_2 = (2.4, 6) with U = 0.833 is NOT schedulable under RM!

How we will carry out the plan

For an arbitrary taskset that is <u>critically schedulable:</u>
Find some taskset parameter x for which $U(x)$, the utilization of the <u>critically schedulable</u> task set as a function of parameter x, is such that:

(a) for $x < x_0$, $U(x)$ decreases as x increases
(b) for $x > x_0$, $U(x)$ increases as x increases



$U(x)$ is minimum at x_0

U
Utilization decreases with x
Utilization increases with x
$U_{\text{lub}} = U(x_0)$
x_0
x

Deriving $U_{\mathrm{lub}}(\mathrm{RM})$

- Two tasks case first
- Consider the same two cases as in proof of optimality in previous lecture

Finding the utilization bound for RM scheduling

U(x): utilization of <u>critically schedulable</u> taskset

Find some taskset parameter *x*

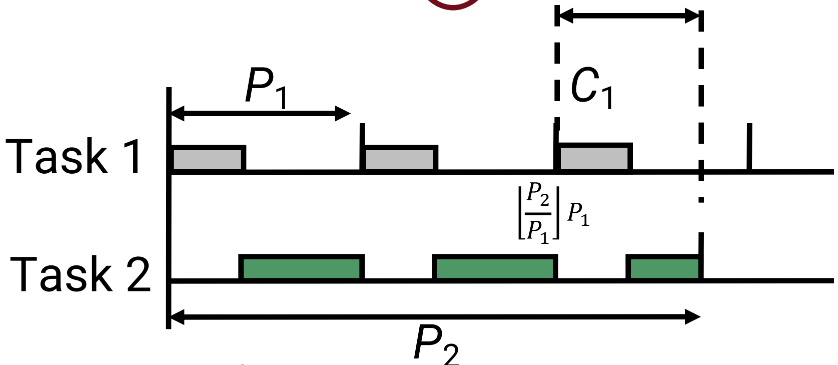
such that

- Case (a): *x*<*x*$_o$ such that *U(x)* decreases as *x* increases
- Case (b): *x*>*x*$_o$ such that *U(x)* increases as *x* increases

Thus *U(x)* is minimum when *x*=*x*$_o$ → Find *U(x$_o$)*

$$C_1 \leqslant P_2 - \left\lfloor \frac{P_2}{P_1} \right\rfloor P_1$$

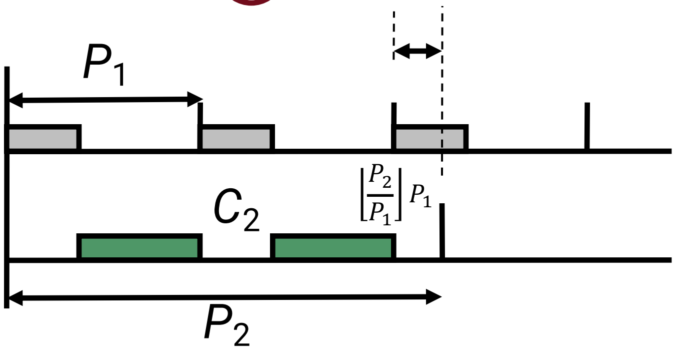
$$C_1 > P_2 - \left\lfloor \frac{P_2}{P_1} \right\rfloor P_1$$



For such C_1: Inflate C_2 so that the taskset is <u>critically schedulable</u>

$$C_2 = P_2 - C_1 \left\lceil \frac{P_2}{P_1} \right\rceil = P_2 - C_1 \left(\left\lfloor \frac{P_2}{P_1} \right\rfloor + 1 \right)$$

$$U = \frac{C_1}{P_1} + \frac{C_2}{P_2} = 1 + \frac{C_1}{P_2} \left(\frac{P_2}{P_1} - \left\lfloor \frac{P_2}{P_1} \right\rfloor - 1 \right)$$



$$C_2 = (P_1 - C_1) \left\lfloor \frac{P_2}{P_1} \right\rfloor$$

$$U = \frac{P_1}{P_2} \left\lfloor \frac{P_2}{P_1} \right\rfloor + \frac{C_1}{P_2} \left(\frac{P_2}{P_1} - \left\lfloor \frac{P_2}{P_1} \right\rfloor \right)$$

Finding the utilization bound for RM scheduling

Case 1:

$$C_1 \leqslant P_2 - \left\lfloor \frac{P_2}{P_1} \right\rfloor P_1$$

$$C_2 = P_2 - C_1 \left\lceil \frac{P_2}{P_1} \right\rceil = P_2 - C_1 \left(\left\lfloor \frac{P_2}{P_1} \right\rfloor + 1 \right)$$

$$U = \frac{C_1}{P_1} + \frac{C_2}{P_2} = 1 + \frac{C_1}{P_2} \left(\frac{P_2}{P_1} - \left\lfloor \frac{P_2}{P_1} \right\rfloor - 1 \right)$$

$$U = 1 + \frac{P_1}{P_2} \left(\frac{P_2}{P_1} - \left\lfloor \frac{P_2}{P_1} \right\rfloor \right) \left(\frac{P_2}{P_1} - \left\lfloor \frac{P_2}{P_1} \right\rfloor - 1 \right)$$

What value of *C*$_1$ minimizes *U*?
- Term in ( ) is < 0
- $C_1 \mapsto U$ is a decreasing map ↓

The same C_1 minimizes *U* of **case 2** and the resulting *U*(C_1) for case 2 is the same as that of case 1

$$C_1 = P_2 - \left\lfloor \frac{P_2}{P_1} \right\rfloor P_1 = P_1 \left(\frac{P_2}{P_1} - \left\lfloor \frac{P_2}{P_1} \right\rfloor \right)$$

Finding the utilization bound for RM scheduling

Case 1:

$$U = 1 + \frac{P_1}{P_2}\left(\frac{P_2}{P_1} - \left\lfloor\frac{P_2}{P_1}\right\rfloor\right)\left(\frac{P_2}{P_1} - \left\lfloor\frac{P_2}{P_1}\right\rfloor - 1\right)$$

To minimize *U*, we must have

$$\left\lfloor\frac{P_2}{P_1}\right\rfloor = 1$$

Why?

$0 \le G < 1$ and so is $1 - G \Rightarrow G(1 - G) \ge 0$

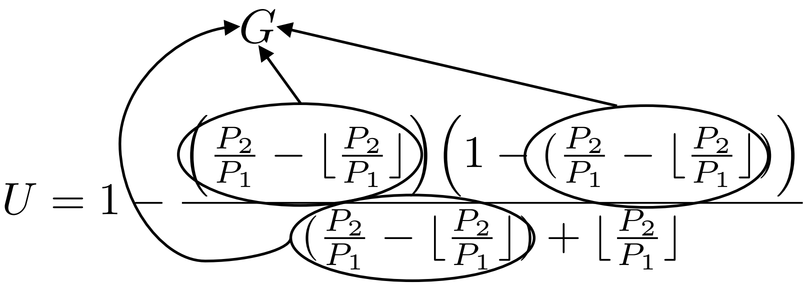
Then $\left\lfloor\frac{P_2}{P_1}\right\rfloor \mapsto U$ is increasing ↑

U minimized at minimum possible $\left\lfloor\frac{P_2}{P_1}\right\rfloor$

But $P_2 \ge P_1$ (RM) $\Rightarrow \left\lfloor\frac{P_2}{P_1}\right\rfloor \ge 1$

Then min. possible $\left\lfloor\frac{P_2}{P_1}\right\rfloor$ is 1

$$U = 1 - \frac{\left(\frac{P_2}{P_1} - \lfloor\frac{P_2}{P_1}\rfloor\right)\left(1 - (\frac{P_2}{P_1} - \lfloor\frac{P_2}{P_1}\rfloor)\right)}{P_2/P_1}$$



$$U = 1 - \frac{\left(\frac{P_2}{P_1} - \lfloor\frac{P_2}{P_1}\rfloor\right)\left(1 - (\frac{P_2}{P_1} - \lfloor\frac{P_2}{P_1}\rfloor)\right)}{(\frac{P_2}{P_1} - \lfloor\frac{P_2}{P_1}\rfloor) + \lfloor\frac{P_2}{P_1}\rfloor}$$

$$U = 1 - \frac{G(1 - G)}{G + \lfloor\frac{P_2}{P_1}\rfloor}$$

Finding the utilization bound for RM scheduling

The minimum utilization case

$$U = 1 - \frac{\left(\frac{P_2}{P_1} - \lfloor\frac{P_2}{P_1}\rfloor\right)\left(1 - (\frac{P_2}{P_1} - \lfloor\frac{P_2}{P_1}\rfloor)\right)}{P_2/P_1}$$

To minimize U, we must have $\left\lfloor\frac{P_2}{P_1}\right\rfloor = 1$

$$U = 1 + \frac{(P_2/P_1 - 1)(P_2/P_1 - 2)}{P_2/P_1}$$

Real-valued function in one variable P_2/P_1

Differentiable!

Then $\frac{dU}{d(P_2/P_1)} = 0 \Rightarrow \frac{P_2}{P_1} = \sqrt{2}$

Finally, $U = 0.83$

Finding the utilization bound for RM scheduling

$$U = 1 + \frac{P_1}{P_2}\left(\frac{P_2}{P_1} - \left\lfloor \frac{P_2}{P_1} \right\rfloor\right)\left(\frac{P_2}{P_1} - \left\lfloor \frac{P_2}{P_1} \right\rfloor - 1\right)$$

Observe:

If P_2 is an integer multiple of $P_1 \Rightarrow \frac{P_2}{P_1} = \left\lfloor \frac{P_2}{P_1} \right\rfloor \Rightarrow U = 1$

Generalization for *n* tasks

What were the worst case conditions for the 2-task case?

$$\begin{cases} \left\lfloor \frac{P_2}{P_1} \right\rfloor = 1 \\ C_1 = P_2 - \left\lfloor \frac{P_2}{P_1} \right\rfloor P_1 = P_2 - P_1 \\ C_2 = (P_1 - C_1)\left\lfloor \frac{P_2}{P_1} \right\rfloor = P_1 - C_1 \end{cases} \quad \Longleftrightarrow \quad \begin{cases} P_1 < P_2 < 2P_1 \\ C_1 = P_2 - P_1 \\ C_2 = P_1 - C_1 = 2P_1 - P_2 \end{cases}$$

Generalizing to n tasks

$$\begin{cases} P_1 < P_n < 2P_1 \\ C_1 = P_2 - P_1 \\ C_2 = P_3 - P_2 \\ C_3 = P_4 - P_3 \\ \qquad \vdots \\ C_{n-1} = P_n - P_{n-1} \\ C_n = P_1 - (C_1 + \cdots + C_{n-1}) = 2P_1 - P_n \end{cases}$$

Generalization for n tasks

$$\begin{cases} P_1 < P_n < 2P_1 \\ C_1 = P_2 - P_1 \\ C_2 = P_3 - P_2 \\ C_3 = P_4 - P_3 \\ \qquad \vdots \\ C_{n-1} = P_n - P_{n-1} \\ C_n = P_1 - (C_1 + \cdots + C_{n-1}) = 2P_1 - P_n \end{cases}$$

$$\begin{aligned} U &= \frac{C_1}{P_1} + \frac{C_2}{P_2} + \cdots \frac{C_n}{P_n} \\ &= \frac{P_2 - P_1}{P_1} + \frac{P_3 - P_2}{P_2} + \cdots + \frac{2P_1 - P_n}{P_n} \\ &= \frac{P_2}{P_1} + \frac{P_3}{P_2} + \cdots + \frac{2P_1}{P_n} - n \end{aligned}$$

Generalization for n tasks

$$U = \frac{P_2}{P_1} + \frac{P_3}{P_2} + \cdots + \frac{2P_1}{P_n} - n$$

$$\frac{\partial U}{\partial (P_2/P_1)} = 0; \frac{\partial U}{\partial (P_3/P_2)} = 0; \ldots ; \frac{\partial U}{\partial (P_n/P_1)} = 0$$

We can then obtain

Liu & Layland, 1973

$$\frac{P_{i+1}}{P_i} = 2^{1/n} \Rightarrow U = n(2^{1/n} - 1)$$

For large *n*: $\lim_{n \to \infty} U = \lim_{n \to \infty} n(2^{1/n} - 1) = \ln 2 \approx 0.69$

Sanity check

Let $b_n = \inf\{U(\Gamma) : \Gamma \text{ is a critically schedulable set of } n \text{ tasks}\}$

Let U_n be utilization factor of n tasks

Question: If $U_n \leq b_n$, is the task set schedulable?

Lecture summary

- Understanding utilization bounds
- The utilization bound for rate-monotonic scheduling
- For RM scheduling the bound decreases with the number of tasks, approaching an asymptotic limit of 0.69
- Coming up: Why is RM priority assignment the optimal static priority policy? Are there better schedulability tests?