Real-Time Scheduling Analysis

Real Time Operating Systems and Middleware

Luca Abeni
luca.abeni@unitn.it
Response Time Analysis

- Necessary and sufficient test: compute the worst-case response time for every task
- For every task $\tau_i$:
  - Compute worst case response time $R_i$ for $\tau_i$
  - Remember? $R_i = \max_j \{\rho_{i,j}\}$; $\rho_{i,j} = f_{i,j} - r_{i,j}$
  - If $R_i \leq D_i$, then the task is schedulable
  - otherwise, the task is not schedulable
- No assumption on the priority assignment
- Algorithm valid for arbitrary priority assignments
- Not only RM / DM...
- Periodic tasks with no offsets, or sporadic tasks
The Critical Instant

- Tasks ordered by decreasing priority \((i < j \rightarrow p_i > p_j)\)
- No assumptions about tasks offsets
  - \(\Rightarrow\) Consider the *worst possible offsets combination*
  - A job \(J_{i,j}\) released at the *critical instant* experiences the maximum response time for \(\tau_i\): \(\forall k, \rho_{i,j} \geq \rho_{i,k}\)
  - Simplified definition (jobs deadlines should be considered...)
- **Theorem:** The critical instant for task \(\tau_i\) occurs when job \(J_{i,j}\) is released at the same time with a job in every high priority task
- If all the offsets are 0, the first job of every task is released at the critical instant!!!
Worst Case Response Time

- Worst case response time $R_i$ for task $\tau_i$ depends on:
  - Its execution time...
  - ...And the execution time of higher priority tasks
  - Higher priority tasks can preempt task $\tau_i$, and increase its response time

\[ R_i = C_i + \sum_{h=1}^{i-1} \left\lfloor \frac{R_i}{T_h} \right\rfloor C_h \]
Computing the Response Time - I

\[ R_i = C_i + \sum_{h=1}^{i-1} \left\lceil \frac{R_i}{T_h} \right\rceil C_h \]

- Urk!!! \( R_i = f(R_i) \) ... How can we solve it?
- There is no closed-form expression for computing the worst case response time \( R_i \)
- We need an iterative method to solve the equation
Computing the Response Time - II

- **Iterative solution**
  - \( R_i = \lim_{k \to \infty} R_i^{(k)} \)
  - \( R_i^{(k)} \): worst case response time for \( \tau_i \), at step \( k \)
  - \( R_i^{(0)} \): first estimation of the response time

- **We can start with** \( R_i^{(0)} = C_i \)

- **\( R_i^{(0)} = C_i + \sum_{h=1}^{i-1} C_h \) saves 1 step**

\[
R_i^{(0)} = C_i + \sum_{h=1}^{i-1} C_h = C_i + \sum_{h=1}^{i-1} \left[ R_i^{(k-1)} \right] C_h
\]
Computing the Response Time - III

- Problem: are we sure that we find a valid solution?
- The iteration stops when:
  - \( R_i^{(k+1)} = R_i^{(k)} \) or
  - \( R_i^{(k)} > D_i \) (non schedulable);
- This is a standard method to solve non-linear equations in an iterative way.
- If a solution exists (the system is not overloaded), \( R_i^{(k)} \) converges to it.
- Otherwise, the “\( R_i^{(k)} > D_i \)” condition avoids infinite iterations.
Example

Task set: \( \tau_1 = (2, 5), \tau_2 = (2, 9), \tau_3 = (5, 20); U = 0.872 \)

\[
R_i^{(k)} = C_i + \sum_{h=1}^{i-1} \left[ \frac{R_i^{(k-1)}}{T_h} \right] C_h
\]

\[
R_3^{(0)} = C_3 + 1 \cdot C_1 + 1 \cdot C_2 = 9
\]
**Example**

Task set: \( \tau_1 = (2, 5), \tau_2 = (2, 9), \tau_3 = (5, 20); U = 0.872 \)

\[
R_i^{(k)} = C_i + \sum_{h=1}^{i-1} \left\lfloor \frac{R_i^{(k-1)}}{T_h} \right\rfloor C_h
\]

\[
R_3^{(1)} = C_3 + 2 \cdot C_1 + 1 \cdot C_2 = 11
\]
Example

Task set: \( \tau_1 = (2, 5), \tau_2 = (2, 9), \tau_3 = (5, 20); U = 0.872 \)

\[
R^{(k)}_i = C_i + \sum_{h=1}^{i-1} \left[ \frac{R^{(k-1)}_i}{T_h} \right] C_h
\]

\[
R^{(2)}_3 = C_3 + 3 \cdot C_1 + 2 \cdot C_2 = 15
\]
Task set: $\tau_1 = (2, 5), \tau_2 = (2, 9), \tau_3 = (5, 20); U = 0.872$

$$R_i^{(k)} = C_i + \sum_{h=1}^{i-1} \left[ \frac{R_i^{(k-1)}}{T_h} \right] C_h$$

$$R_3^{(3)} = C_3 + 3 \cdot C_1 + 2 \cdot C_2 = 15 = R_3^{(2)}$$
What about different priority assignments and deadlines different from periods?

\[
\tau_1 = (1, 4, 4), p_1 = 3, \tau_2 = (4, 6, 15), p_2 = 2, \\
\tau_3 = (3, 10, 10), p_3 = 1; U = 0.72
\]

\[
R_i^{(k)} = C_i + \sum_{h=1}^{i-1} \left( \frac{R_i^{(k-1)}}{T_h} \right) C_h
\]

\[
R_3^{(0)} = C_3 + 1 \cdot C_1 + 1 \cdot C_2 = 8
\]
What about different priority assignments and deadlines different from periods?

\( \tau_1 = (1, 4, 4), p_1 = 3, \tau_2 = (4, 6, 15), p_2 = 2, \)
\( \tau_3 = (3, 10, 10), p_3 = 1; U = 0.72 \)

\[
R_{i}^{(k)} = C_i + \sum_{h=1}^{i-1} \left[ \frac{R_{i}^{(k-1)}}{T_h} \right] C_h
\]

\[
R_3^{(1)} = C_3 + 2 \cdot C_1 + 1 \cdot C_2 = 9
\]
What about different priority assignments and deadlines different from periods?

\( \tau_1 = (1, 4, 4), p_1 = 3, \tau_2 = (4, 6, 15), p_2 = 2, \tau_3 = (3, 10, 10), p_3 = 1; U = 0.72 \)

\[ R_i^{(k)} = C_i + \sum_{h=1}^{i-1} \left[ \frac{R_i^{(k-1)}}{T_h} \right] C_h \]

\[ R_3^{(2)} = C_3 + 3 \cdot C_1 + 1 \cdot C_2 = 10 \]
What about different priority assignments and deadlines different from periods?

\[ \tau_1 = (1, 4, 4), p_1 = 3, \tau_2 = (4, 6, 15), p_2 = 2, \]

\[ \tau_3 = (3, 10, 10), p_3 = 1; U = 0.72 \]

\[ R_i^{(k)} = C_i + \sum_{h=1}^{i-1} \left[ \frac{R_i^{(k-1)}}{T_h} \right] C_h \]

\[ R_3^{(3)} = C_3 + 3 \cdot C_1 + 1 \cdot C_2 = 10 = R_3^{(2)} \]
Considerations

- The response time analysis is an efficient algorithm
- In the worst case, the number of steps $N$ for the algorithm to converge is exponential
- Depends on the total number of jobs of higher priority tasks in the interval $[0, D_i]$:

$$N \propto \sum_{h=1}^{i-1} \left\lceil \frac{D_h}{T_h} \right\rceil$$

- If $s$ is the minimum granularity of the time, then in the worst case $N = \frac{D_i}{s}$;
- However, such worst case is very rare: usually, the number of steps is low.
Processor Demand approach has been proposed by Lehoczky and others in 1989. Refined by Audsley and others in 1993 and by Baruah in 1990. The basic idea is very simple: in any interval, the computation demanded by all tasks in the set must never exceed the available time. The problem is: how to compute the time demanded by a task set $T$? Remember: we have to look only at jobs released at the critical instant. Offsets $= 0 \implies$ only consider the first job of each task...
The Processor Demand

- Given an interval \([t_1, t_2]\),
- let \(J_{t_1,t_2}\) be the set of jobs started after \(t_1\) and with deadline lower than or equal to \(t_2\):

\[
J_{t_1,t_2} = \{ J_{i,j} : r_{i,j} \geq t_1 \land d_{i,j} \leq t_2 \}
\]

- the processor demand in \([t_1, t_2]\) is defined as:

\[
W(t_1, t_2) = \sum_{J_{i,j} \in J_{t_1,t_2}} c_{i,j}
\]

- Worst case: use \(C_i\) instead of \(c_{i,j}\)
Guaranteeing a task set \( \mathcal{T} \) based on \( W(t_1, t_2) \) can take a loooong time

- \( \forall (t_1, t_2), W(t_1, t_2) \leq t_2 - t_1 \)
- Need to check all the \((t_1, t_2)\) combinations in a hyperperiod?

But...

- We only need to check the first job of every task \( \tau_i \)!

\[ W_i(t_1, t_2): \text{time demanded in } [t_1, t_2] \text{ by all tasks } \tau_j \text{ with } p_j \geq p_i \quad \Rightarrow \quad j \leq i \]
Computing the Processor Demand - 2

- We can consider only $W_i(0, t)$
- For task $\tau_i$, only check $W_i(0, t)$ for $0 \leq t \leq D_i$
- Change $\forall$ into $\exists$: consider worst case for $W_i()$
  - Number of jobs in $[0, t]$: $\left\lfloor \frac{t}{T_i} \right\rfloor$
  - Use $\lceil \rceil$ instead!
- We already have hints about computing an upper bound for $W_i(0, t)$...

$$W_i(0, t) = C_i + \sum_{h=1}^{i-1} \left\lceil \frac{t}{T_h} \right\rceil C_h$$
TDA: The Guarantee

- Task $\tau_i$ is schedulable iff $\exists t : 0 \leq t \leq D_i \land W_i(0, t) \leq t$
- A task set $\mathcal{T}$ is schedulable iff

$$\forall \tau_i \in \mathcal{T}, \exists t : 0 \leq t \leq D_i \land W_i(0, t) \leq t$$

- Sometimes, different notations in literature:
  - $W_i(0, t) \rightarrow W_i(t) = \sum_{h=1}^{i} \left\lfloor \frac{t}{T_h} \right\rfloor C_h$ (this is equivalent, because $0 \leq t \leq T_i$)
Someone defines

\[ L_i(t_1, t_2) = \frac{W_i(t_1, t_2)}{t_2 - t_1} \]

\[ L_i = \min_{0 \leq t \leq D_i} L_i(0, t); \quad L = \max_{\tau_i \in T} L_i \]

The guarantee tests then becomes

Task \( \tau_i \) is schedulable \iff \( L_i \leq 1 \)

\( T \) is schedulable \iff \( L \leq 1 \)
The test might still be long (need to check many values of $L(0, t)$ to find the minimum)...

The number of points to check for computing $W_i / L_i$ can be reduced

- **Scheduling points:** $S_i = \{kT_h | h \leq i; 1 \leq k \leq \left\lfloor \frac{T_i}{T_h} \right\rfloor \}$
  - (multiples of $T_h$ for $h \leq i$)
  - $L_i = \min_{t \in S_i} L_i(0, t)$
Example - 1

- $\tau_1 = (20, 100)$, $\tau_2 = (40, 150)$, $\tau_3 = (100, 350)$
- $\tau_1$ is schedulable: $20 < 100$
- What about $\tau_2$?
  - $S_2 = \{100, 150\}$
  - $W_2(0, 100) = 40 + 20 = 60 \leq 100$: $\tau_2$ is schedulable
Example - 2

- And now, $\tau_3$:
  - $S_3 = \{100, 150, 200, 300, 350\}$
  - $W_3(0, 100) = 100 + 20 + 40 = 160 > 100$
  - $W_3(0, 150) = 100 + 2 \times 20 + 40 = 180 > 150$
  - $W_3(0, 200) = 100 + 2 \times 20 + 2 \times 40 = 220 > 200$
  - $W_3(0, 300) = 100 + 3 \times 20 + 2 \times 40 = 240 \leq 300$: $\tau_3$ is schedulable
Example - Continued 1

- But we already knew that the task set is schedulable:
  \[
  \frac{20}{100} + \frac{40}{150} + \frac{100}{350} = 0.2 + 0.26 + 0.2857 = 0.753 < 0.779
  \]
- Now, let’s change \( C_1 \) to 40:
  - \( \tau_1 = (40, 100), \tau_2 = (40, 150), \tau_3 = (100, 350) \)
  - \( U = \frac{40}{100} + \frac{40}{150} + \frac{100}{350} = 0.953 \)
  - \( U_{lb} \leq U \leq 1 \): using utilisation-based analysis, we cannot say is \( T \) is schedulable or not...
  - \( \tau_1 \) is schedulable: \( 40 < 100 \)
• $\tau_1$ is schedulable, what about the other tasks?

• $\tau_2$: 
  - $W_2(0, 100) = 40 + 40 \leq 100$: $\tau_2$ is schedulable

• $\tau_3$: 
  - $W_3(0, 300) = 100 + 3 \times 40 + 2 \times 40 = 300 \leq 300$: $\tau_3$ is still schedulable!!!

• Note: to simplify, I directly checked the “right” scheduling point...
Example - Checking the Results

- TDA says that the task set is schedulable, but I do not believe it!!! ($U = 0.953$ looks very high)
Example - Checking the Results

- TDA says that the task set is schedulable, but I do not believe it!!! ($U = 0.953$ looks very high)
- So, here is the schedule...

![Schedule Diagram]
Checking the Results - Response Time

- And what about using response time analysis?
- \( \mathcal{T} = \{(40, 100), (40, 150), (100, 350)\} \)

- \( R_3^{(0)} = 100 \)
- \( R_3^{(1)} = 100 + \left\lceil \frac{100}{100} \right\rceil \cdot 40 + \left\lceil \frac{100}{150} \right\rceil \cdot 40 = 100 + 40 + 40 = 180 \)
- \( R_3^{(2)} = 100 + \left\lceil \frac{180}{100} \right\rceil \cdot 40 + \left\lceil \frac{180}{150} \right\rceil \cdot 40 = 100 + 80 + 80 = 260 \)
- \( R_3^{(3)} = 100 + \left\lceil \frac{260}{100} \right\rceil \cdot 40 + \left\lceil \frac{260}{150} \right\rceil \cdot 40 = 100 + 120 + 80 = 300 \)
- \( R_3^{(4)} = 100 + \left\lceil \frac{300}{100} \right\rceil \cdot 40 + \left\lceil \frac{300}{150} \right\rceil \cdot 40 = 100 + 120 + 80 = 300 \)
- \( R_3^{(4)} = R_3^{(3)} \Rightarrow \text{stop. } R_3 = 300 \)

- \( R_3 = 300 \leq 350 \Rightarrow \text{the system is schedulable.} \)
- The previous result is confirmed...