Real-Time Scheduling

Introduction to Real-Time

Review

- Main vocabulary
  - 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%

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)

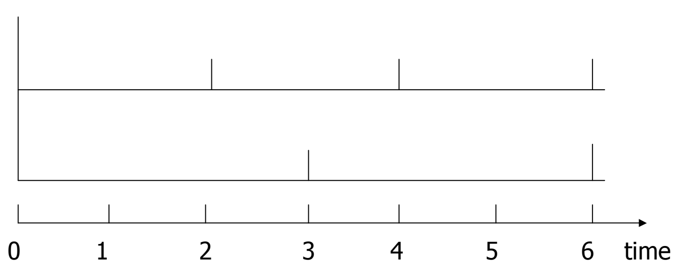
Utilization Bounds

- 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

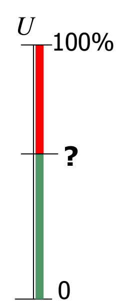
Example (Rate-Monotonic Scheduling)

Task 1
P_1=2
C_1=1

Task 2
P_2=3
C_2=1.01



$$U = \frac{C_1}{P_1} + \frac{C_2}{P_2} = \frac{1}{2} + \frac{1.01}{3} \approx 83.3\%$$

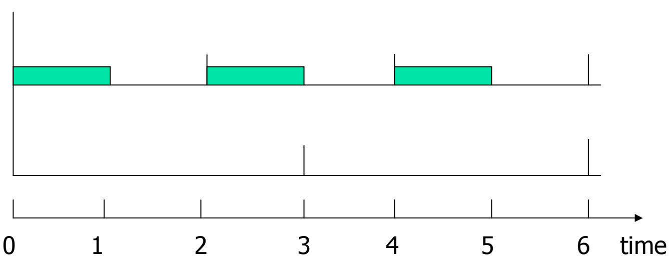


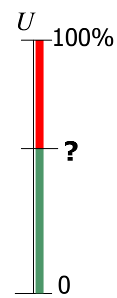
- Question: is there a threshold U_{bound} such that
  - When $U < U_{bound}$ deadlines are met
  - When $U > U_{bound}$ deadlines are missed

Example (Rate-Monotonic Scheduling)

Task 1
P_1=2
C_1=1

Task 2
P_2=3
C_2=1.01





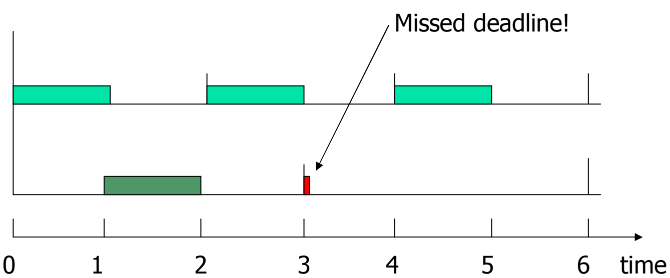
$$U = \frac{C_1}{P_1} + \frac{C_2}{P_2} = \frac{1}{2} + \frac{1.01}{3} \approx 83.3\%$$

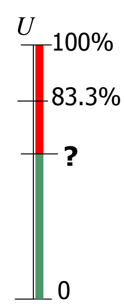
- Question: is there a threshold U_{bound} such that
  - When $U < U_{bound}$ deadlines are met
  - When $U > U_{bound}$ deadlines are missed

Example (Rate-Monotonic Scheduling)

Task 1
P_1=2
C_1=1

Task 2
P_2=3
C_2=1.01





$$U = \frac{C_1}{P_1} + \frac{C_2}{P_2} = \frac{1}{2} + \frac{1.01}{3} \approx 83.3\%$$

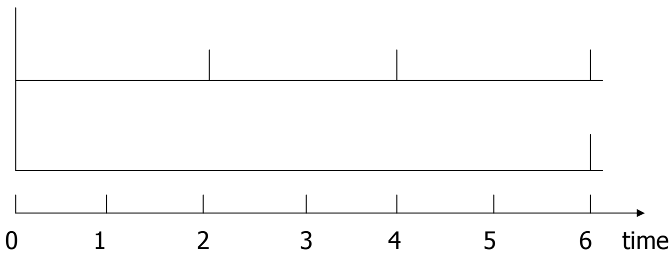
Unschedulable

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

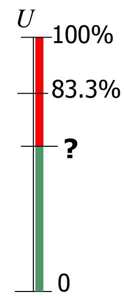
Another Example (Rate-Monotonic Scheduling)

Task 1
P_1=2
C_1=1

Task 2
P_2=6
C_2=2.4



$$U = \frac{C_1}{P_1} + \frac{C_2}{P_2} = \frac{1}{2} + \frac{2.4}{6} = 90\%$$

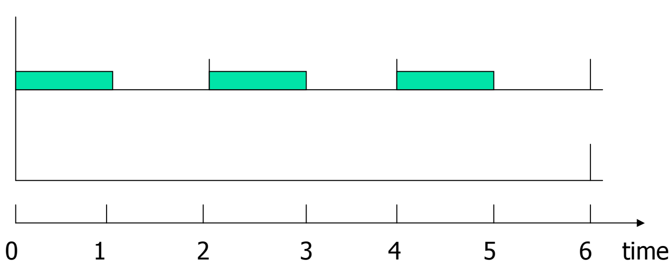


- Question: is there a threshold U_{bound} such that
  - When $U < U_{bound}$ deadlines are met
  - When $U > U_{bound}$ deadlines are missed

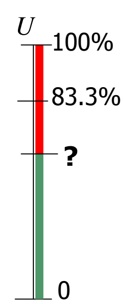
Another Example (Rate-Monotonic Scheduling)

Task 1
P_1=2
C_1=1

Task 2
P_2=6
C_2=2.4



$$U = \frac{C_1}{P_1} + \frac{C_2}{P_2} = \frac{1}{2} + \frac{2.4}{6} = 90\%$$

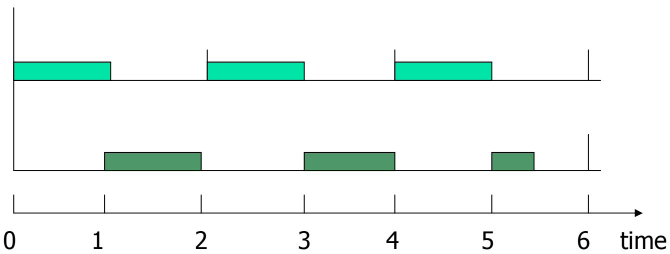


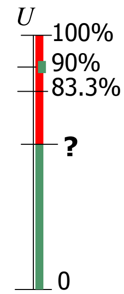
- Question: is there a threshold U_{bound} such that
  - When $U < U_{bound}$ deadlines are met
  - When $U > U_{bound}$ deadlines are missed

Another Example (Rate-Monotonic Scheduling)

Task 1
P_1=2
C_1=1

Task 2
P_2=6
C_2=2.4





$$U = \frac{C_1}{P_1} + \frac{C_2}{P_2} = \frac{1}{2} + \frac{2.4}{6} = 90\%$$

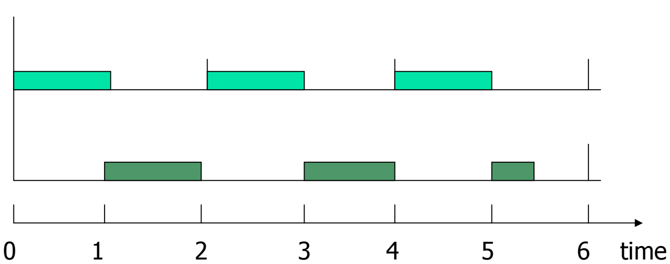
Schedulable!

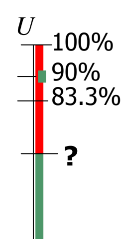
- Question: is there a threshold U_{bound} such that
  - When $U < U_{bound}$ deadlines are met
  - When $U > U_{bound}$ deadlines are missed

Another Example (Rate-Monotonic Scheduling)

Task 1
P_1=2
C_1=1

Task 2
P_2=6
C_2=2.4



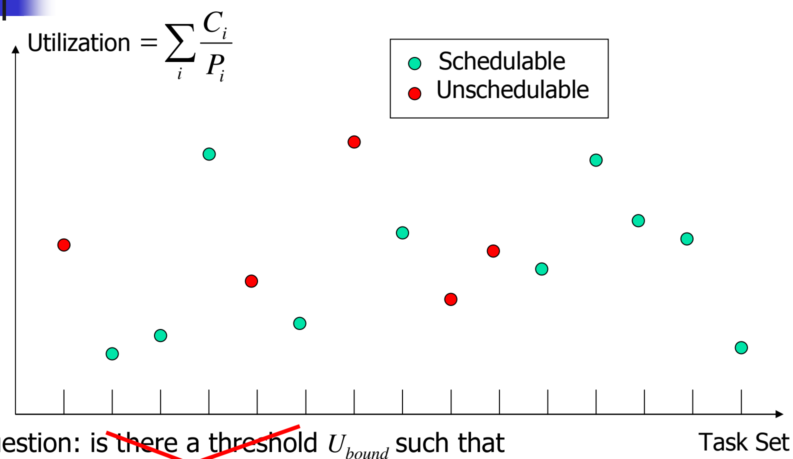


$$\frac{C_1}{} \quad \frac{C_2}{} \quad \frac{1}{} + \frac{2.4}{} = 90\%$$

Schedulability depends on task set!
No clean utilization threshold between schedulable and unschedulable task sets!

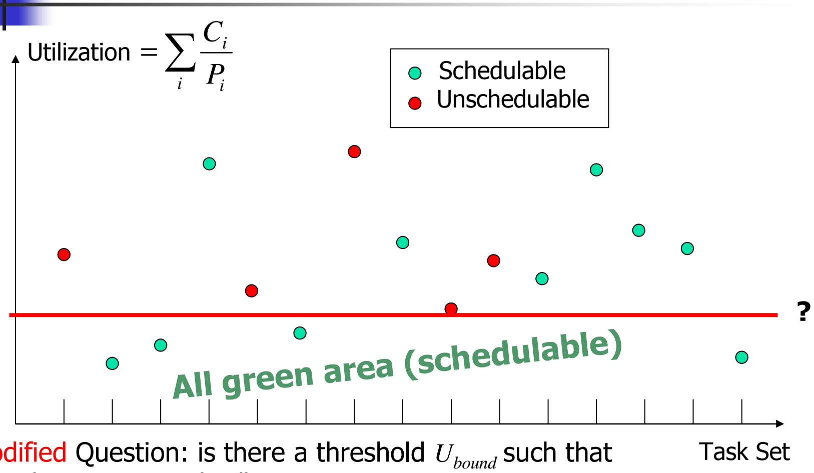
- Question: is there a th...
  - When $U < U_{bound}$ deadlines are met
  - When $U > U_{bound}$ deadlines are missed

A Conceptual View of Schedulability



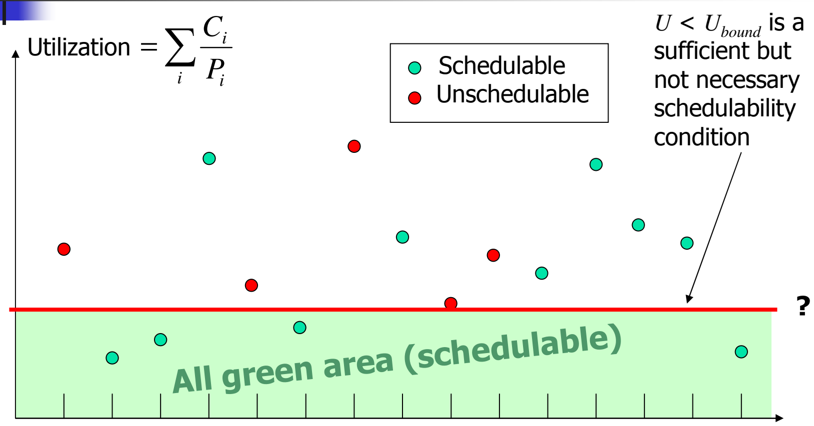
- Question: is there a threshold U_{bound} such that
  - When $U < U_{bound}$ deadlines are met
  - When $U > U_{bound}$ deadlines are missed

A Conceptual View of Schedulability



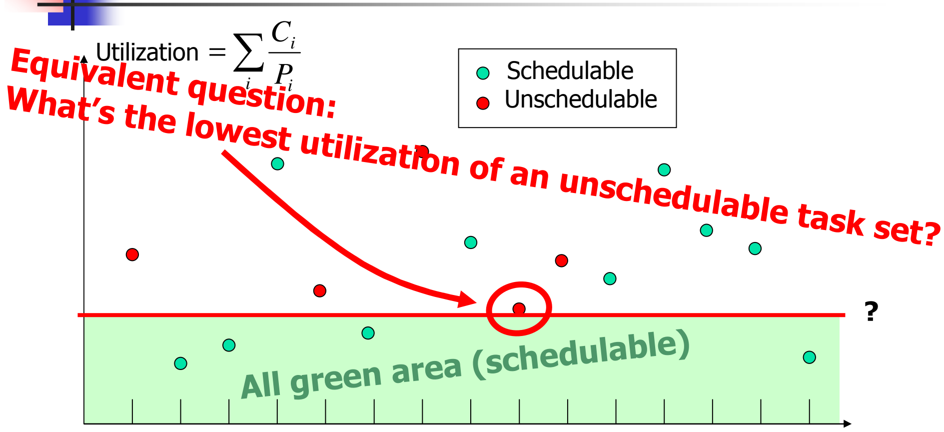
- Modified Question: is there a threshold U_{bound} such that
  - When $U < U_{bound}$ deadlines are met
  - When $U > U_{bound}$ deadlines **may or may not be** missed

A Conceptual View of Schedulability



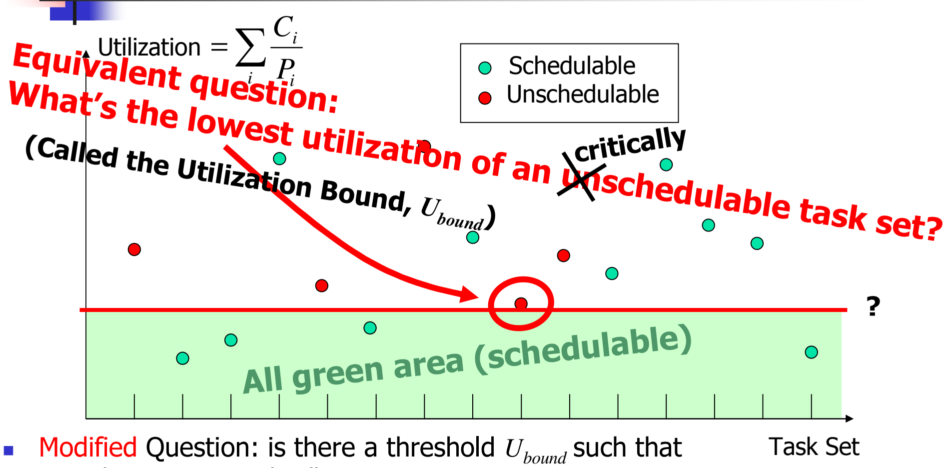
- Modified Question: is there a threshold U_{bound} such that
  - When $U < U_{bound}$ deadlines are met
  - When $U > U_{bound}$ deadlines **may or may not be** missed

A Conceptual View of Schedulability



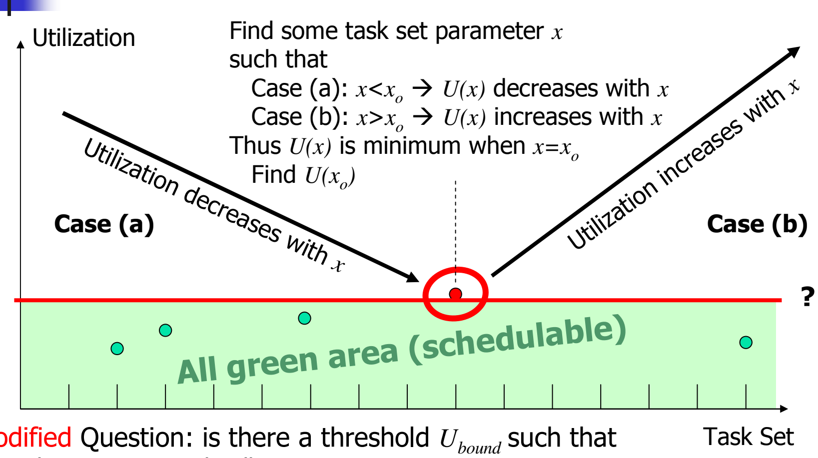
- Modified Question: is there a threshold U_{bound} such that
  - When $U < U_{bound}$ deadlines are met
  - When $U > U_{bound}$ deadlines **may or may not be** missed

A Conceptual View of 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

Solution Approach: Look at Critically-Schedulable Task Sets



- 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

Deriving the Utilization Bound for Rate Monotonic Scheduling

- Consider a simple case: 2 tasks

Find some task set parameter x such that

Case (a): $x<x_o \rightarrow U(x)$ decreases with x

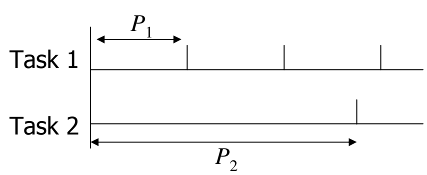
Case (b): $x>x_o \rightarrow U(x)$ increases with x

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

Find $U(x_o)$

Deriving the Utilization Bound for Rate Monotonic Scheduling

- Consider a simple case: 2 tasks



Find some task set parameter x such that

Case (a): $x<x_o \rightarrow U(x)$ decreases with x

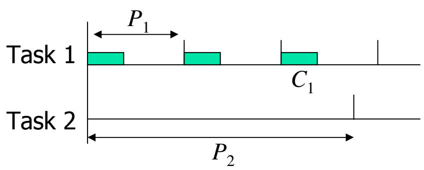
Case (b): $x>x_o \rightarrow U(x)$ increases with x

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

Find $U(x_o)$

Deriving the Utilization Bound for Rate Monotonic Scheduling

- Consider a simple case: 2 tasks



Find some task set parameter x such that

Case (a): $x<x_o$ → $U(x)$ decreases with x

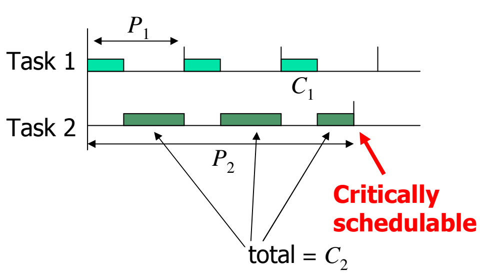
Case (b): $x>x_o$ → $U(x)$ increases with x

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

Find $U(x_o)$

Deriving the Utilization Bound for Rate Monotonic Scheduling

- Consider a simple case: 2 tasks



Find some task set parameter x such that

Case (a): $x<x_o$ → $U(x)$ decreases with x

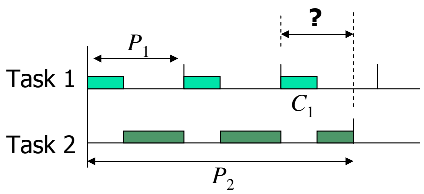
Case (b): $x>x_o$ → $U(x)$ increases with x

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

Find $U(x_o)$

Deriving the Utilization Bound for Rate Monotonic Scheduling

- Consider a simple case: 2 tasks



Find some task set parameter x such that

Case (a): $x<x_o$ → $U(x)$ decreases with x

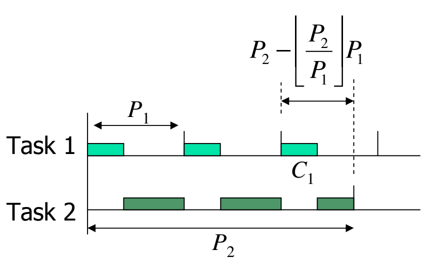
Case (b): $x>x_o$ → $U(x)$ increases with x

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

Find $U(x_o)$

Deriving the Utilization Bound for Rate Monotonic Scheduling

- Consider a simple case: 2 tasks



Find some task set parameter x such that

Case (a): $x<x_o$ → $U(x)$ decreases with x

Case (b): $x>x_o$ → $U(x)$ increases with x

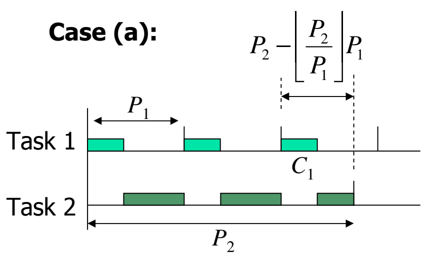
Thus $U(x)$ is minimum when $x=x_o$

Find $U(x_o)$

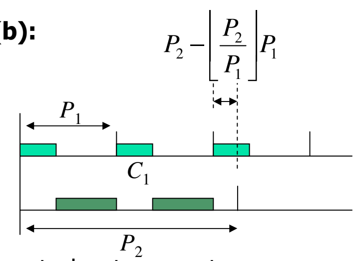
Deriving the Utilization Bound for Rate Monotonic Scheduling

- Consider these two sub-cases:

Case (a):



Case (b):

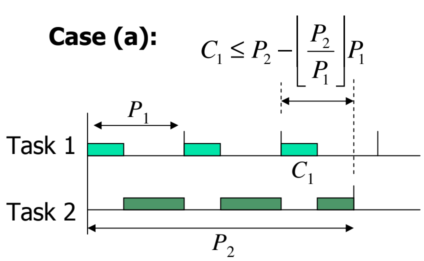


Find some task set parameter x such that
Case (a): $x<x_o$ → $U(x)$ decreases with x
Case (b): $x>x_o$ → $U(x)$ increases with x
Thus $U(x)$ is minimum when $x=x_o$
Find $U(x_o)$

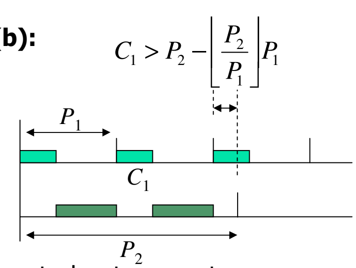
Deriving the Utilization Bound for Rate Monotonic Scheduling

- Consider these two sub-cases:

Case (a):



Case (b):

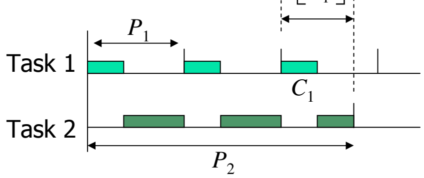


Find some task set parameter x such that
Case (a): $x<x_o$ → $U(x)$ decreases with x
Case (b): $x>x_o$ → $U(x)$ increases with x
Thus $U(x)$ is minimum when $x=x_o$
Find $U(x_o)$

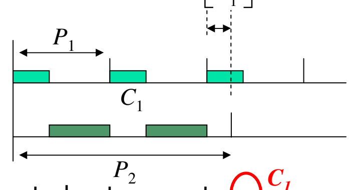
Deriving the Utilization Bound for Rate Monotonic Scheduling

- Consider these two sub-cases:

Case (a): $C_1 \le P_2 - \left\lfloor \frac{P_2}{P_1} \right\rfloor P_1$



Case (b): $C_1 > P_2 - \left\lfloor \frac{P_2}{P_1} \right\rfloor P_1$



Find some task set parameter x (C_1) such that

Case (a): $x<x_o$ → $U(x)$ decreases with x

Case (b): $x>x_o$ → $U(x)$ increases with x

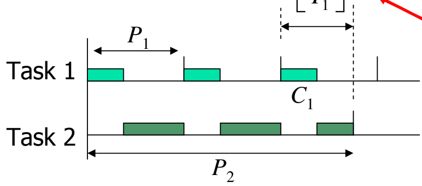
Thus $U(x)$ is minimum when $x=x_o$

Find $U(x_o)$

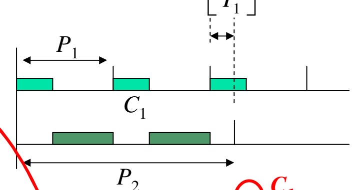
Deriving the Utilization Bound for Rate Monotonic Scheduling

- Consider these two sub-cases:

Case (a): $C_1 \le P_2 - \left\lfloor \frac{P_2}{P_1} \right\rfloor P_1$



Case (b): $C_1 > P_2 - \left\lfloor \frac{P_2}{P_1} \right\rfloor P_1$



Find some task set parameter x (C_1) such that

Case (a): $x<x_o$ → $U(x)$ decreases with x

Case (b): $x>x_o$ → $U(x)$ increases with x

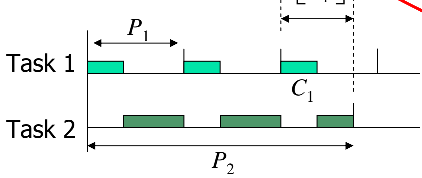
Thus $U(x)$ is minimum when $x=x_o$

Find $U(x_o)$

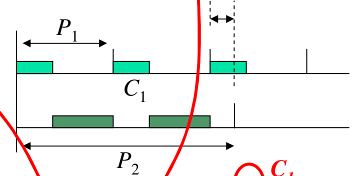
Deriving the Utilization Bound for Rate Monotonic Scheduling

- Consider these two sub-cases:

Case (a): $C_1 \le P_2 - \left\lfloor \frac{P_2}{P_1} \right\rfloor P_1$



Case (b): $C_1 > P_2 - \left\lfloor \frac{P_2}{P_1} \right\rfloor P_1$



Find some task set parameter x (C_1)
such that

Case (a): $x < x_o$ → $U(x)$ decreases with x

Case (b): $x > x_o$ → $U(x)$ increases with x

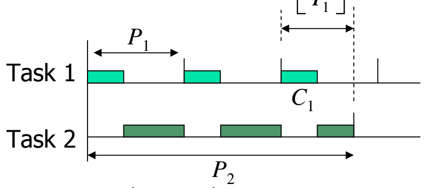
Thus $U(x)$ is minimum when $x = x_o$

Find $U(x_o)$

Deriving the Utilization Bound for Rate Monotonic Scheduling

- Consider these two sub-cases:

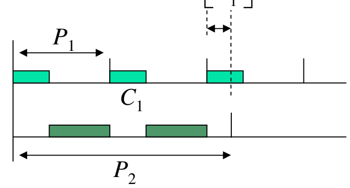
Case (a): $C_1 \le P_2 - \left\lfloor \frac{P_2}{P_1} \right\rfloor P_1$



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

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

Case (b): $C_1 > P_2 - \left\lfloor \frac{P_2}{P_1} \right\rfloor P_1$



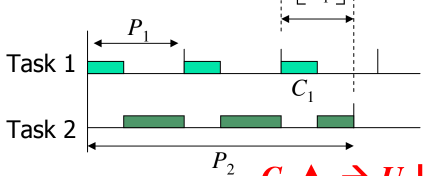
$$C_2 = (P_1 - C_1)\left\lfloor \frac{P_2}{P_1} \right\rfloor$$

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

Deriving the Utilization Bound for Rate Monotonic Scheduling

- Consider these two sub-cases:

Case (a): $C_1 \le P_2 - \left\lfloor \frac{P_2}{P_1} \right\rfloor P_1$

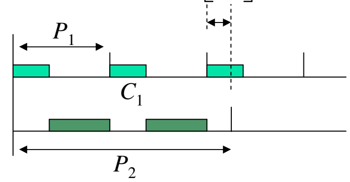


$C_2 = P_2 - C_1\left(\left\lfloor \frac{P_2}{P_1} \right\rfloor + 1\right)$

$C_1 \uparrow \rightarrow U \downarrow$

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

Case (b): $C_1 > P_2 - \left\lfloor \frac{P_2}{P_1} \right\rfloor P_1$



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

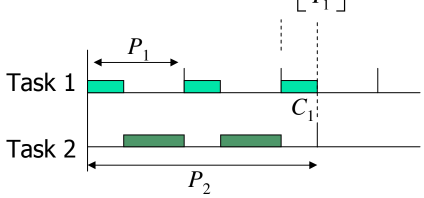
$C_1 \downarrow \rightarrow U \downarrow$

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

Deriving the Utilization Bound for Rate Monotonic Scheduling

- The minimum utilization case:

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



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

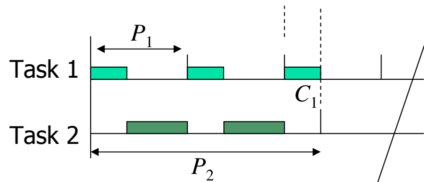
Deriving the Utilization Bound for Rate Monotonic Scheduling

- The minimum utilization case:

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

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

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



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

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

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

$$\Rightarrow U \approx 0.83$$

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

Note that $C_1 = P_2 - P_1$

Generalizing to N Tasks

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

...

Generalizing to N Tasks

$$\left.\begin{aligned} C_1 &= P_2 - P_1 \\ C_2 &= P_3 - P_2 \\ C_3 &= P_3 - P_2 \end{aligned}\right\} \quad U = \frac{C_1}{P_1} + \frac{C_2}{P_2} + \frac{C_3}{P_3} + \ldots$$

...

$$\frac{dU}{d\left(P_2/P_1\right)} = 0 \qquad \frac{dU}{d\left(P_3/P_2\right)} = 0 \qquad \frac{dU}{d\left(P_4/P_3\right)} = 0 \qquad \ldots$$

Generalizing to N Tasks

$$\left.\begin{aligned} C_1 &= P_2 - P_1 \\ C_2 &= P_3 - P_2 \\ C_3 &= P_3 - P_2 \end{aligned}\right\} \quad U = \frac{C_1}{P_1} + \frac{C_2}{P_2} + \frac{C_3}{P_3} + \ldots$$

...

$$\frac{dU}{d\left(P_2/P_1\right)} = 0 \qquad \frac{dU}{d\left(P_3/P_2\right)} = 0 \qquad \frac{dU}{d\left(P_4/P_3\right)} = 0 \qquad \ldots$$

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

Generalizing to N Tasks

$$\left.\begin{aligned} C_1 &= P_2 - P_1 \\ C_2 &= P_3 - P_2 \\ C_3 &= P_3 - P_2 \end{aligned}\right\} \qquad U = \frac{C_1}{P_1} + \frac{C_2}{P_2} + \frac{C_3}{P_3} + \ldots$$

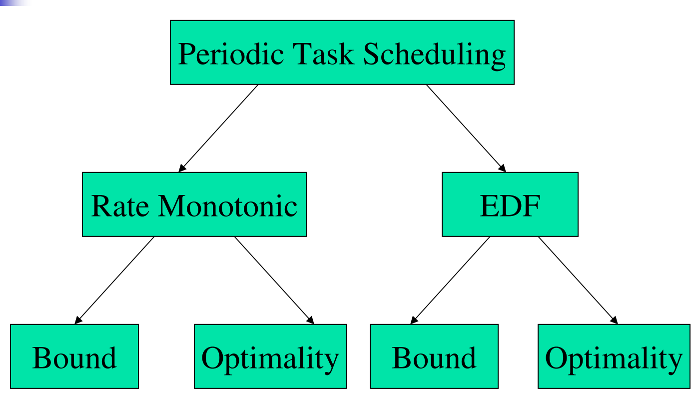
$\ldots$

$$\frac{dU}{d\left(P_2/P_1\right)} = 0 \qquad \frac{dU}{d\left(P_3/P_2\right)} = 0 \qquad \frac{dU}{d\left(P_4/P_3\right)} = 0 \qquad \ldots$$

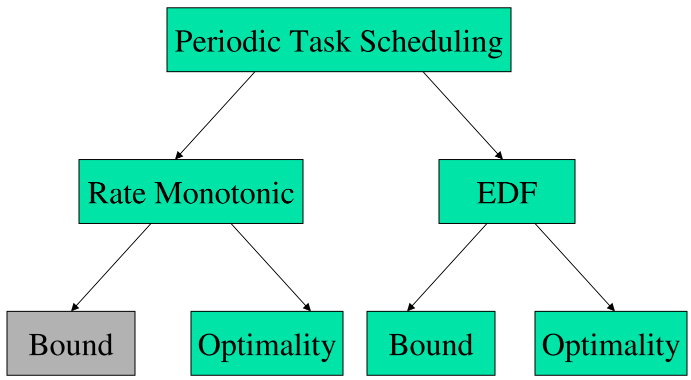
$$\Rightarrow \frac{P_{i+1}}{P_i} = 2^{1/n} \qquad \Rightarrow U = n\left(2^{1/n} - 1\right)$$

$$n \to \infty \quad U \to \ln 2$$

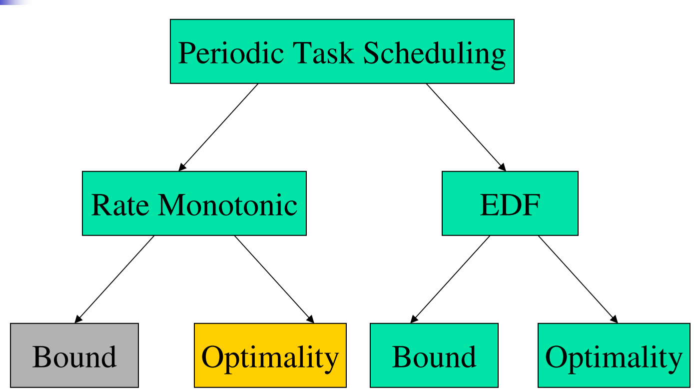
Periodic Tasks



Periodic Tasks



Coming Up



Rate Monotonic Continued

- Rate monotonic scheduling is the optimal fixed-priority scheduling policy for periodic tasks.
  - Optimality (Trial #1):

Rate Monotonic Continued

- Rate monotonic scheduling is the optimal fixed-priority scheduling policy for periodic tasks.

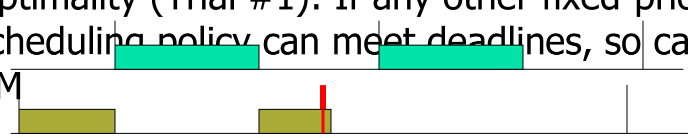
  - Optimality (Trial #1): If any other fixed-priority scheduling policy can meet deadlines, so can RM.

Rate Monotonic Continued

- Rate monotonic scheduling is the optimal fixed-priority scheduling policy for periodic tasks.
  - Optimality (Trial #1): If any other fixed-priority scheduling policy can meet deadlines, so can RM

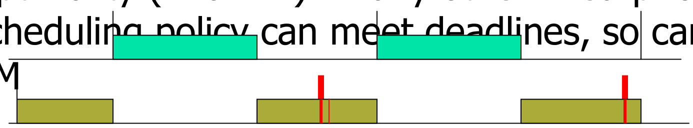
Rate Monotonic Continued

- Rate monotonic scheduling is the optimal fixed-priority scheduling policy for periodic tasks.
  - Optimality (Trial #1): If any other fixed-priority scheduling policy can meet deadlines, so can RM



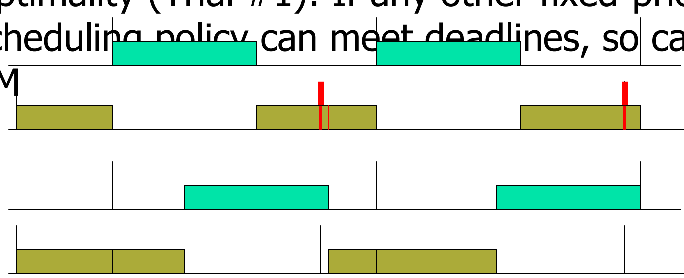
Rate Monotonic Continued

- Rate monotonic scheduling is the optimal fixed-priority scheduling policy for periodic tasks.
  - Optimality (Trial #1): If any other fixed-priority scheduling policy can meet deadlines, so can RM



Rate Monotonic Continued

- Rate monotonic scheduling is the optimal fixed-priority scheduling policy for periodic tasks.
  - Optimality (Trial #1): If any other fixed-priority scheduling policy can meet deadlines, so can RM



Rate Monotonic Continued

- Rate monotonic scheduling is the optimal fixed-priority scheduling policy for periodic tasks.
  - Optimality (Trial #2): If any other fixed-priority scheduling policy can meet deadlines *in the worst case scenario*, so can RM.
- How to prove it?

Rate Monotonic Continued

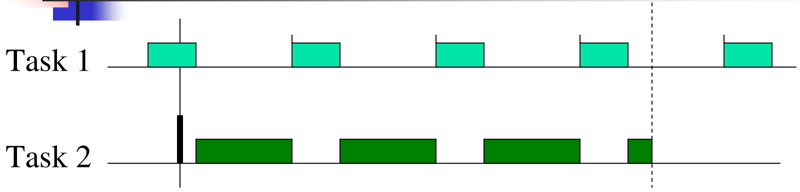
- Rate monotonic scheduling is the optimal fixed-priority scheduling policy for periodic tasks.
  - Optimality (Trial #2): If any other fixed-priority scheduling policy can meet deadlines *in the worst case scenario*, so can RM.
- How to prove it?
  - Consider the worst case scenario
  - If someone else can schedule then RM can

The Worst-Case Scenario

- Q: When does a periodic task, *T*, experience the maximum delay?
- A: When it arrives together with all the higher-priority tasks (critical instance)

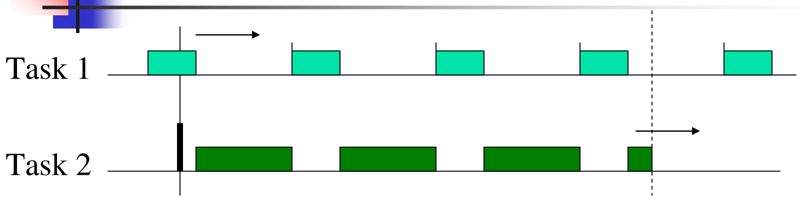
- Idea of Proof
  - If some higher-priority task does not arrive together with *T*, aligning the arrival times can only increase the completion time of *T*

Proof (Case 1)



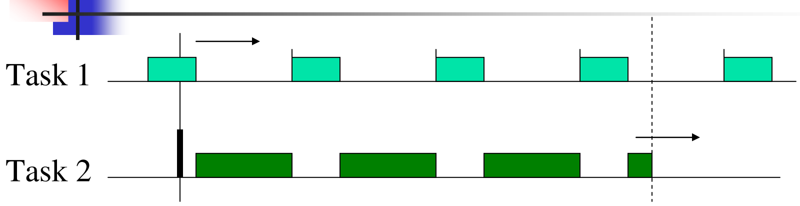
Case 1: higher priority task 1 is running when task 2 arrives

Proof

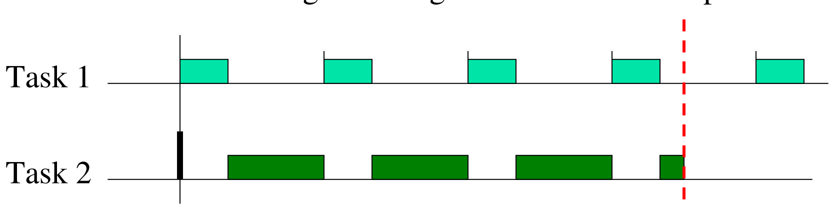


Case 1: higher priority task 1 is running when task 2 arrives
→ shifting task 1 right will increase completion time of 2

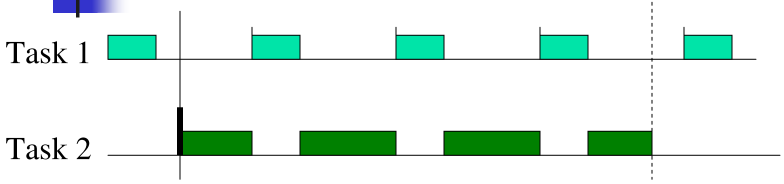
Proof



Case 1: higher priority task 1 is running when task 2 arrives
→ shifting task 1 right will increase completion time of 2

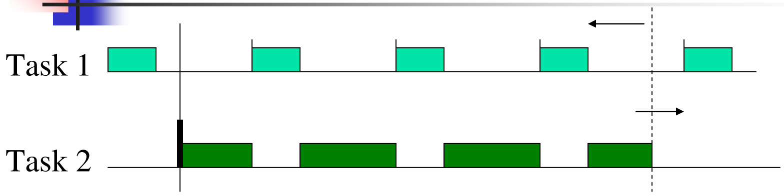


Proof (Case 2)



Case 2: processor is idle when task 2 arrives

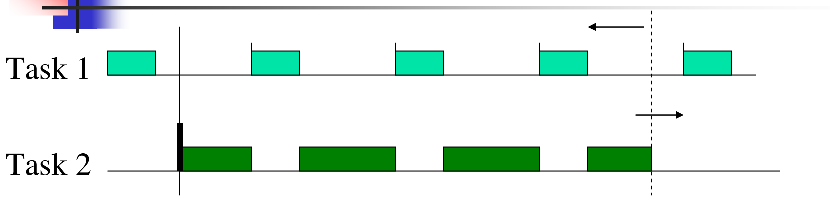
Proof (Case 2)



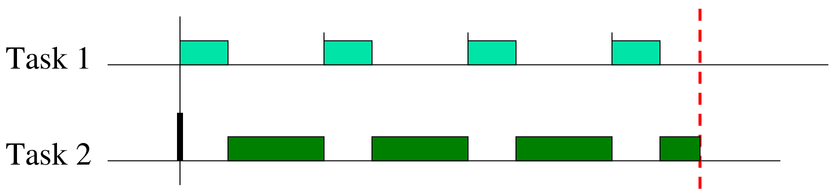
Case 2: processor is idle when task 2 arrives

→ shifting task 1 left cannot decrease completion time of 2

Proof (Case 2)

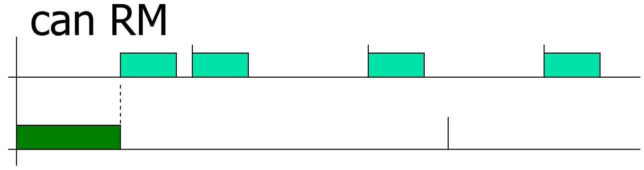


Case 2: processor is idle when task 2 arrives
→ shifting task 1 left cannot decrease completion time of 2



Optimality of Rate Monotonic

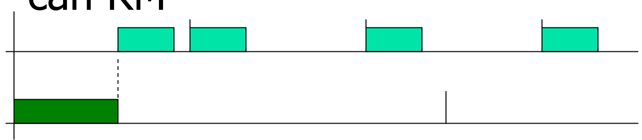
- If any other policy can meet deadlines so can RM



Policy X meets deadlines?

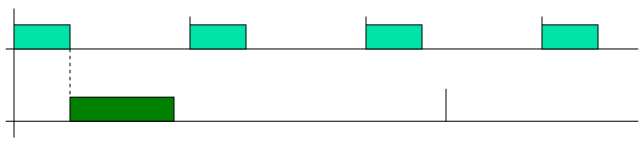
Optimality of Rate Monotonic

- If any other policy can meet deadlines so can RM

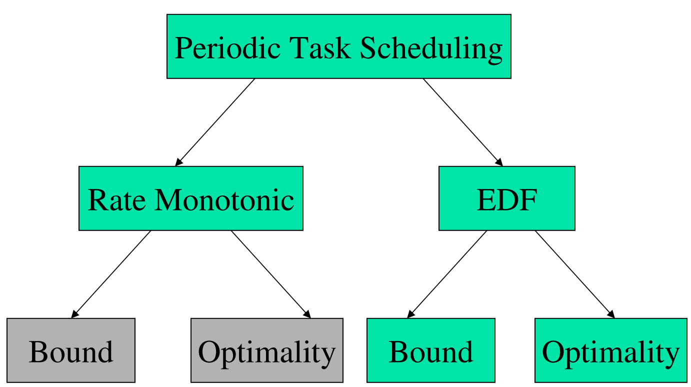


Policy X meets deadlines? *YES*

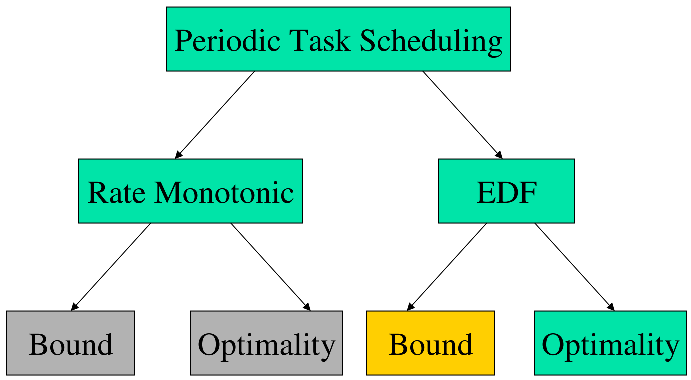
→ RM meets deadlines



Coming Up



Coming Up

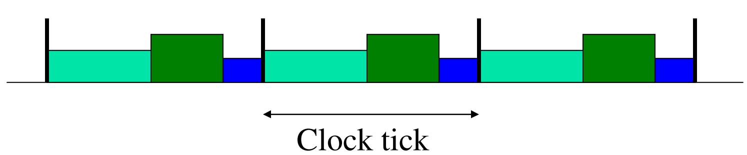


Utilization Bound of EDF

- Why is it 100%?
- Consider a task set where:

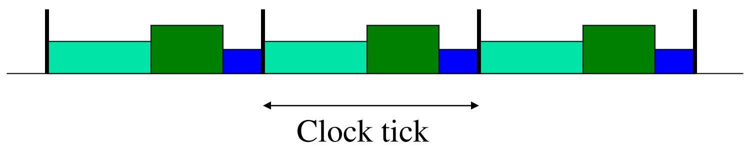
$$\sum_i \frac{C_i}{P_i} = 1$$

- Imagine a policy that reserves for each task *i* a fraction f_i of each clock tick, where $f_i = C_i/P_i$



Utilization Bound of EDF

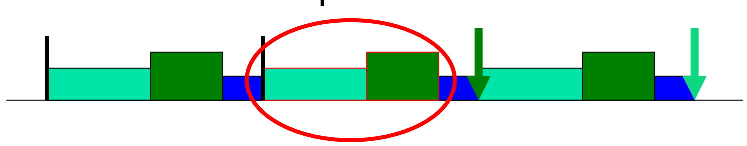
- Imagine a policy that reserves for each task i a fraction f_i of each time unit, where $f_i = C_i/P_i$



- This policy meets all deadlines, because within each period P_i it reserves for task i a total time
  - Time = $f_i P_i = (C_i / P_i) P_i = C_i$ (i.e., enough to finish)

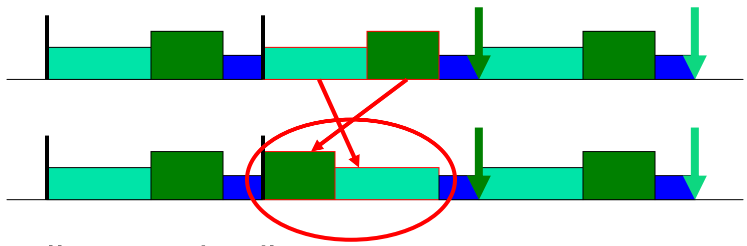
Utilization Bound of EDF

- Pick any two execution chunks that are not in EDF order and swap them



Utilization Bound of EDF

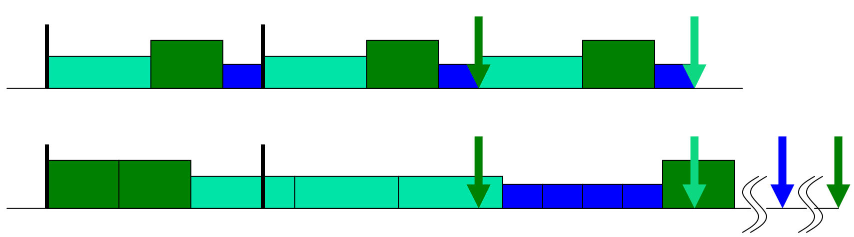
- Pick any two execution chunks that are not in EDF order and swap them



- Still meets deadlines!

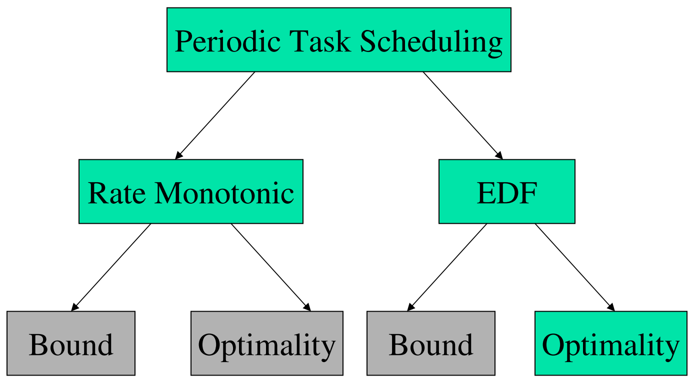
Utilization Bound of EDF

- Pick any two execution chunks that are not in EDF order and swap them

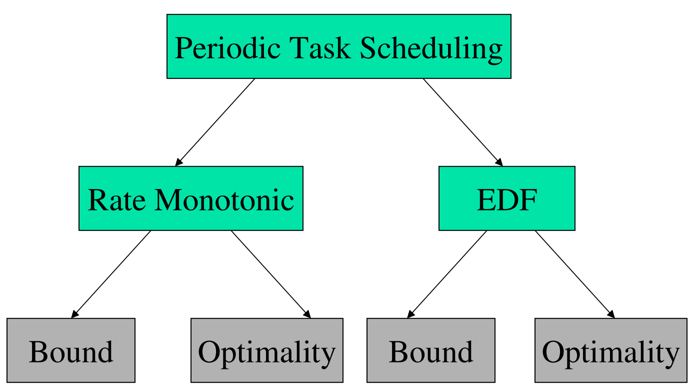


- Still meets deadlines!
- Repeat swap until all in EDF order
  → EDF meets deadlines

Periodic Tasks



Done



Exercise: Know Your Worst Case Scenario

- Consider a periodic system of two tasks
- Let $U_i = C_i / P_i$ (for $i = 1,2$)
- What is the maximum value of:

  $\Pi_i(1+U_i)$

  for a schedulable system?

Deriving the Utilization Bound for Rate Monotonic Scheduling

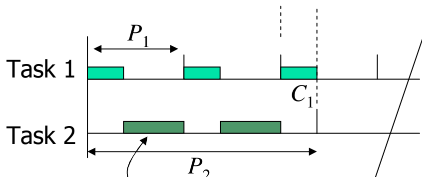
- The minimum utilization case:

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

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

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

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



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

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

Deriving the Utilization Bound for Rate Monotonic Scheduling

- The minimum utilization case:

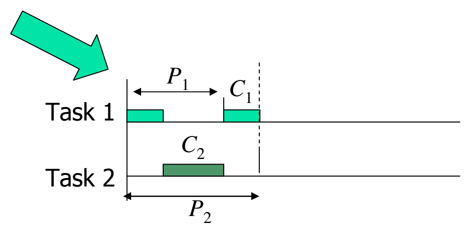
$$C_1 = P_1\left(\frac{P_2}{P_1} - \left\lfloor \frac{P_2}{P_1} \right\rfloor\right)$$

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

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

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

[redacted]



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

Deriving the Utilization Bound for Rate Monotonic Scheduling

- The minimum utilization case:

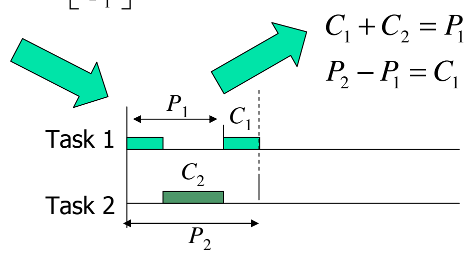
$$C_1 = P_1\left(\frac{P_2}{P_1} - \left\lfloor \frac{P_2}{P_1} \right\rfloor\right)$$

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

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

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

[redacted]



$$C_1 + C_2 = P_1$$

$$P_2 - P_1 = C_1$$

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

Solutions

Critically Schedulable

$$\begin{cases} C_1 = P_2 - P_1 \\ C_2 = P_1 - C_1 = 2P_1 - P_2 \end{cases}$$

Schedulable

Solutions

Critically Schedulable

$$\begin{cases} C_1 = P_2 - P_1 \\ C_2 = P_1 - C_1 = 2P_1 - P_2 \\ U_1 + 1 = \frac{C_1}{P_1} + 1 = \frac{C_1 + P_1}{P_1} = \frac{P_2}{P_1} \end{cases}$$

Schedulable

Solutions

Critically Schedulable

$$\begin{cases} C_1 = P_2 - P_1 \\ C_2 = P_1 - C_1 = 2P_1 - P_2 \\ U_1 + 1 = \dfrac{C_1}{P_1} + 1 = \dfrac{C_1 + P_1}{P_1} = \dfrac{P_2}{P_1} \\ U_2 + 1 = \dfrac{C_2}{P_2} + 1 = \dfrac{C_2 + P_2}{P_2} = \dfrac{2P_1}{P_2} \end{cases}$$

Schedulable

Solutions

Critically Schedulable

$$\begin{cases} C_1 = P_2 - P_1 \\ C_2 = P_1 - C_1 = 2P_1 - P_2 \\ U_1 + 1 = \dfrac{C_1}{P_1} + 1 = \dfrac{C_1 + P_1}{P_1} = \dfrac{P_2}{P_1} \\ U_2 + 1 = \dfrac{C_2}{P_2} + 1 = \dfrac{C_2 + P_2}{P_2} = \dfrac{2P_1}{P_2} \\ \prod_i (U_i + 1) = 2 \end{cases}$$

Schedulable $\prod_i (U_i + 1) \le 2$

The General Case

Critically Schedulable

$$\begin{cases} C_i = P_{i+1} - P_i \\ C_n = 2P_1 - P_n \end{cases}$$

Schedulable

The General Case

Critically Schedulable

$$\begin{cases} C_i = P_{i+1} - P_i \\ C_n = 2P_1 - P_n \\ U_i + 1 = \frac{C_i}{P_i} + 1 = \frac{C_i + P_i}{P_i} = \frac{P_{i+1}}{P_i} \end{cases}$$

Schedulable

The General Case

Critically Schedulable

$$\begin{cases} C_i = P_{i+1} - P_i \\ C_n = 2P_1 - P_n \\ U_i + 1 = \dfrac{C_i}{P_i} + 1 = \dfrac{C_i + P_i}{P_i} = \dfrac{P_{i+1}}{P_i} \\ U_n + 1 = \dfrac{C_n}{P_n} + 1 = \dfrac{C_n + P_n}{P_n} = \dfrac{2P_1}{P_n} \end{cases}$$

Schedulable

The General Case

Critically Schedulable

$$\begin{cases} C_i = P_{i+1} - P_i \\ C_n = 2P_1 - P_n \\ U_i + 1 = \dfrac{C_i}{P_i} + 1 = \dfrac{C_i + P_i}{P_i} = \dfrac{P_{i+1}}{P_i} \\ U_n + 1 = \dfrac{C_n}{P_n} + 1 = \dfrac{C_n + P_n}{P_n} = \dfrac{2P_1}{P_n} \\ \prod_i (U_i + 1) = \dfrac{P_2}{P_1}\dfrac{P_3}{P_2}\cdots\dfrac{P_n}{P_{n-1}}\dfrac{2P_1}{P_n} = 2 \end{cases}$$

Schedulable

$$\prod_i (U_i + 1) \le 2$$

The Hyperbolic Bound for Rate Monotonic Scheduling

- A set of periodic tasks is schedulable if:

$$\prod_i (U_i + 1) \le 2$$

The Hyperbolic Bound for Rate Monotonic Scheduling

- A set of periodic tasks is schedulable if:

$$\prod_i (U_i + 1) \le 2$$

- It's a better bound than $\sum_i U_i \le n\left(2^{1/n} - 1\right)$
  - Example:
    - A system of two tasks with U_1=0.8, U_2=0.1

The Hyperbolic Bound for Rate Monotonic Scheduling

- A set of periodic tasks is schedulable if:

$$\prod_i (U_i + 1) \leq 2$$

- It's a better bound!
  - Example:
    - A system of two tasks with U_1=0.8, U_2=0.1
    - Liu and Layland bound: U_1+U_2 = 0.9 > 0.83

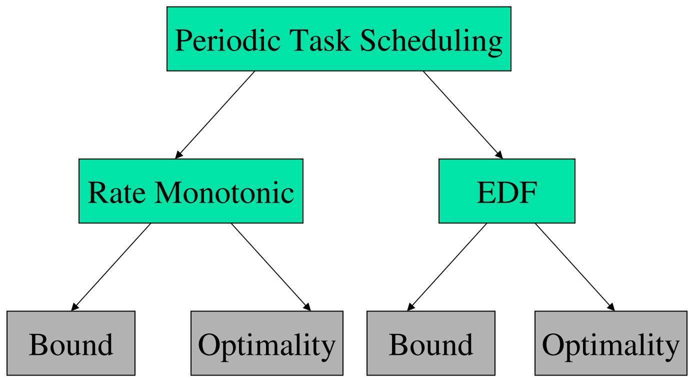
The Hyperbolic Bound for Rate Monotonic Scheduling

- A set of periodic tasks is schedulable if:

$$\prod_i (U_i + 1) \leq 2$$

- It's a better bound!
  - Example:
    - A system of two tasks with U_1=0.8, U_2=0.1
    - Liu and Layland bound: U_1+U_2 = 0.9 > 0.83
    - Hyperbolic bound (U_1+1)(U_2+1) =1.8 x 1.1=1.98 < 2

Scheduling Taxonomy



Scheduling Taxonomy