Exact Schedulability Test

Tarek Abdelzaher
The 4th Credit Project
(Suggested: 1-2 persons per project)

- Option 1: Develop a 30 min survey presentation on an advanced topic of your choice in real-time and embedded computing.
  - Topic name due 10/24.
  - Slides due 11/28.
  - Presentation the week of 12/5

- Example topics:
  - Self-driving cars: the state of the art and future challenges
  - Real-time operating systems
  - Multicore scheduling – main challenges and results
  - Space applications
  - Scheduling Map/Reduce workflows (with emphasis on time support)
  - Participatory and social sensing (crowd-sensing)
  - Software model checking (proving software correctness)
  - Energy/smart grid
The 4th Credit Project
(Suggested: 1-2 persons per project)

- Option 2: Implement a real-time or embedded systems service
  - Service name due 10/24.
  - Slides due 11/28.
  - Presentation + Demo the week of 11/29

- Example services:
  - A real-time scheduler for Roomba
  - Security and diagnostics
  - Real-time Hadoop
  - Social sensing services
  - Your idea here…
Scheduling Taxonomy

Periodic Task Scheduling

- Rate Monotonic
- EDF

With Deadline < Period
Deadline Monotonic Scheduling

- Consider a set of periodic tasks where each task, $i$, has a computation time, $C_i$, a period, $P_i$, and a relative deadline $D_i < P_i$. 

\[ \begin{align*}
  &D_i \\
  &P_i
\end{align*} \]
Deadline Monotonic Scheduling

- Consider a set of periodic tasks where each task, $i$, has a computation time, $C_i$, a period, $P_i$, and a relative deadline $D_i < P_i$.

- What is the schedulability condition?
Deadline Monotonic Scheduling

- Consider a set of periodic tasks where each task, \( i \), has a computation time, \( C_i \), a period, \( P_i \), and a relative deadline \( D_i < P_i \).

  \[ \sum_{i} \frac{C_i}{D_i} \leq n \left(2^{1/n} - 1 \right) \]

- Schedulability can’t be worse than if \( P_i \) was reduced to \( D_i \). Thus:
Deadline Monotonic Scheduling

- Consider a set of periodic tasks where each task, \( i \), has a computation time, \( C_i \), a period, \( P_i \), and a relative deadline \( D_i < P_i \).

- Schedulability can’t be worse than if \( P_i \) was reduced to \( D_i \). Thus:

\[
\sum_{i} \frac{C_i}{D_i} \leq n \left(2^{1/n} - 1\right)
\]
A Better Condition

- Worst case interference from a higher priority task, $j$?
A Better Condition

- Worst case interference from a higher priority task, $j$?

\[
\begin{bmatrix}
D_i \\
\frac{D_i}{P_j}
\end{bmatrix}
C_j
\]
A Better Condition

- Worst case interference from a higher priority task, $j$?

- Schedulability condition: $C_i + \sum_j \left\lfloor \frac{D_i}{P_j} \right\rfloor C_j \leq D_i$
A Better Condition

- Worst case interference from a higher priority task, $j$?

- Schedulability condition:

\[
C_i + \sum_j \left\lceil \frac{D_i}{P_j} \right\rceil C_j \leq D_i
\]

Worst case interference, $I$, from higher priority tasks.

My exec. time

My deadline
A Better Condition

- Worst case interference from a higher priority task, \( j \)?

- Schedulability condition:
  \[
  C_i + \sum_j \left( \frac{D_i}{P_j} \right) C_j \leq D_i
  \]
An Exact Condition

Note: Interference exists only during the response time $R_i$ not the entire $D_i$

$$I = \sum_{j} \left[ \frac{R_i}{D_j} \right] C_j$$

Worst case interference, $I$, From higher priority tasks
An Exact Condition

Note: Interference exists only during the response time $R_i$ not the entire $D_i$

$$I = \sum_{j} \left[ \frac{R_i}{D_i} \right] C_j$$

where

$$R_i = I + C_i$$
An Exact Condition

Note: Interference exists only during the response time $R_i$ not the entire $D_i$

\[
I = \sum_j \left[ \frac{R_i}{D_j} \right] C_j
\]

where

\[
R_i = I + C_i
\]

Solve iteratively for the smallest $R_i$ that satisfies both equations
Example

Consider a system of two tasks:

Task 1: $P_1 = 1.7, D_1 = 0.5, C_1 = 0.5$
Task 2: $P_2 = 8, D_2 = 3.2, C_2 = 2$

$$I = \sum_j \left[ \frac{R_i}{P_j} \right] C_j$$

$$R_i = I + C_i$$
Consider a system of two tasks:

Task 1: \( P_1 = 1.7, \) \( D_1 = 0.5, \) \( C_1 = 0.5 \)
Task 2: \( P_2 = 8, \) \( D_2 = 3.2, \) \( C_2 = 2 \)

\[
I^{(0)} = C_1 = 0.5
\]
\[
R_2^{(0)} = I^{(0)} + C_2 = 2.5
\]
Example

Consider a system of two tasks:

Task 1: \( P_1 = 1.7, \ D_1 = 0.5, \ C_1 = 0.5 \)
Task 2: \( P_2 = 8, \ D_2 = 3.2, \ C_2 = 2 \)

\[
I = \sum_j \left( \frac{R_j}{P_j} \right) C_j
\]

\[
R_i = I + C_i
\]

\[
I^{(0)} = C_1 = 0.5
\]

\[
R_2^{(0)} = I^{(0)} + C_2 = 2.5
\]

\[
I^{(1)} = \left[ \frac{R_2^{(0)}}{P_1} \right] C_1 = \left[ \frac{2.5}{1.7} \right] 0.5 = 1
\]

\[
R_2^{(1)} = I^{(1)} + C_2 = 3
\]
Consider a system of two tasks:

Task 1: $P_1=1.7$, $D_1=0.5$, $C_1=0.5$

Task 2: $P_2=8$, $D_2=3.2$, $C_2=2$

$$I = \sum_j \left( \frac{R_i}{P_j} \right) C_j$$

$$R_i = I + C_i$$

$$I^{(0)} = C_1 = 0.5$$

$$R_2^{(0)} = I^{(0)} + C_2 = 2.5$$

$$I^{(1)} = \left[ \frac{R_2^{(0)}}{P_1} \right] C_1 = \left[ \frac{2.5}{1.7} \right] 0.5 = 1$$

$$R_2^{(1)} = I^{(1)} + C_2 = 3$$

$$I^{(2)} = \left[ \frac{R_2^{(1)}}{P_1} \right] C_1 = \left[ \frac{3}{1.7} \right] 0.5 = 1$$

$$R_2^{(2)} = I^{(2)} + C_2 = 3$$
Consider a system of two tasks:

Task 1: $P_1 = 1.7$, $D_1 = 0.5$, $C_1 = 0.5$

Task 2: $P_2 = 8$, $D_2 = 3.2$, $C_2 = 2$

$I^{(0)} = C_1 = 0.5$

$R_2^{(0)} = I^{(0)} + C_2 = 2.5$

$I^{(1)} = \left[ \frac{R_2^{(0)}}{P_1} \right] C_1 = \left[ \frac{2.5}{1.7} \right] 0.5 = 1$

$R_2^{(1)} = I^{(1)} + C_2 = 3$

$I^{(2)} = \left[ \frac{R_2^{(1)}}{P_1} \right] C_1 = \left[ \frac{3}{1.7} \right] 0.5 = 1$

$R_2^{(2)} = I^{(2)} + C_2 = 3$

$3 < 3.2 \rightarrow \text{Ok!}$
Mixed Periodic and Aperiodic Task Systems

Question: how to execute aperiodic tasks without violating schedulability guarantees given to periodic tasks?
Mixed Periodic and Aperiodic Task Systems

- Question: how to execute aperiodic tasks without violating schedulability guarantees given to periodic tasks?
- One Answer: Execute aperiodic tasks at lowest priority
  - Problem: Poor performance for aperiodic tasks
Mixed Periodic and Aperiodic Task Systems

- Idea: aperiodic tasks can be served by periodically invoked servers
- The server can be accounted for in periodic task schedulability analysis
- The server has a period $P_s$ and a budget $B_s$
- Server can serve aperiodic tasks until budget expires
- Servers have different flavors depending on the details of when they are invoked, what priority they have, and how budgets are replenished
Mixed Periodic and Aperiodic Task Systems

- Idea: aperiodic tasks can be served by periodically invoked servers
- The server can be accounted for in periodic task schedulability analysis
- The server has a period $P_s$ and a budget $B_s$
- Server can serve aperiodic tasks until budget expires
- Servers have different flavors depending on the details of when they are invoked, what priority they have, and how budgets are replenished
Mixed Periodic and Aperiodic Task Systems

- Idea: aperiodic tasks can be served by periodically invoked servers
- The server can be accounted for in periodic task schedulability analysis
- The server has a period $P_s$ and a budget $B_s$
- Server can serve aperiodic tasks until budget expires
- Servers have different flavors depending on the details of when they are invoked, what priority they have, and how budgets are replenished
Polling Server

- Runs as a periodic task (priority set according to RM)
- Aperiodic arrivals are queued until the server task is invoked
- When the server is invoked it serves the queue until it is empty or until the budget expires then suspends itself
  - If the queue is empty when the server is invoked it suspends itself immediately.
- Server is treated as a regular periodic task in schedulability analysis
Example of a Polling Server

- Polling server:
  - Period $P_s = 5$
  - Budget $B_s = 2$

- Periodic task
  - $P = 4$
  - $C = 1.5$

- All aperiodic arrivals have $C=1$
Example of a Polling Server

- Polling server:
  - Period $P_s = 5$
  - Budget $B_s = 2$

- Periodic task
  - $P = 4$
  - $C = 1.5$

- All aperiodic arrivals have $C=1$
Example of a Polling Server

- Polling server:
  - Period $P_s = 5$
  - Budget $B_s = 2$

- Periodic task
  - $P = 4$
  - $C = 1.5$

- All aperiodic arrivals have $C=1$
Example of a Polling Server

- Polling server:
  - Period $P_s = 5$
  - Budget $B_s = 2$

- Periodic task
  - $P = 4$
  - $C = 1.5$

- All aperiodic arrivals have $C=1$

Why not execute immediately?
Deferrable Server

- Keeps the balance of the budget until the end of the period
- Example (continued)
Worst-Case Scenario

Exercise: Derive the utilization bound for a deferrable server plus one periodic task

\[ U_p \leq \ln \left( \frac{U_s + 2}{2U_s + 1} \right) \]
Worst-Case Scenario

Exercise: Derive the utilization bound for a deferrable server plus one periodic task

\[ U_p \leq \ln \left( \frac{U_s + 2}{2U_s + 1} \right) \]
Priority Exchange Server

- Like the deferrable server, it keeps the budget until the end of server period
- Unlike the deferrable server the priority slips over time: When not used the priority is exchanged for that of the executing periodic task
Priority Exchange Server

Example

Aperiodic tasks

Priority Exchange Server

Periodic Tasks
Priority Exchange Server

Example

Aperiodic tasks

Priority Exchange Server

Periodic Tasks

\[ U_p \leq \ln \left( \frac{2}{U_s + 1} \right) \]
Sporadic Server

- Server is said to be *active* if it is in the *running* or *ready* queue, otherwise it is *idle*.
- When an aperiodic task comes and the budget is not zero, the server becomes active.
- Every time the server becomes *active*, say at $t_A$, it sets replenishment time one period into the future, $t_A + P_s$ (but does not decide on replenishment amount).
- When the server becomes idle, say at $t_I$, set replenishment amount to capacity consumed in $[t_A, t_I]$

\[ U_p \leq \ln \left( \frac{2}{U_s + 1} \right) \]
Slack Stealing Server

- Compute a slack function $A(t_s, t_f)$ that says how much total slack is available
- Admit aperiodic tasks while slack is not exceeded
Putting It All Together

- Covered so far:
  - System reliability
  - Data reliability
  - Timeliness

- Design problem: Design a “safe” robot
Putting It All Together

- Covered so far:
  - System reliability
  - Data reliability
  - Timeliness

- Design problem: Design a “safe” robot