Periodic Task 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)

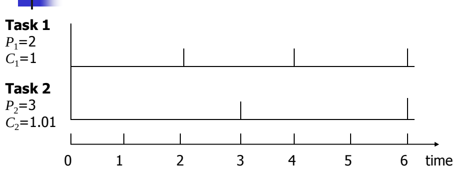
Schedulability Analysis of Periodic Tasks

- 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)

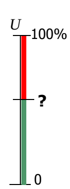
Utilization Bounds

- 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)

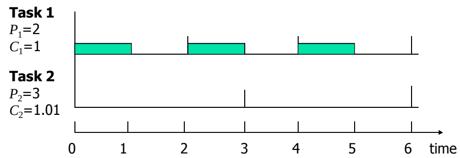


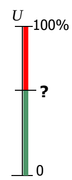
$$U = \frac{C_1}{P_1} + \frac{C_2}{P_2} = \frac{1}{2} + \frac{1.01}{3} \approx 83.3\%$$



- 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)

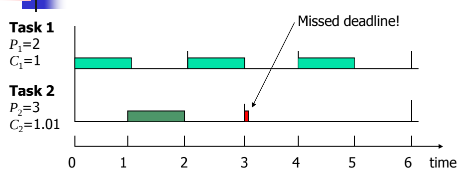


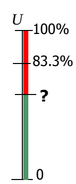


$$U = \frac{C_1}{P_1} + \frac{C_2}{P_2} = \frac{1}{2} + \frac{1.01}{3} \approx 83.3\%$$

- 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)



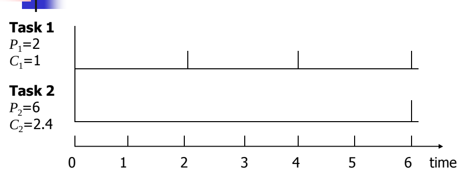


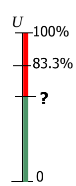
$$U = \frac{C_1}{P_1} + \frac{C_2}{P_2} = \frac{1}{2} + \frac{1.01}{3} \approx 83.3\%$$

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)

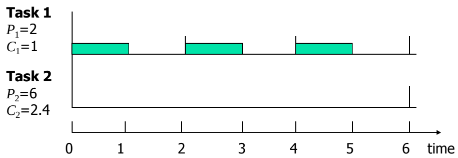


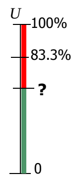


$$U = \frac{C_1}{P_1} + \frac{C_2}{P_2} = \frac{1}{2} + \frac{2.4}{6} = 90\%$$

- 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)

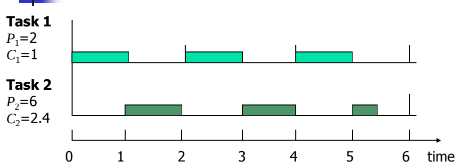


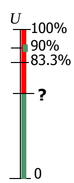


$$U = \frac{C_1}{P_1} + \frac{C_2}{P_2} = \frac{1}{2} + \frac{2.4}{6} = 90\%$$

- 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)



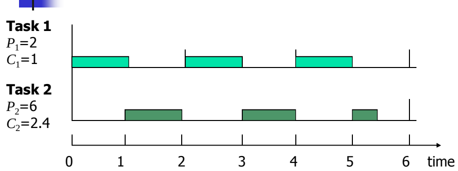


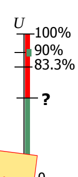
$$U = \frac{C_1}{P_1} + \frac{C_2}{P_2} = \frac{1}{2} + \frac{2.4}{6} = 90\%$$

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)





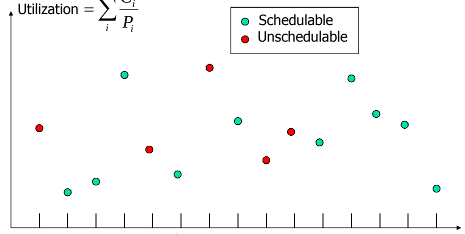
$$U = \frac{C_1}{P_1} + \frac{C_2}{P_2} = \frac{1}{2} + \frac{2.4}{6} = 90\%$$

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

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

A Conceptual View of Schedulability

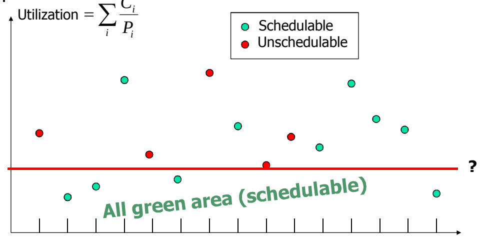
Utilization $= \sum_i \frac{C_i}{P_i}$



- 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

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

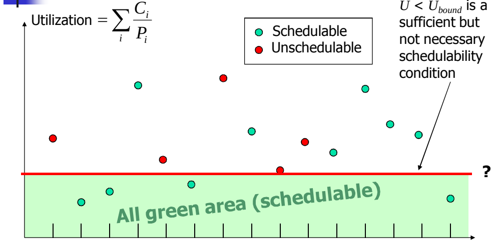


- Modified Question: is there a threshold U_{bound} such that
  - When $U < U_{bound}$ deadlines are met
  - When $U > U_{bound}$ deadlines **may or may not be** missed

A Conceptual View of Schedulability

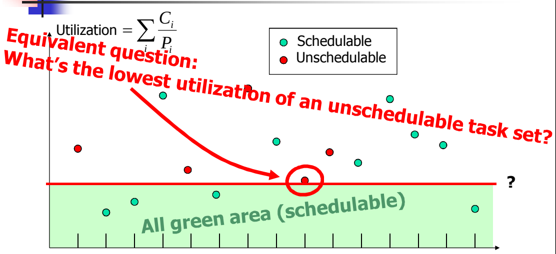
Utilization $= \sum_i \frac{C_i}{P_i}$

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



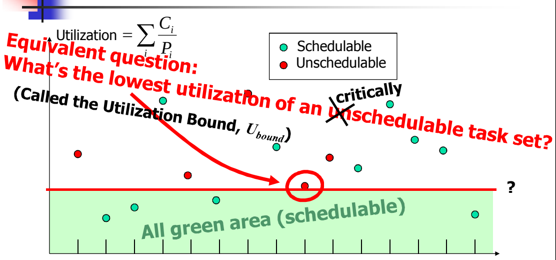
- Modified Question: is there a threshold U_{bound} such that
  - When $U < U_{bound}$ deadlines are met
  - When $U > U_{bound}$ deadlines **may or may not be** missed

A Conceptual View of Schedulability



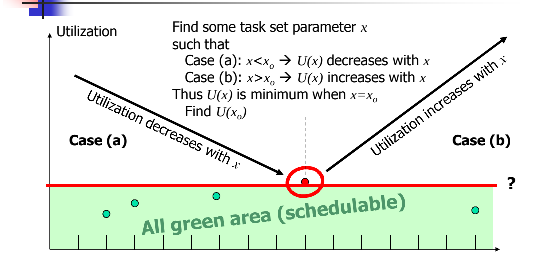
- Modified Question: is there a threshold U_{bound} such that
  - When $U < U_{bound}$ deadlines are met
  - When $U > U_{bound}$ deadlines **may or may not be** missed

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

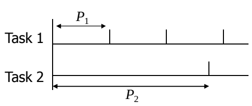
Deriving the Utilization Bound for Rate Monotonic Scheduling

- Consider a simple case: 2 tasks

Find some task set parameter x such that
Case (a): $x<x_o$ → $U(x)$ 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

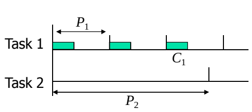
- Consider a simple case: 2 tasks



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

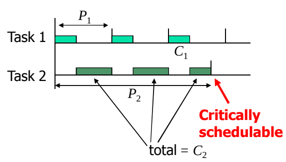
- Consider a simple case: 2 tasks



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

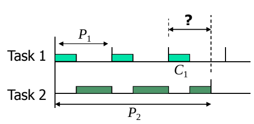
- Consider a simple case: 2 tasks



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

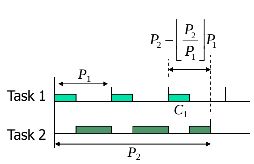
- Consider a simple case: 2 tasks



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



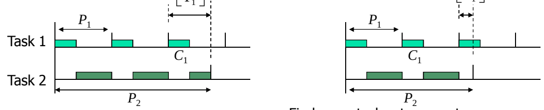
Find some task set parameter x such that
Case (a): $x<x_o$ → $U(x)$ decreases with x
Case (b): $x>x_o$ → $U(x)$ increases with x
Thus $U(x)$ is minimum when $x=x_o$
Find $U(x_o)$

Deriving the Utilization Bound for Rate Monotonic Scheduling

- Consider these two sub-cases:

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

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



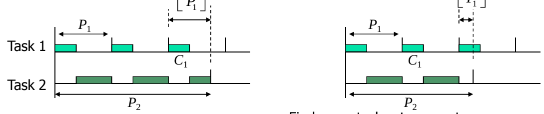
Find some task set parameter x such that
Case (a): $x<x_o$ → $U(x)$ decreases with x
Case (b): $x>x_o$ → $U(x)$ increases with x
Thus $U(x)$ is minimum when $x=x_o$
Find $U(x_o)$

Deriving the Utilization Bound for Rate Monotonic Scheduling

- Consider these two sub-cases:

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

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



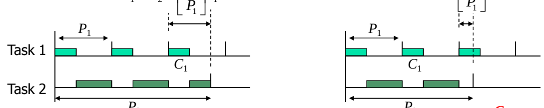
Find some task set parameter x such that
Case (a): $x<x_o$ → $U(x)$ decreases with x
Case (b): $x>x_o$ → $U(x)$ increases with x
Thus $U(x)$ is minimum when $x=x_o$
Find $U(x_o)$

Deriving the Utilization Bound for Rate Monotonic Scheduling

- Consider these two sub-cases:

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

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



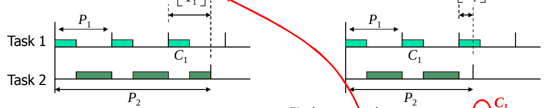
Find some task set parameter x C_1 such that
Case (a): $x<x_o$ → $U(x)$ decreases with x
Case (b): $x>x_o$ → $U(x)$ increases with x
Thus $U(x)$ is minimum when $x=x_o$
Find $U(x_o)$

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 $x=x_o$

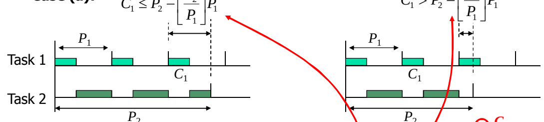
Find $U(x_o)$

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 $x=x_o$

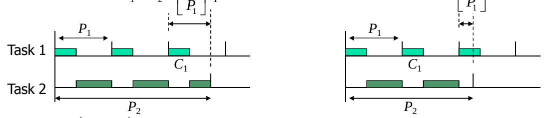
Find $U(x_o)$

Deriving the Utilization Bound for Rate Monotonic Scheduling

- Consider these two sub-cases:

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

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



Case (a):

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

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

Case (b):

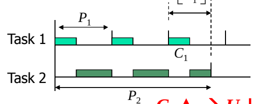
$$C_2 = (P_1 - C_1)\left\lfloor \frac{P_2}{P_1} \right\rfloor$$

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

Deriving the Utilization Bound for Rate Monotonic Scheduling

- Consider these two sub-cases:

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

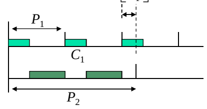


$C_1 \uparrow \rightarrow U \downarrow$

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

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

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



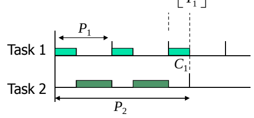
$$C_2 = (P_1 - C_1)\left\lfloor \frac{P_2}{P_1} \right\rfloor \quad C_1 \downarrow \rightarrow U \downarrow$$

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

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

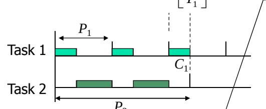
- The minimum utilization case:

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

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

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

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



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

$$\Rightarrow U \approx 0.83$$

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

Note that $C_1 = P_2 - P_1$

Generalizing to N Tasks

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

Generalizing to N Tasks

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

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

Generalizing to N Tasks

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

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

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

Generalizing to N Tasks

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

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

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

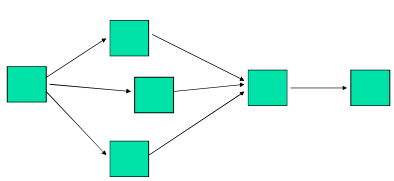
$$n \to \infty \quad U \to \ln 2$$

Aperiodic Tasks

- What if tasks do not arrive periodically?
- Sporadic tasks
  - There is a minimum separation between successive invocation arrivals
  - Treat minimum separation as period
- Aperiodic tasks
  - Feasible region calculus

Precedence Constraints

- Precedence constraints
  - Adjust arrival times and deadlines

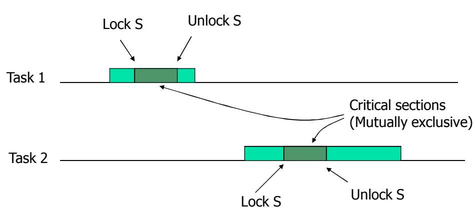


Processors and Resources

- In addition to the CPU, tasks may need resources
  - Memory
  - Disk
  - Access to shared data structures
  - etc.
- Resource types
  - Space-multiplexed (e.g., memory: different tasks have different parts of the resource)
  - Time multiplexed (one task can access at a time)
    - Serial: Two tasks can't interleave their accesses (e.g., lock-protected data structures)
- How do resource constraints affect scheduling and schedulability analysis?

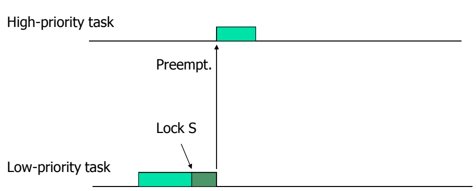
Mutual Exclusion Constraints

- Tasks that lock/unlock the same semaphore are said to have a mutual exclusion constraint



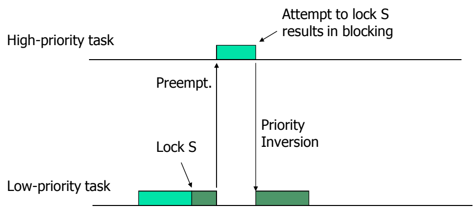
Priority Inversion

- Locks and priorities may be at odds. Locking results in priority inversion



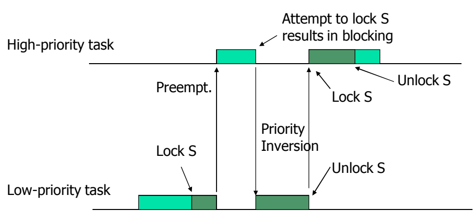
Priority Inversion

- Locks and priorities may be at odds. Locking results in priority inversion



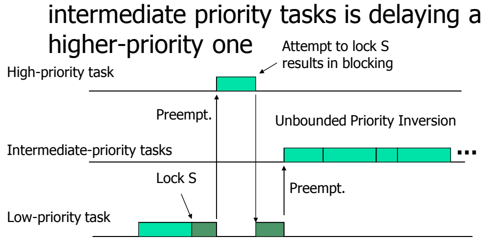
Priority Inversion

- How to account for priority inversion?



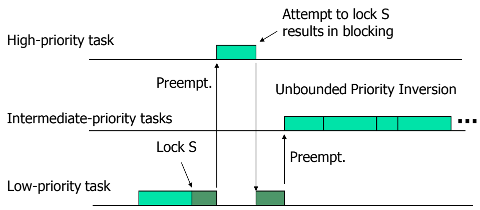
Unbounded Priority Inversion

- Consider the case below: a series of intermediate priority tasks is delaying a higher-priority one



Unbounded Priority Inversion

- How to prevent unbounded priority inversion?



Notes on Projects

- Some pointers to embedded hardware:
  - www.xbow.com
  - www.moteiv.com
    (Sensors, motes, processor boards, …)
  - www.x10.com
    (Environmental control interfaces, security cameras, …)
  - www.phidgets.com
    (Smart room gadgets)