

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

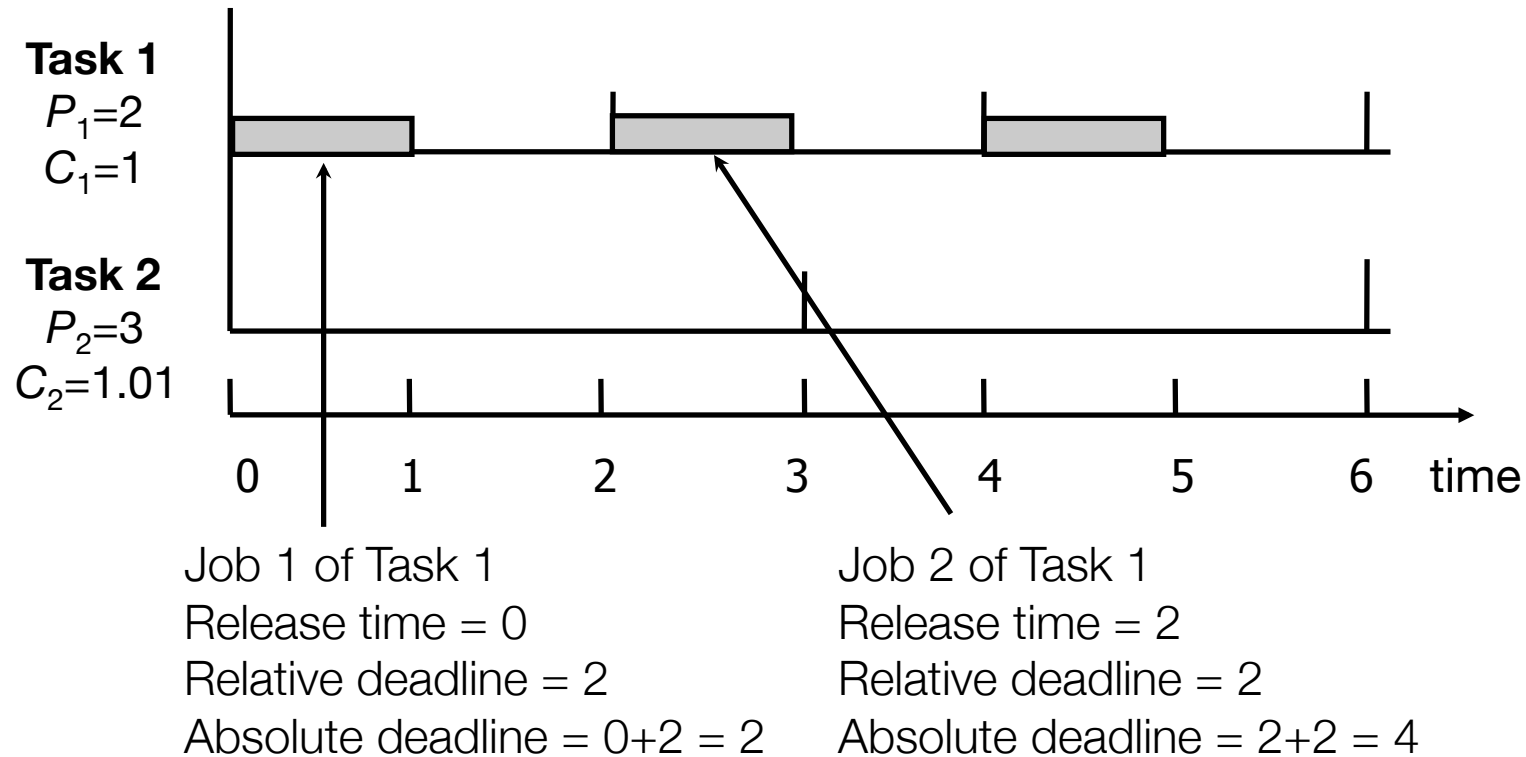
- Utilization (U) and schedulability

- Main problem: Find *Bound* for scheduling policy such that
 - $U < \text{Bound} \rightarrow$ All deadlines met!

- Optimality of EDF scheduling

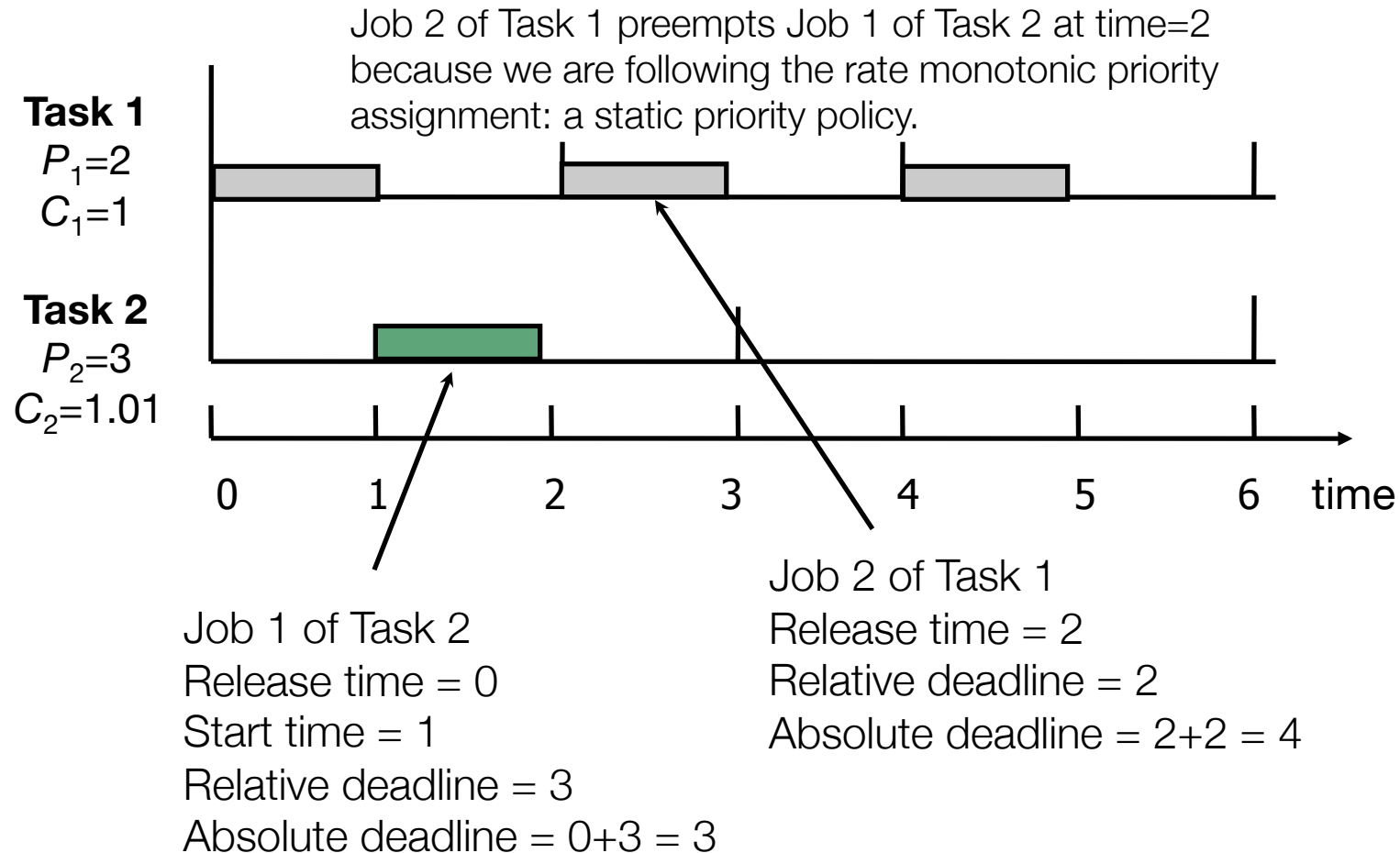
- $\text{Bound}_{EDF} = 100\%$

A quick refresher



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



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)
 - Heuristics
- **Two most important scheduling policies**
 - Earliest deadline first (dynamic)
 - Rate monotonic (static)

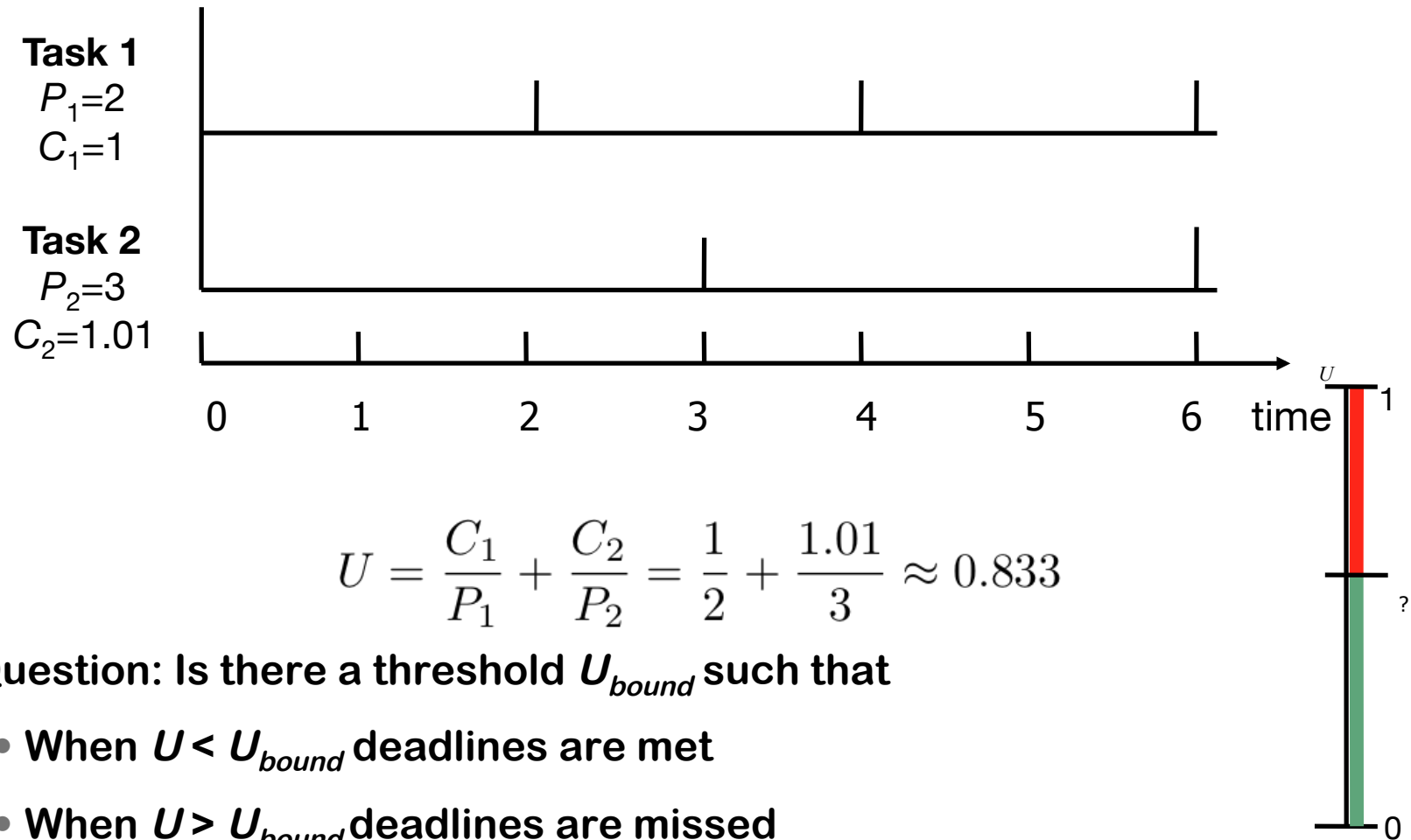
Schedulability analysis of periodic tasks

- 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)
 - Heuristics
- Two most important scheduling policies
 - Earliest deadline first (Dynamic)
 - Rate monotonic (static)

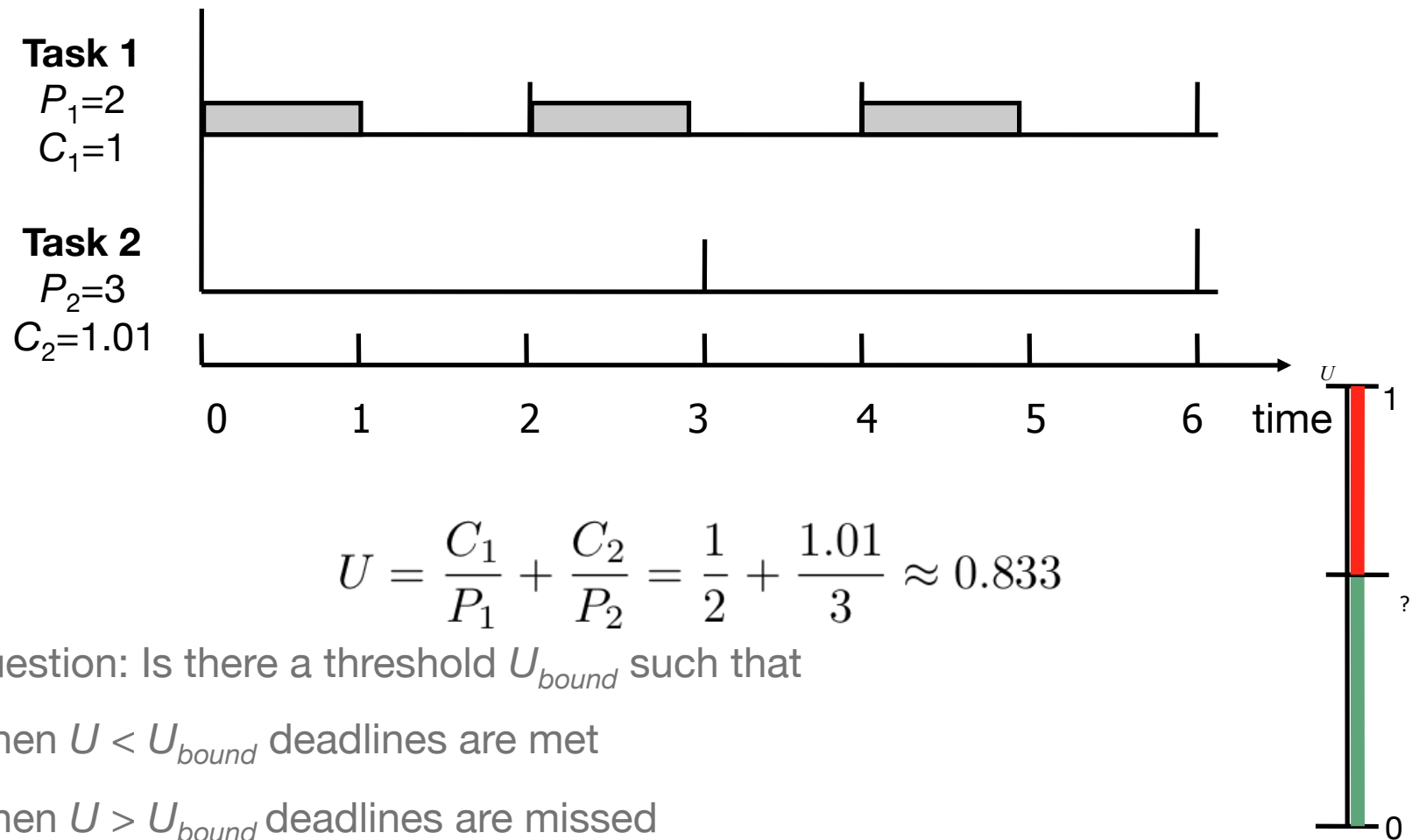
Utilization bounds

- 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

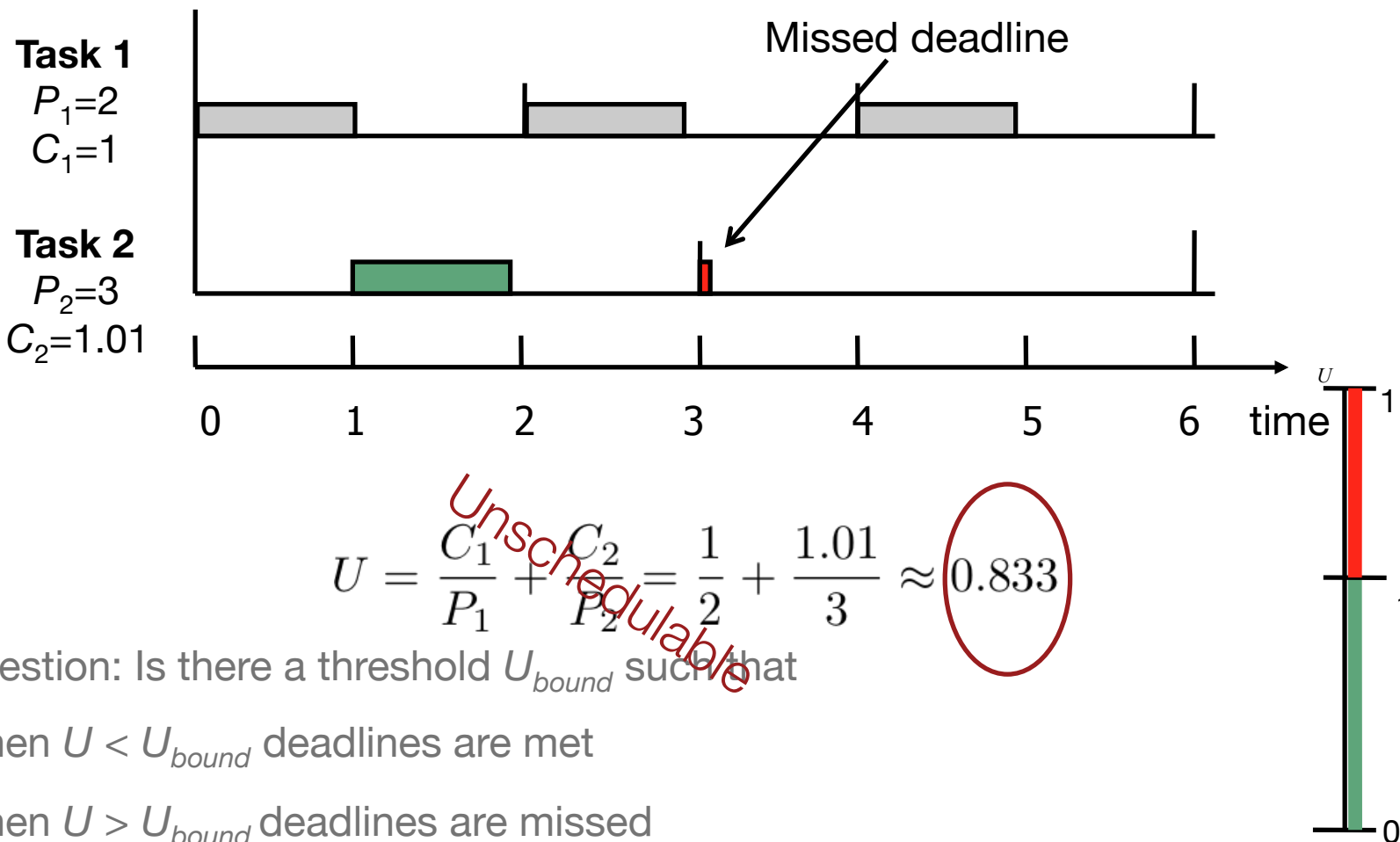
Example (Rate monotonic scheduling)



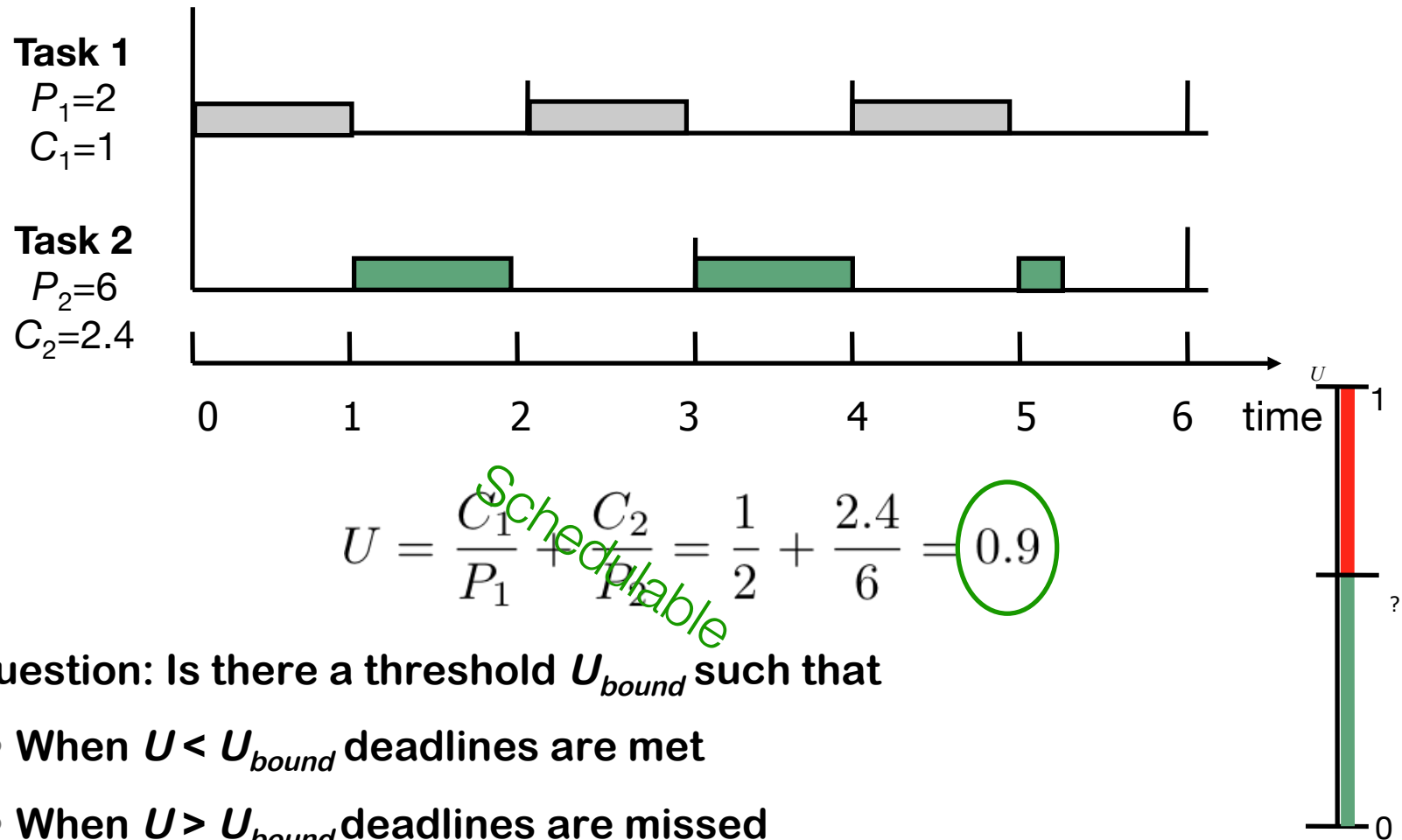
Example (Rate monotonic scheduling)



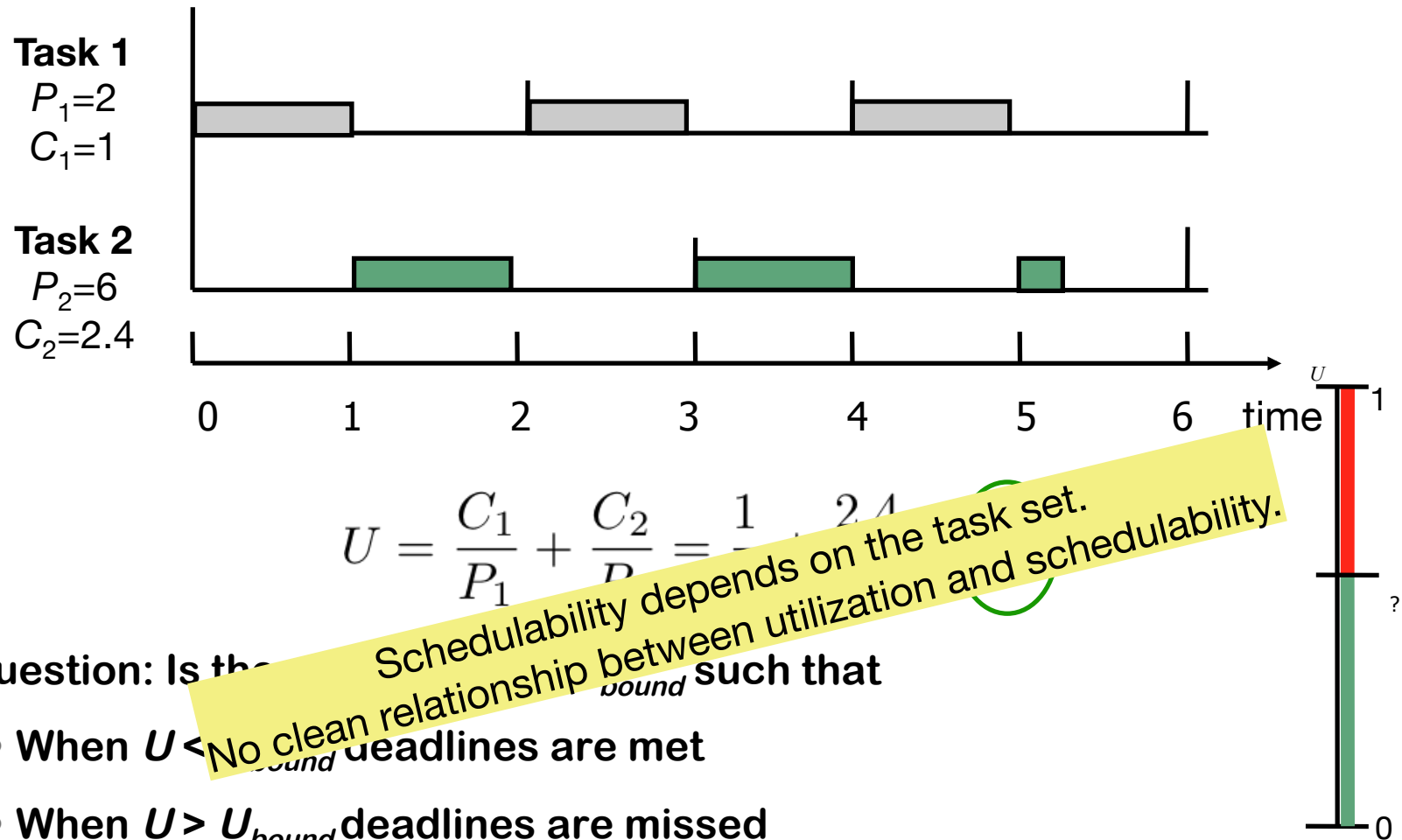
Example (Rate monotonic scheduling)



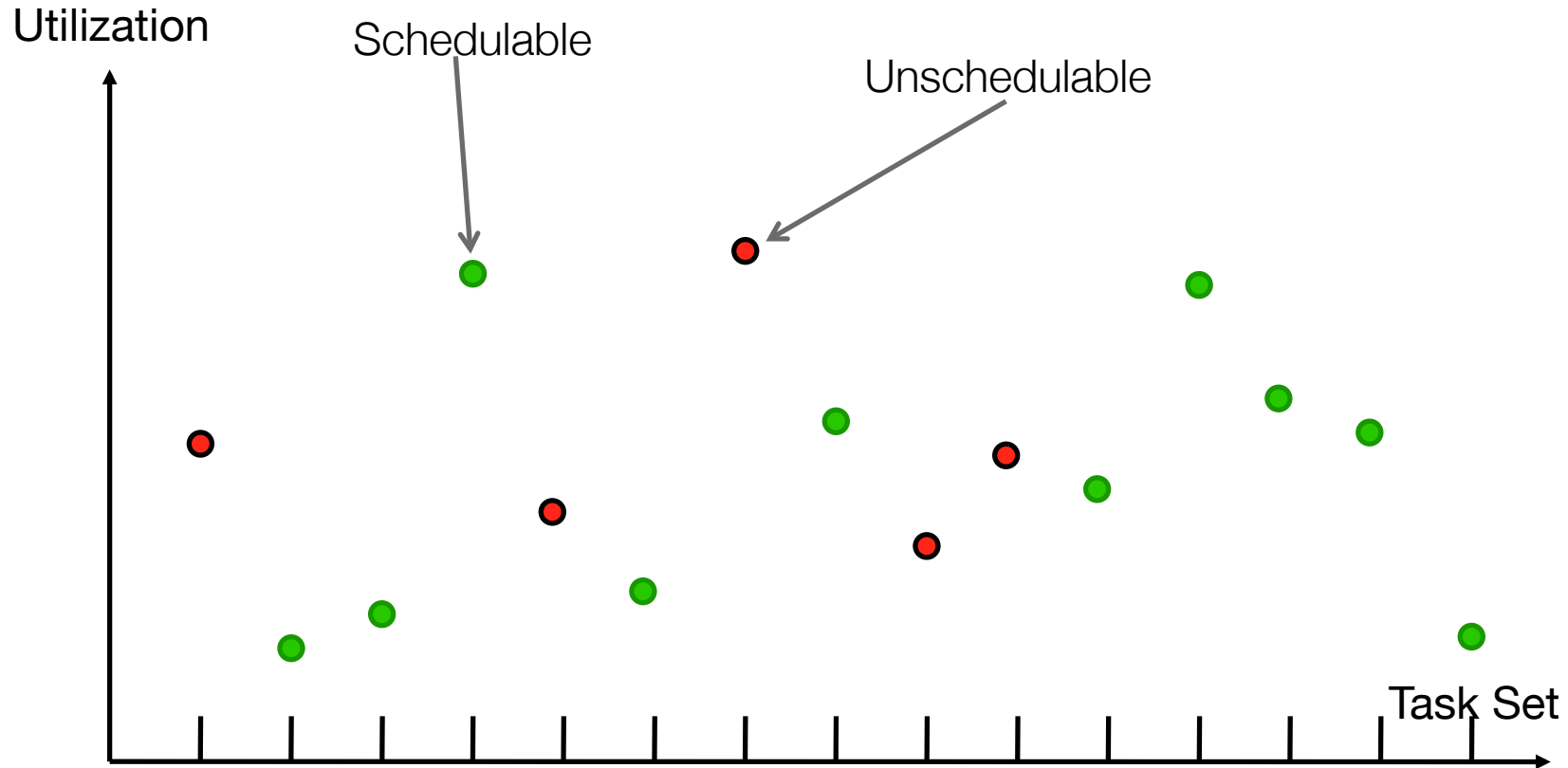
Another example (Rate monotonic scheduling)



Another example (Rate monotonic scheduling)

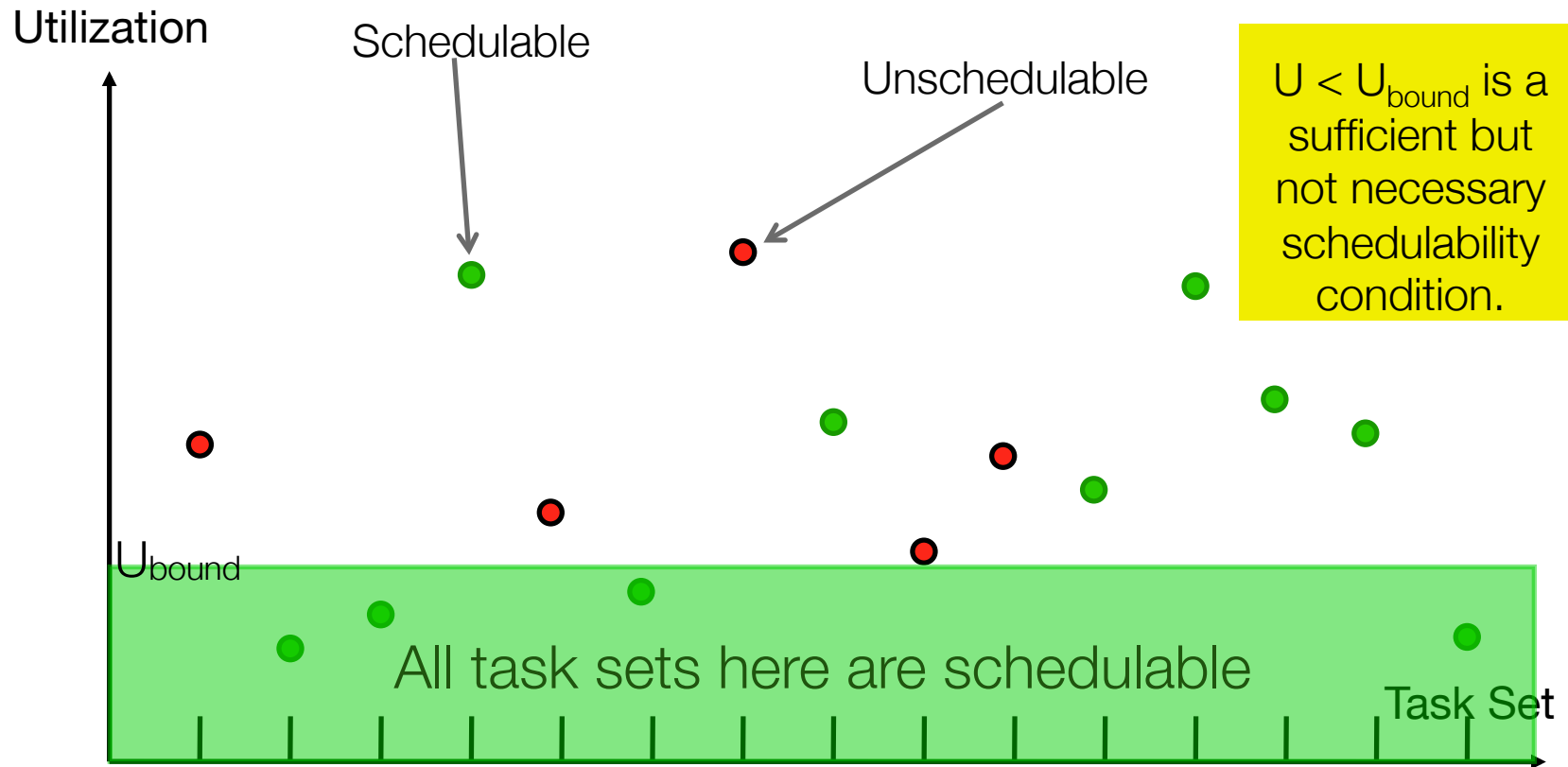


Visualizing schedulability



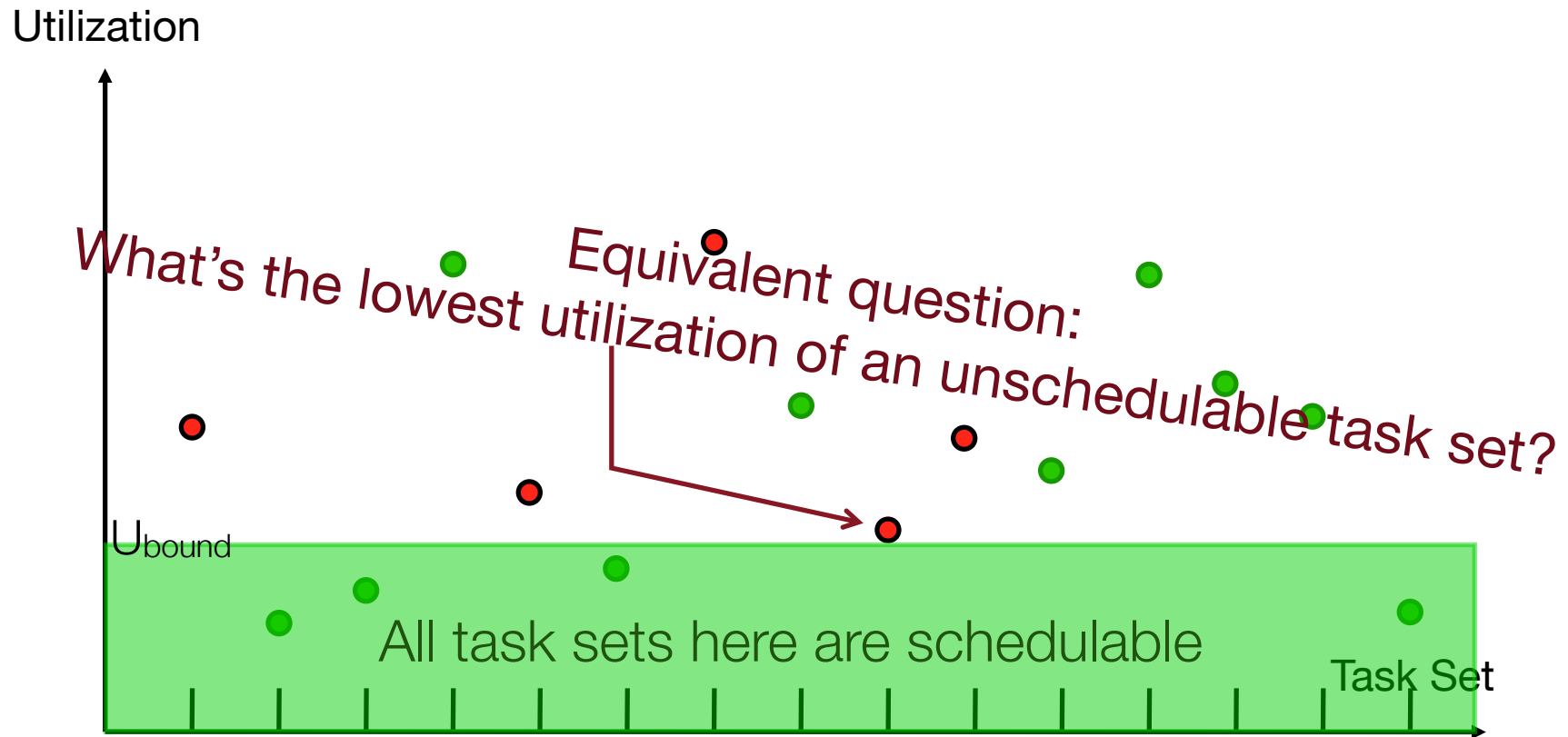
- Question: is there a threshold U_{bound} such that
 - When $U < U_{bound}$ deadlines are met
 - When $U > U_{bound}$ deadlines are missed

Visualizing schedulability



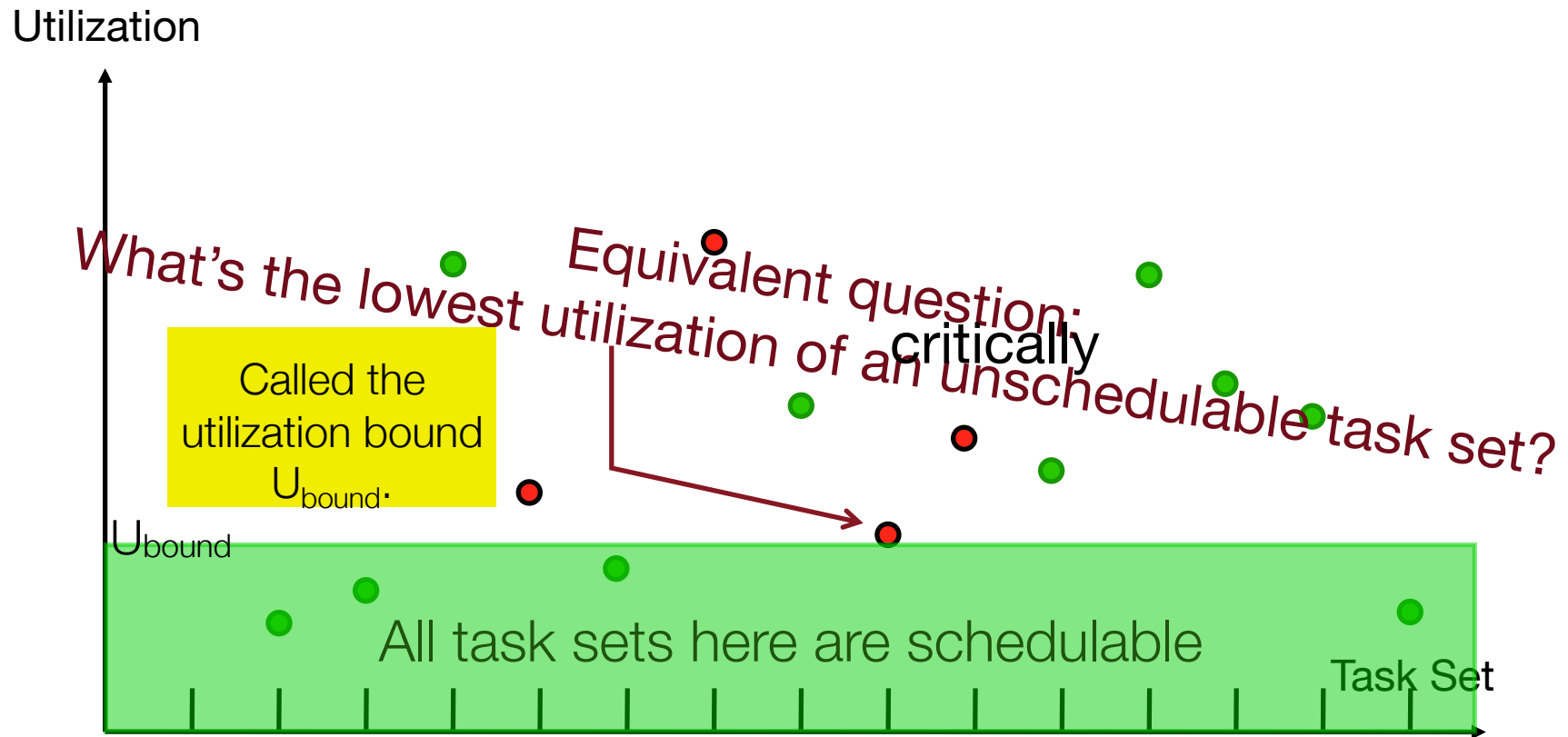
- **Modified** question: Is there a threshold U_{bound} such that
- When $U < U_{\text{bound}}$ deadlines are met
- When $U > U_{\text{bound}}$ deadlines **may or may not be** missed

Visualizing schedulability



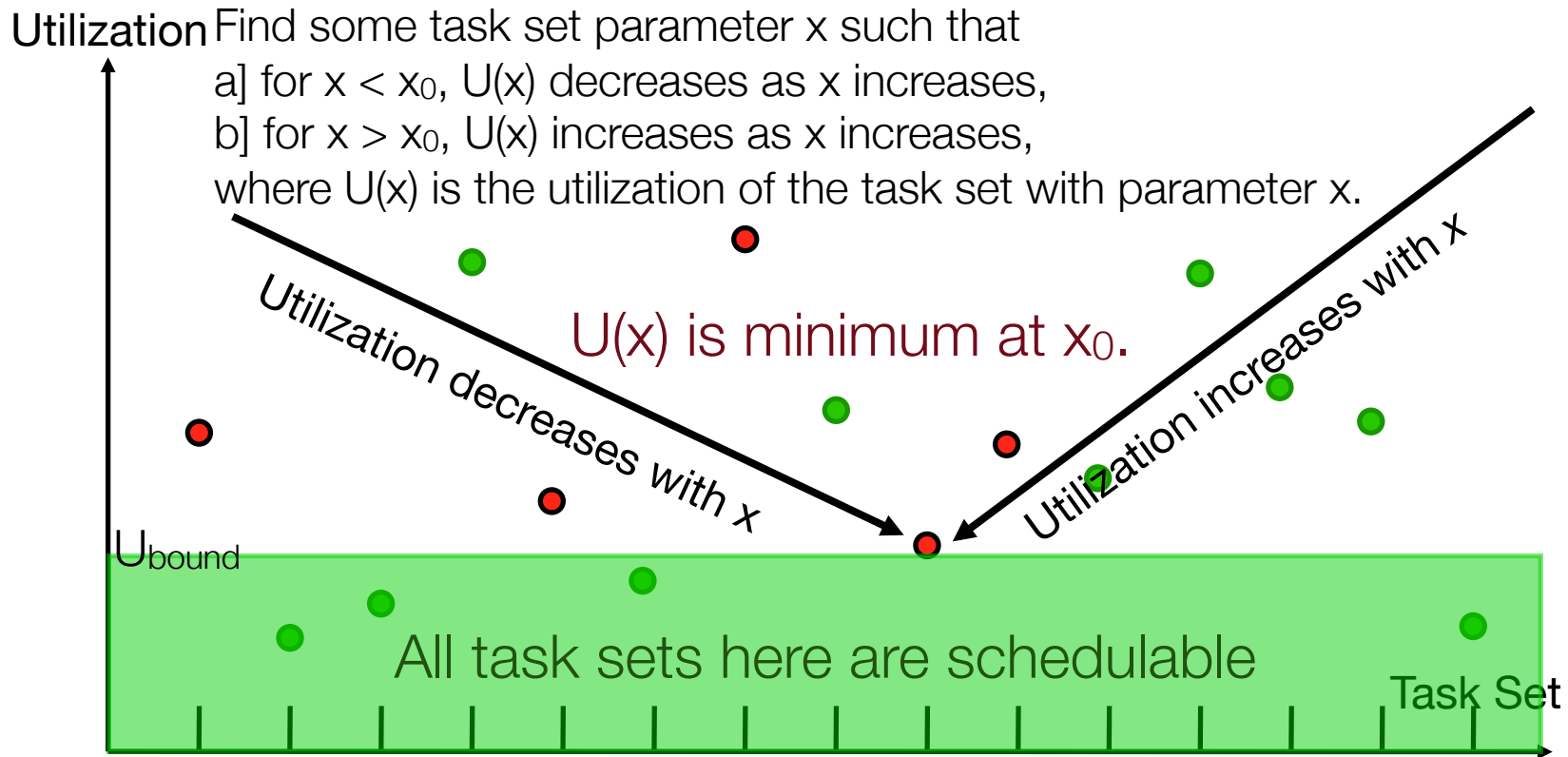
- **Modified** question: Is there a threshold U_{bound} such that
- When $U < U_{bound}$ deadlines are met
- When $U > U_{bound}$ deadlines **may or may not be** missed

Visualizing schedulability



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

Finding the utilization bound for RM scheduling



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

Finding the utilization bound for RM scheduling

- Consider the simplest case with two tasks

Find some task set parameter x
such that

Case (a): $x < x_0$ such that $U(x)$ decreases as x increases

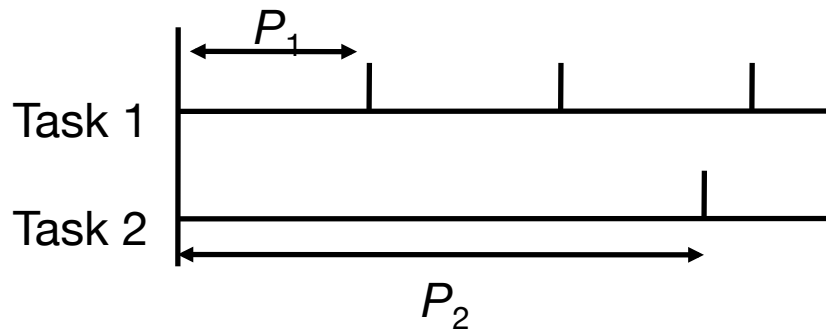
Case (b): $x > x_0$ such that $U(x)$ increases as x increases

Thus $U(x)$ is minimum when $x = x_0$

Find $U(x_0)$

Finding the utilization bound for RM scheduling

- Consider the simplest case with two tasks



Find some task set parameter x such that

Case (a): $x < x_0$ such that $U(x)$ decreases as x increases

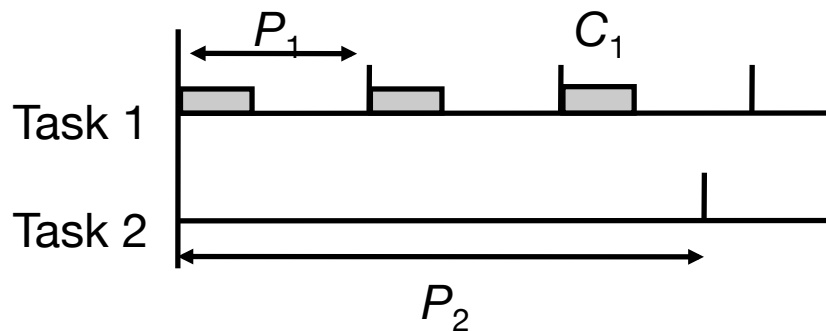
Case (b): $x > x_0$ such that $U(x)$ increases as x increases

Thus $U(x)$ is minimum when $x = x_0$

Find $U(x_0)$

Finding the utilization bound for RM scheduling

- Consider the simplest case with two tasks



Find some task set parameter x such that

Case (a): $x < x_0$ such that $U(x)$ decreases as x increases

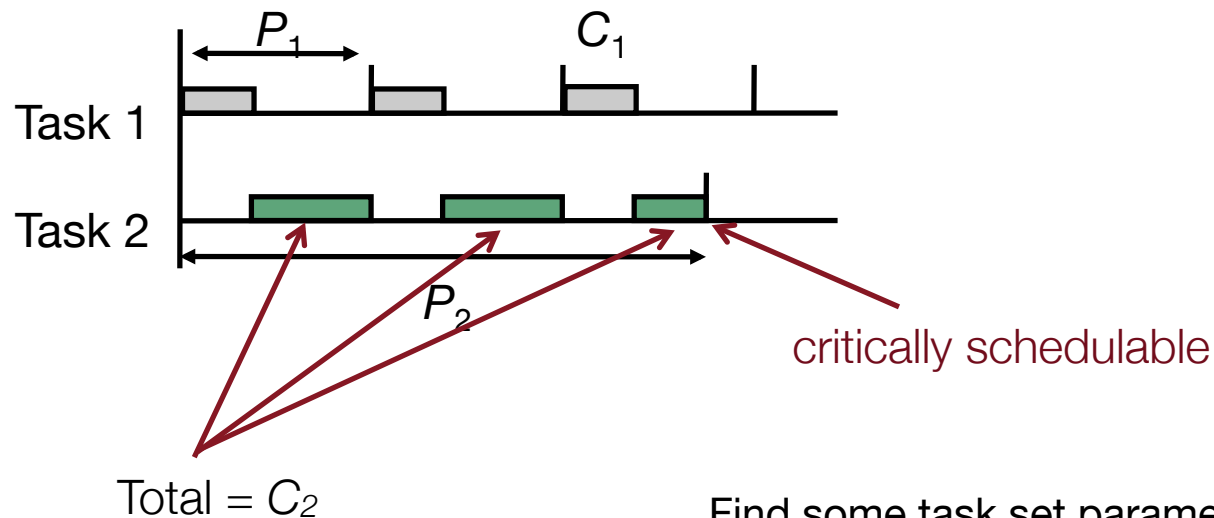
Case (b): $x > x_0$ such that $U(x)$ increases as x increases

Thus $U(x)$ is minimum when $x = x_0$

Find $U(x_0)$

Finding the utilization bound for RM scheduling

- Consider the simplest case with two tasks



Find some task set parameter x such that

Case (a): $x < x_0$ such that $U(x)$ decreases as x increases

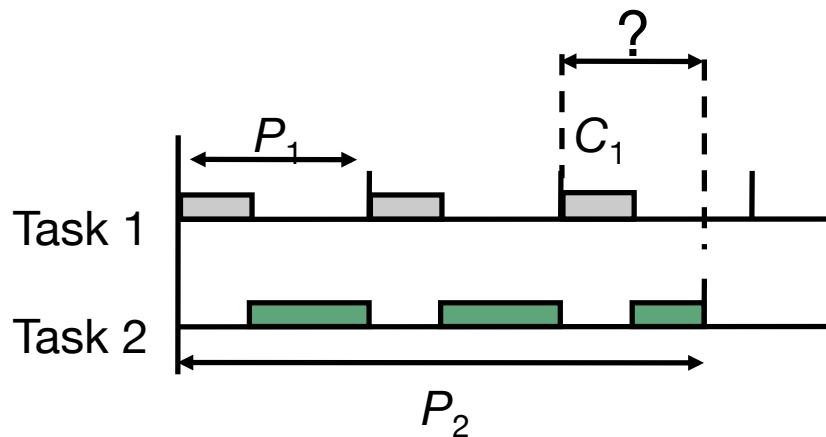
Case (b): $x > x_0$ such that $U(x)$ increases as x increases

Thus $U(x)$ is minimum when $x = x_0$

Find $U(x_0)$

Finding the utilization bound for RM scheduling

- Consider the simplest case with two tasks



Find some task set parameter x such that

Case (a): $x < x_0$ such that $U(x)$ decreases as x increases

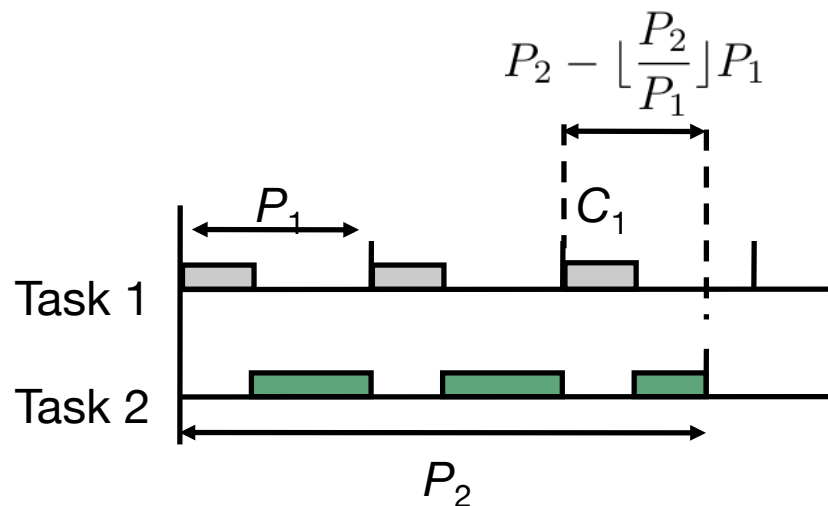
Case (b): $x > x_0$ such that $U(x)$ increases as x increases

Thus $U(x)$ is minimum when $x = x_0$

Find $U(x_0)$

Finding the utilization bound for RM scheduling

- Consider the simplest case with two tasks



Find some task set parameter x such that

Case (a): $x < x_0$ such that $U(x)$ decreases as x increases

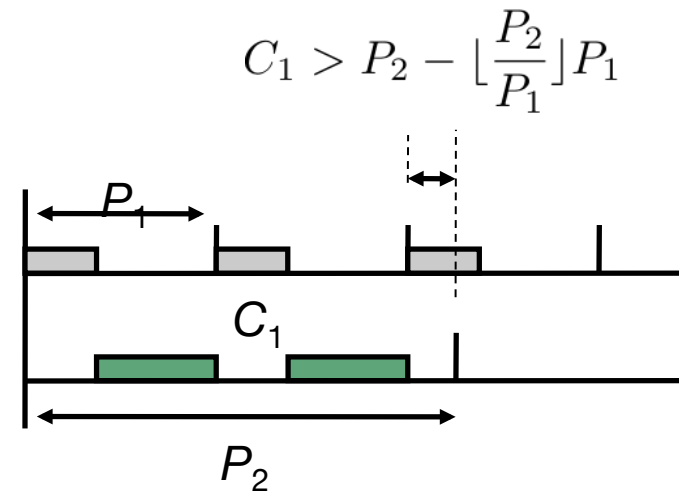
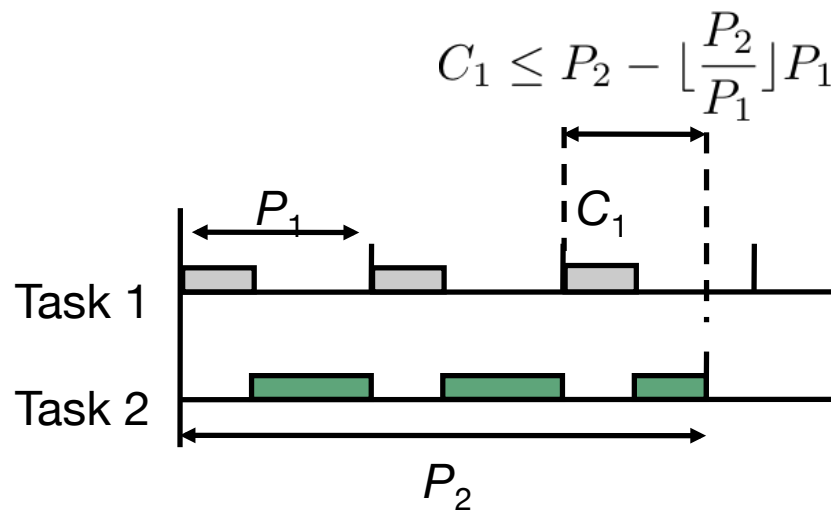
Case (b): $x > x_0$ such that $U(x)$ increases as x increases

Thus $U(x)$ is minimum when $x = x_0$

Find $U(x_0)$

Finding the utilization bound for RM scheduling

- Consider the simplest case with two tasks
 - There are two sub-cases...



Find some task set parameter x
such that

Case (a): $x < x_0$ such that $U(x)$ decreases as x increases

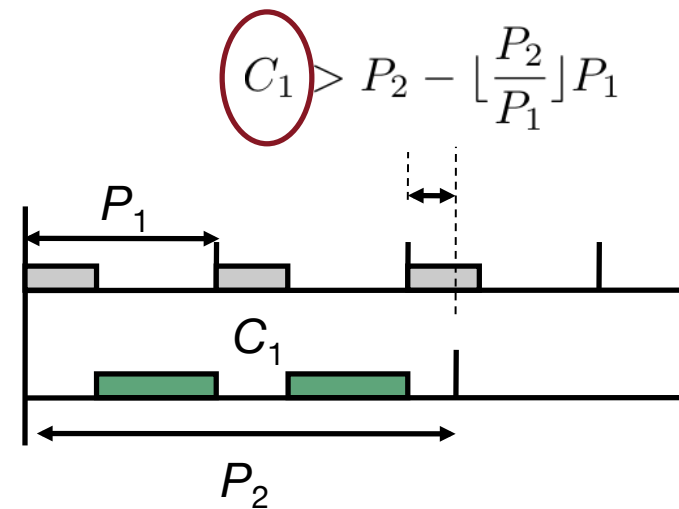
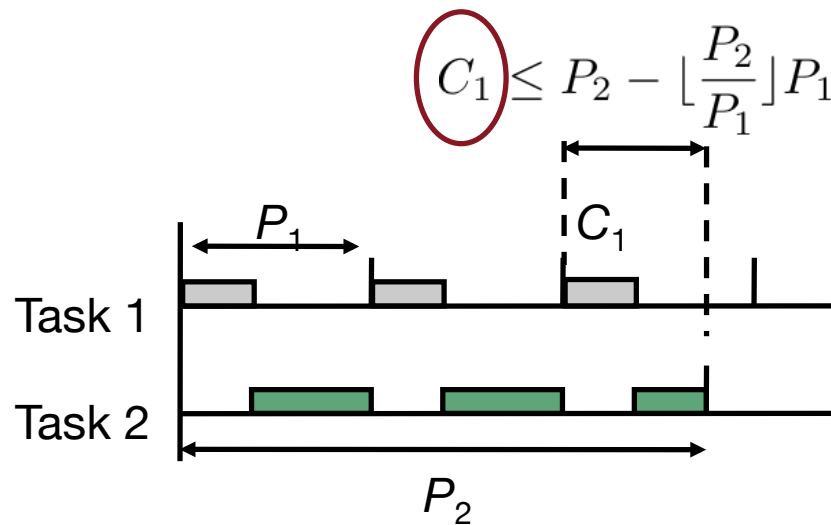
Case (b): $x > x_0$ such that $U(x)$ increases as x increases

Thus $U(x)$ is minimum when $x = x_0$

Find $U(x_0)$

Finding the utilization bound for RM scheduling

- Consider the simplest case with two tasks
 - There are two sub-cases...



Find some task set parameter x such that

Case (a): $x < x_0$ such that $U(x)$ decreases as x increases

Case (b): $x > x_0$ such that $U(x)$ increases as x increases

Thus $U(x)$ is minimum when $x = x_0$

Find $U(x_0)$

Finding the utilization bound for RM scheduling

Find some task set parameter x such that

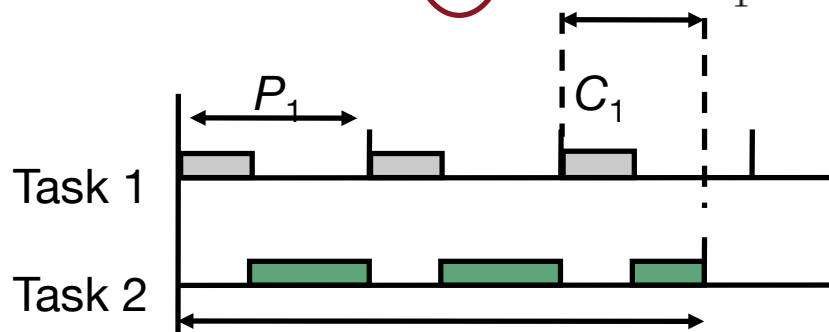
Case (a): $x < x_0$ such that $U(x)$ decreases as x increases

Case (b): $x > x_0$ such that $U(x)$ increases as x increases

Thus $U(x)$ is minimum when $x = x_0$

Find $U(x_0)$

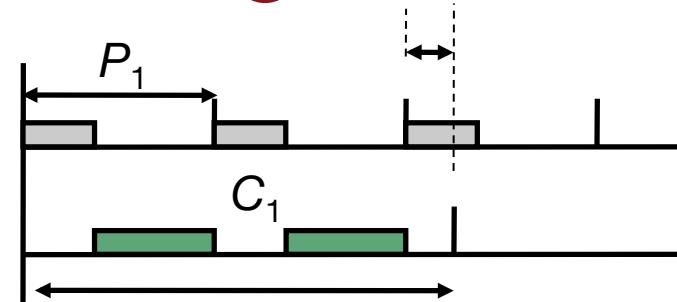
$$C_1 \leq P_2 - \lfloor \frac{P_2}{P_1} \rfloor P_1$$



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

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

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



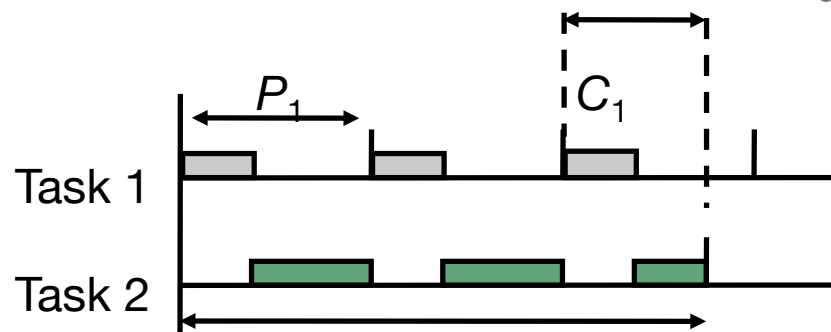
$$C_2 = (P_1 - C_1) \lfloor \frac{P_2}{P_1} \rfloor$$

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

Finding the utilization bound for RM scheduling

The minimum utilization case

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



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

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

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

To minimize U , we must have

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

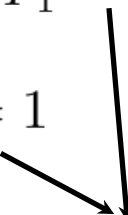
Why?

Finding the utilization bound for RM scheduling

The minimum utilization case

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

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


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

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

Finally, $U = 0.83$

Note that $C_1 = P_2 - P_1$ and $C_2 = 2P_1 - P_2$

Generalization for n tasks

$$\left. \begin{array}{lcl} C_1 & = & P_2 - P_1 \\ C_2 & = & P_3 - P_2 \\ C_3 & = & P_4 - P_3 \\ \dots & & \end{array} \right\} U = \frac{C_1}{P_1} + \frac{C_2}{P_2} + \frac{C_3}{P_3} + \dots$$

Generalization for n tasks

$$\left. \begin{array}{lcl} C_1 & = & P_2 - P_1 \\ C_2 & = & P_3 - P_2 \\ C_3 & = & P_4 - P_3 \\ \dots & & \end{array} \right\} U = \frac{C_1}{P_1} + \frac{C_2}{P_2} + \frac{C_3}{P_3} + \dots$$

$$\frac{dU}{d(P_2/P_1)} = 0; \frac{dU}{d(P_3/P_2)} = 0; \frac{dU}{d(P_4/P_3)} = 0; \dots$$

Generalization for n tasks

$$\left. \begin{array}{l} C_1 = P_2 - P_1 \\ C_2 = P_3 - P_2 \\ C_3 = P_4 - P_3 \\ \dots \end{array} \right\} U = \frac{C_1}{P_1} + \frac{C_2}{P_2} + \frac{C_3}{P_3} + \dots$$

$$\frac{dU}{d(P_2/P_1)} = 0; \frac{dU}{d(P_3/P_2)} = 0; \frac{dU}{d(P_4/P_3)} = 0; \dots$$

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

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?