Static priority scheduling

Michal Sojka

Czech Technical University in Prague,
FEE and CIIRCC

November 24, 2021

Some slides are derived from lectures by Steve Goddard and James H. Anderson
Classification of scheduling algorithms
(used in real-time systems)

Scheduling algorithms

- Off-line scheduling (static, clock-driven)
  - Static-priority scheduling (VxWorks, SCHED_FIFO)

- On-line scheduling (dynamic)
  - Deadline-driven scheduling (EDF, ...)
  - General purpose OS scheduling (fair, interactive, ...)

General purpose OS scheduling (fair, interactive, ...)
1 Introduction

2 RM and DM scheduling and their optimality

3 Utilization-based schedulability tests

4 Time demand analysis and variants
   - Time demand analysis
   - Response-time analysis
   - Tasks with arbitrary deadlines

5 Summary
Outline

1. Introduction
2. RM and DM scheduling and their optimality
3. Utilization-based schedulability tests
4. Time demand analysis and variants
   - Time demand analysis
   - Response-time analysis
   - Tasks with arbitrary deadlines
5. Summary
Static priority scheduling

Fixed-priority scheduling

- All jobs of a single task have the same (static, fixed) priority
- We will assume that tasks are indexed in decreasing priority order, i.e. \( \tau_i \) has higher priority than \( \tau_k \) if \( i < k \).
- We will assume that no two tasks have the same priority.

Notation

- \( p_i \) denotes the priority of \( \tau_i \).
- \( hp(\tau_i) \) denotes the subset of tasks with higher priority than \( \tau_i \).
Basic questions

- How to assign task priorities?
- How to verify that all deadlines are met (schedulability)?
Outline

1 Introduction

2 RM and DM scheduling and their optimality

3 Utilization-based schedulability tests

4 Time demand analysis and variants
   - Time demand analysis
   - Response-time analysis
   - Tasks with arbitrary deadlines

5 Summary
Rate-Monotonic scheduling
CZ: Rozvrhování podle frekvence (Liu, Layland)

Rate-Monotonic priority assignment

The less period $T_i$ the higher priority $p_i$.
For every two tasks $\tau_i$ and $\tau_j$: $T_i < T_j \Rightarrow p_i > p_j$.

Example (RM schedule)

Three tasks $(T, C)$: $\tau_1 = (3, 0.5)$, $\tau_2 = (4, 1)$ a $\tau_3 = 6, 2$. 
Rate-Monotonic priority assignment

<table>
<thead>
<tr>
<th>Task</th>
<th>Period</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>τ₁</td>
<td>25</td>
<td>0</td>
</tr>
<tr>
<td>τ₂</td>
<td>60</td>
<td>2</td>
</tr>
<tr>
<td>τ₃</td>
<td>42</td>
<td>1</td>
</tr>
<tr>
<td>τ₄</td>
<td>105</td>
<td>4</td>
</tr>
<tr>
<td>τ₅</td>
<td>75</td>
<td>3</td>
</tr>
</tbody>
</table>
Deadline-Monotonic scheduling
Rozvrhování podle termínu dokončení (Leung, Whitehead)

Deadline-Monotonic priority assignment

The earlier deadline \( D_i \), the higher priority \( p_i \).

Pro each two tasks \( \tau_i \) a \( \tau_j \):
\[ D_i < D_j \Rightarrow p_i > p_j. \]

Example (DM schedule)

Let’s change the RM example by tightening deadline of \( \tau_2 = (T, C, D) \):
\[ \tau_1 = (3, 0.5), \tau_2 = (4, 1, 2) \text{ a } \tau_3 = 6, 2. \]
### RM vs. DM

- **The periodic tasks given by:**

<table>
<thead>
<tr>
<th>task</th>
<th>$r_i$</th>
<th>$T_i$</th>
<th>$C_i$</th>
<th>$D_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>50</td>
<td>50</td>
<td>25</td>
<td>100</td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>0</td>
<td>62.5</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>$\tau_3$</td>
<td>0</td>
<td>125</td>
<td>25</td>
<td>50</td>
</tr>
</tbody>
</table>

- **DM schedule:**

  ![Graph showing DM schedule](image_url)
### RM vs. DM

- **The periodic tasks given by:**

<table>
<thead>
<tr>
<th>task</th>
<th>$r_i$</th>
<th>$T_i$</th>
<th>$C_i$</th>
<th>$D_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_1$</td>
<td>50</td>
<td>50</td>
<td>25</td>
<td>100</td>
</tr>
<tr>
<td>$\tau_2$</td>
<td>0</td>
<td>62.5</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>$\tau_3$</td>
<td>0</td>
<td>125</td>
<td>25</td>
<td>50</td>
</tr>
</tbody>
</table>

- **RM schedule:**

  - $\tau_1$
  - $\tau_2$
  - $\tau_3$

- **DM fails $\Rightarrow$ RM fails. DM can produce feasible schedule when RM fails.**
Theorem

*Neither RM nor DM is optimal.*

Proof.

Consider \( \tau_1 = (2, 1) \) a \( \tau_2 = (5, 2.5) \). Total system utilization is 1, so the system is schedulable (see the EDF lecture). However, under RM or DM, a deadline will be missed, regardless of how we choose to (statically) prioritize \( \tau_1 \) and \( \tau_2 \).
Simply periodic systems

RM algorithm is optimal when the periodic tasks in the system are simply periodic and the deadlines of the tasks are no less than their respective periods.

Definition

A system of periodic tasks is **simply periodic**\(^1\) if for every pair of tasks \(\tau_i\) and \(\tau_k\) in the system where \(T_i < T_k\), \(T_k\) is an integer multiple of \(T_i\).

Theorem

A system \(\mathcal{T}\) of simply periodic, independent, preemptable tasks, whose relative deadlines are at least their periods, is schedulable on one processor according to the RM algorithm if and only if its total utilization is at most one.

In practice, people often use periods: 1 ms, 10 ms, 100 ms and 1 s.

\(^1\)CZ: jednoduše periodický
Proof

- We wish to show: \( U \leq 1 \Rightarrow \mathcal{T} \) is schedulable.
- We prove the contrapositive: \( \mathcal{T} \) is not schedulable \( \Rightarrow U > 1. \)
- Assume \( \mathcal{T} \) is not schedulable. Let \( J_{i,k} \) be the first job to miss its deadline.

Note: We suppose that tasks are in phase (not shown in the figure above) and processor never idles before \( J_{i,k} \) missed its deadline.

Because the system is simply periodic, \( \frac{r_{i,k+1} - t_{-1}}{\mathcal{T}_j} \) is integer.
Because $J_{i,k}$ missed its deadline, the demand placed on the processor in $[t_{-1}, r_{i,k+1})$ by jobs of tasks $\tau_1, \ldots, \tau_i$ is greater than the available processor time in $[t_{-1}, r_{i,k+1}]$. Thus:

$$r_{i,k+1} - t_{-1} = \text{available processor time in } [t_{-1}, r_{i,k+1}] < \sum_{j=1}^{i} (\text{the number of jobs of } \tau_j \text{ released in } [t_{-1}, r_{i,k+1}) \cdot C_j \leq \sum_{j=1}^{i} \frac{r_{i,k+1} - t_{-1}}{T_j} \cdot C_j$$
This we have

\[ r_{i,k+1} - t_{-1} < \sum_{j=1}^{i} \frac{r_{i,k+1} - t_{-1}}{T_j} \cdot C_j \]

Canceling \( r_{i,k+1} - t_{-1} \) yields

\[ 1 < \sum_{j=1}^{i} \frac{C_j}{T_j} , \]

i.e.

\[ 1 < U_i < U , \]

This completes the proof.
Optimality among fixed-priority algorithms

**Theorem**

A system $T$ of independent, preemptable, periodic, synchronous tasks that have relative deadlines at most their respective periods can be feasibly scheduled on one processor according to the DM algorithm whenever it can be feasibly scheduled according to any fixed-priority algorithm.

**Corollary**

The RM algorithm is optimal among all fixed-priority algorithms whenever the relative deadlines of all tasks are proportional to their periods.
Proof

- We can always transform a feasible static-priority schedule that is not a DM schedule into one that is.

- Suppose \( \tau_1, \ldots, \tau_i \) are prioritized not in accordance with DM. Suppose \( \tau_i \) has a longer relative deadline than \( \tau_{i+1} \), but \( \tau_i \) a higher priority than \( \tau_{i+1} \). Then, we can interchange \( \tau_i \) and \( \tau_{i+1} \) (switch the priorities) and adjust the schedule accordingly by swapping “pieces” of \( \tau_i \) with “pieces” of \( \tau_{i+1} \).

![Diagram showing the transformation of tasks](image-url)
After the switch, the priorities of the two tasks are assigned on the DM basis relative to the other tasks.

By induction, we can correct all such situations and transform the given schedule into DM schedule.

Note: It is always possible to switch the priorities of tasks and hence the time intervals without leading to any missed deadline when tasks are in phase. Why?
Outline

1 Introduction

2 RM and DM scheduling and their optimality

3 Utilization-based schedulability tests

4 Time demand analysis and variants
   - Time demand analysis
   - Response-time analysis
   - Tasks with arbitrary deadlines

5 Summary
Utilization-based RM schedulability test

A system of \( n \) independent, preemptable periodic tasks with relative deadlines equal to their respective periods can be feasibly scheduled on one processor according to the RM algorithm if its total utilization \( U = \sum_{i=1}^{n} u_i \) satisfies:

\[
U \leq n(2^{\frac{1}{n}} - 1)
\]

\( U_{RM}(n) = n(2^{\frac{1}{n}} - 1) \) is the schedulable utilization of the RM algorithm.

**Note:** This is only a sufficient (not necessary) schedulability test.
**$U_{RM}$ as a function of $n$**

<table>
<thead>
<tr>
<th>$n$</th>
<th>$U_{RM}(n)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0.828</td>
</tr>
<tr>
<td>3</td>
<td>0.779</td>
</tr>
<tr>
<td>4</td>
<td>0.756</td>
</tr>
<tr>
<td>5</td>
<td>0.743</td>
</tr>
<tr>
<td>6</td>
<td>0.734</td>
</tr>
<tr>
<td>7</td>
<td>0.728</td>
</tr>
<tr>
<td>8</td>
<td>0.724</td>
</tr>
<tr>
<td>9</td>
<td>0.720</td>
</tr>
<tr>
<td>10</td>
<td>0.717</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>$\infty$</td>
<td>$\ln 2 \approx 0.693$</td>
</tr>
</tbody>
</table>
Proof

See Liu’s book.
Other utilization-based tests

Liu’s book presents several other utilization-based schedulability tests.

- Some of these tests result in higher schedulable utilizations for certain kinds of task sets.
- Other deal with different task models such as those similar to MPEG decoder.
Outline

1 Introduction

2 RM and DM scheduling and their optimality

3 Utilization-based schedulability tests

4 Time demand analysis and variants
   - Time demand analysis
   - Response-time analysis
   - Tasks with arbitrary deadlines

5 Summary
**Critical instant**

The following analysis methods are based on a notion of “critical instant”.

**Definition (Critical instant)**

Critical instant of a task $\tau_i$ is a time instant such that:

1. the job of $\tau_i$ released at this instant has the maximum response time of all jobs in $\tau_i$, if the response time of every job of $\tau_i$ is at most $D_i$, the relative deadline of $\tau_i$ and
2. the response time of the job released at this instant is greater than $D_i$ if the response time of some jobs in $\tau_i$ exceeds $D_i$.

Informally, a critical instant of $\tau_i$ represents a worst-case scenario from $\tau_i$ standpoint.
Critical instant in static-priority systems

Theorem (Liu, Layland)

*In a fixed-priority system where every job completes before the next job of the same task is released, a critical instant of any task $\tau_i$ occurs when one of its job $J_{i,c}$ is released at the same time with a job of every higher priority task.*

We are not saying that $\tau_1, \ldots, \tau_i$ will all necessarily release jobs at the same time, but if this does happen, we are claiming that the time of release will be a critical instant for $\tau_i$. 
Time-demand analysis (TDA)

- Compute the total demand for processor time by a job released at a critical instant of the task and by all the higher-priority tasks.
- Check whether this demand can be met before the deadline of the job.
- TDA can be applied to produce a schedulability test for any static-priority algorithm that ensures that each job of every task completes before the next job of that task is released.
- For some important task models and scheduling algorithms, this schedulability test will be necessary and sufficient.
- Time-demand analysis was proposed by Lehoczky, Sha, and Ding.
Scheduling condition

Definition

The **time demand function** of the task $\tau_i$, denoted $w_i(t)$, is defined as follows.

$$w_i(t) = C_i + \sum_{k=1}^{i-1} \left\lfloor \frac{t}{T_k} \right\rfloor \cdot C_k \quad \text{for } 0 < t \leq T_i.$$

**Note:** We are still assuming tasks are indexed by priority.

For any static-priority algorithm $\mathcal{A}$ that ensures that each job of every task completes by the time the next job of that task is released.

Theorem

*System $\mathcal{T}$ of periodic, independent, preemptable tasks is schedulable on one processor by algorithm $\mathcal{A}$ if the following holds.*

\[ \forall i : \exists t : w_i(t) \leq t, \quad 0 < t \leq T_i. \]
Necessity and sufficiency

Condition \( \forall i : \exists t : w_i(t) \leq t, \quad 0 < t \leq T_i \) is necessary for
- synchronous real periodic task systems and
- real sporadic task systems.

Why?

For a given \( i \), we don’t really have to consider all \( t \) in the range \( 0 < t \leq T_i \). Two ways to avoid this:

1. Iterate using \( t^{(k+1)} := w_i(t^{(k)}) \), starting with a suitable \( t^{(0)} \) (e.g. \( t^{(0)} = C_i \)) and stopping when, for some \( n \), \( t^{(n)} \geq w_i(t^{(n)}) \) or \( t^{(n)} > T_i \).
2. Only consider \( t = j \cdot T_k \), where \( k = 1, 2, \ldots, i; \)
   \( j = 1, 2, \ldots, \lceil \min(T_i, D_i) / T_k \rceil \).
   - Explanation is in Liu’s book.
Response-time analysis
CZ: Výpočet doby odezvy

- A special (simple) case of Time Demand Analysis
- If we know the critical instant, we can find task’s worst-case response time by “simulating” the schedule from the critical instant.
- The found worst-case response time $R_i$ is compared with the corresponding deadline:

$$R_i \leq D_i$$

- The response time can be calculated as follows:

$$R_i = C_i + I_i,$$

where $I_i$ denotes the interference from higher priority tasks.
- In time interval $[0, R_i)$ the higher priority task $\tau_j$ will be executed several times:

$$\text{number of } \tau_j \text{ executions} = \left\lfloor \frac{R_i}{T_j} \right\rfloor$$

- Total interference from $\tau_j$ to $\tau_i$ is:

$$I_{i,j} = \left\lfloor \frac{R_i}{T_j} \right\rfloor C_j$$
Response-time analysis (cont.)

\[ R_i = C_i + \sum_{j \in hp(i)} l_{i,j} \]

\[ R_i = C_i + \sum_{j \in hp(i)} \left\lfloor \frac{R_i}{T_j} \right\rfloor C_j \]

Sequence \( w_0, w_1, \ldots \) is monotonically non-decreasing. If \( w_n = w_{n+1} \) then it is the solution of the equation. Choice of \( w_0 \) is important. It should not be greater than \( R_i \). We can start with \( 0 \) or \( C_i \).
Response-time analysis (cont.)

\[ R_i = C_i + \sum_{j \in \text{hp}(i)} I_{i,j} \]

\[ R_i = C_i + \sum_{j \in \text{hp}(i)} \left\lfloor \frac{R_i}{T_j} \right\rfloor C_j \]
Response-time analysis (cont.)

\[ R_i = C_i + \sum_{j \in hp(i)} l_{i,j} \]

\[ R_i = C_i + \sum_{j \in hp(i)} \left\lfloor \frac{R_i}{T_j} \right\rfloor C_j \]

- Can be calculated by the recurrence formula:

\[ w_{i}^{n+1} = C_i + \sum_{j \in hp(i)} \left\lfloor \frac{w_i^n}{T_j} \right\rfloor C_j \]

- Sequence \( w_i^0, w_i^1, \ldots \) is monotonically non-decreasing. If \( w_i^n = w_i^{n+1} \) then it is the solution of the equation. Choice of \( w_i^0 \) is important. It should not be greater than \( R_i \). We can start with 0 or \( C_i \).
Response-time analysis example

<table>
<thead>
<tr>
<th>Task</th>
<th>Period</th>
<th>Execution time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau_a$</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>$\tau_b$</td>
<td>12</td>
<td>3</td>
</tr>
<tr>
<td>$\tau_c$</td>
<td>20</td>
<td>5</td>
</tr>
</tbody>
</table>

$R_a = 3$

$w_0^0 = 3$

$w_0^1 = 3 + \left\lfloor \frac{3}{7} \right\rfloor \cdot 3 = 6$

$w_0^2 = 3 + \left\lfloor \frac{6}{7} \right\rfloor \cdot 3 = 6$

$R_b = 6$
Response-time analysis example (cont.)

Example

\[ w_c^0 = 5 \]
\[ w_c^1 = 5 + \left[ \frac{5}{7} \right] 3 + \left[ \frac{5}{12} \right] 3 = 11 \]
\[ w_c^2 = 5 + \left[ \frac{11}{7} \right] 3 + \left[ \frac{11}{12} \right] 3 = 14 \]
\[ w_c^3 = 5 + \left[ \frac{14}{7} \right] 3 + \left[ \frac{14}{12} \right] 3 = 17 \]
\[ w_c^4 = 5 + \left[ \frac{17}{7} \right] 3 + \left[ \frac{17}{12} \right] 3 = 20 \]
\[ w_c^5 = 5 + \left[ \frac{20}{7} \right] 3 + \left[ \frac{20}{12} \right] 3 = 20 \]
\[ R_c = 20 \]
Fixed-priority tasks with arbitrary deadlines

- $D_i > T_i \Rightarrow$ More than one job of a task can be ready for execution at a time.
- We assume that jobs are schedules in FIFO order.
- TDA schedulability condition holds only if $d_{i,k} < r_{i,k+1}$
Busy interval

Definition (Level-$p_i$ busy interval)

A level-$p_i$ busy interval $(t_0, t]$ begins at an instant $t_0$, when

1. all jobs in $h_p(\tau_i)$ released before the instant have completed and
2. a job in $h_p(\tau_i)$ is released.

The interval ends at the first instant $t$ after $t_0$ when all the jobs in $h_p(t_i)$ released since $t_0$ are complete.

Example (Priority $p_i = i$)
Busy interval

**Definition (Level-$p_i$ busy interval)**

A level-$p_i$ busy interval $(t_0, t]$ begins at an instant $t_0$, when

1. all jobs in $hp(\tau_i)$ released before the instant have completed and
2. a job in $hp(\tau_i)$ is released.

The interval ends at the first instant $t$ after $t_0$ when all the jobs in $hp(\tau_i)$ released since $t_0$ are complete.

**Example (Priority $p_i = i$)**

![Diagram showing busy intervals for tasks $\tau_1$, $\tau_2$, and $\tau_3$. The busy intervals are labeled Level-1, Level-2, and Level-3, respectively.](image-url)
Busy interval

Definition (Level-\(p_i\) busy interval)

A level-\(p_i\) busy interval \((t_0, t]\) begins at an instant \(t_0\), when

1. all jobs in \(hp(\tau_i)\) released before the instant have completed and
2. a job in \(hp(\tau_i)\) is released.

The interval ends at the first instant \(t\) after \(t_0\) when all the jobs in \(hp(t_i)\) released since \(t_0\) are complete.

Example (Priority \(p_i = i\))
In phase busy interval

Definition

A level-$p_i$ busy interval is in phase if the first jobs of all tasks in $hp(\tau_i)$ that are executed in this interval have the same release time.

Example (Priority $p_i = i$)

- $\tau_1$ has an in phase level-2 busy interval from time 4 to 7.
- $\tau_2$ has an in phase level-3 busy interval from time 6 to 8.
- $\tau_3$ has an in phase level-3 busy interval from time 6 to 8.
Analysis of tasks with arbitrary deadlines is not as easy

Lehoczky counterexample

- Consider two tasks $\tau_1 = (70, 26)$, $\tau_2 = (100, 62)$.
- Schedule:

- Seven jobs of $\tau_2$ execute in the first level-2 busy interval.
- Their response times are: 114, 102, 116, 104, 118, 106, 94.
- Response time of the first job is not the largest.
Test one task at a time from $\tau_1$ to $\tau_N$ (see next slides).

For the purpose of determining whether a task $\tau_i$ is schedulable, assume that all the tasks are in phase and the first level-$\rho_i$ busy interval begins at time 0.

While testing whether all the jobs in $\tau_i$ can meet their deadlines (i.e., whether $\tau_i$ is schedulable), consider the subset $hp(\tau_i)$ of tasks.
Response time calculation (cont.)

1. If the first job of every task in \(\text{hp}(\tau_i)\) completes by the end of the first period of the task, check whether the first job \(\tau_{i,1}\) meets its deadline. \(\tau_i\) is schedulable if \(\tau_{i,1}\) completes in time. Otherwise, \(\tau_i\) is not schedulable.\(^2\)

\(^2\)The same case as if \(D_i \leq T_i\)
Response time calculation (cont.)

1. If the first job of every task in $h(p(\tau_i))$ completes by the end of the first period of the task, check whether the first job $\tau_{i,1}$ meets its deadline. $\tau_i$ is schedulable if $\tau_{i,1}$ completes in time. Otherwise, $\tau_i$ is not schedulable.²

2. If the first job of some task in $h(p(\tau_i))$ does not complete by the end of the first period of the task, do the following:

²The same case as if $D_i \leq T_i$
Response time calculation (cont.)

1. If the first job of every task in $hp(\tau_i)$ completes by the end of the first period of the task, check whether the first job $\tau_{i,1}$ meets its deadline. $\tau_i$ is schedulable if $\tau_{i,1}$ completes in time. Otherwise, $\tau_i$ is not schedulable.\(^2\)

2. If the first job of some task in $hp(\tau_i)$ does not complete by the end of the first period of the task, do the following:
   a. Compute the length of the in phase level-$p_i$ busy interval by solving the equation $t = \sum_{k=1}^{i} \left\lceil \frac{t_{i,k}}{T_k} \right\rceil C_k$ iteratively, starting from $t^{(1)} = \sum_{k=1}^{i} C_k$ until $t^{(l+1)} = t^{(l)}$ for some $l \geq 1$. The solution $t^{(l)}$ is the length of the level-$p_i$ busy interval.

\(^2\)The same case as if $D_i \leq T_i$
Response time calculation (cont.)

1. If the first job of every task in $hp(\tau_i)$ completes by the end of the first period of the task, check whether the first job $\tau_{i,1}$ meets its deadline. \( \tau_i \) is schedulable if $\tau_{i,1}$ completes in time. Otherwise, \( \tau_i \) is not schedulable.\(^2\)

2. If the first job of some task in $hp(\tau_i)$ does not complete by the end of the first period of the task, do the following:
   
   a. Compute the length of the in phase level-$p_i$ busy interval by solving the equation $t = \sum_{k=1}^{i} \lceil \frac{t}{T_k} \rceil C_k$ iteratively, starting from $t^{(1)} = \sum_{k=1}^{i} C_k$ until $t^{(l+1)} = t^{(l)}$ for some $l \geq 1$. The solution $t^{(l)}$ is the length of the level-$p_i$ busy interval.
   
   b. Compute the maximum response times of all $\lceil \frac{t^{(l)}}{T_i} \rceil$ jobs of $\tau_i$ in the in-phase level-$p_i$ busy interval in the manner described next and determine whether they complete in time.

---

\(^2\)The same case as if $D_i \leq T_i$
Response time calculation (cont.)

1. If the first job of every task in \( hp(\tau_i) \) completes by the end of the first period of the task, check whether the first job \( \tau_{i,1} \) meets its deadline. \( \tau_i \) is schedulable if \( \tau_{i,1} \) completes in time. Otherwise, \( \tau_i \) is not schedulable.\(^2\)

2. If the first job of some task in \( hp(\tau_i) \) does not complete by the end of the first period of the task, do the following:
   a. Compute the length of the in phase level-\( p_i \) busy interval by solving the equation \( t = \sum_{k=1}^{i} \left\lceil \frac{t_{T_k}}{C_k} \right\rceil \) iteratively, starting from \( t^{(1)} = \sum_{k=1}^{i} C_k \) until \( t^{(l+1)} = t^{(l)} \) for some \( l \geq 1 \). The solution \( t^{(l)} \) is the length of the level-\( p_i \) busy interval.
   b. Compute the maximum response times of all \( \lceil t^{(l)} / T_i \rceil \) jobs of \( \tau_i \) in the in-phase level-\( p_i \) busy interval in the manner described next and determine whether they complete in time.
   c. \( \tau_i \) is schedulable if all these jobs complete in time; otherwise \( \tau_i \) is not schedulable.

\(^2\)The same case as if \( D_i \leq T_i \)
Response time calculation of the first job

Step 2b

- Almost the same as in the time demand analysis for $D_i \leq T_i$.
- The time-demand function $w_{i,1}$ is defined as follows:

$$w_{i,1}(t) = C_i + \sum_{k=1}^{i-1} \left\lfloor \frac{t}{T_k} \right\rfloor \cdot C_k \quad \text{for} \ 0 < t \leq w_{i,1}(t)$$

The maximum possible response time $R_{i,1}$ of job $\tau_{i,1}$ is

$$R_{i,1} = \min\{t | t = w_{i,1}(t)\}$$

- The same iterative computation as in response-time analysis.
Response time calculation for any job in the busy interval

Step 2b

Lemma

The maximum response time $R_{i,j}$ of the $j$-th job of $\tau_i$ in an in-phase level-$p_i$ busy interval is

$$R_{i,j} = \min \left\{ t \mid t = w_{i,j}(t + (j - 1)T_i) - (j - 1)T_i \right\},$$

where

$$w_{i,j}(t) = jC_i + \sum_{k=1}^{i-1} \left\lfloor \frac{t}{T_k} \right\rfloor \cdot C_k \quad \text{for} \quad (j - 1)T_i < t \leq w_{i,j}(t).$$

Can be solved by the recurrence relation as before.
Response time calculation for any job in the busy interval

Step 2b

Lemma

The maximum response time $R_{i,j}$ of the $j$-th job of $\tau_i$ in an in-phase level-$p_i$ busy interval is

$$R_{i,j} = \min \left\{ t | t = w_{i,j}(t + (j - 1)T_i) - (j - 1)T_i \right\},$$

where

$$w_{i,j}(t) = jC_i + \sum_{k=1}^{i-1} \left[ \frac{t}{T_k} \right] \cdot C_k \quad \text{for} \quad (j - 1)T_i < t \leq w_{i,j}(t).$$

Can be solved by the recurrence relation as before.
Example

Let us apply the previous Lemma to the Lehoczky example:
\[ \tau_1 = (70, 26), \tau_2 = (100, 62). \]

- 1) Does not apply because \( R_{2,1} > T_2 \).

- 2a) The length of level-2 busy interval is 694.

- 2b) Compute response times \( R_{2,j} \) for \( 1 \leq j \leq \lceil 695/100 \rceil = 7 \).
Example – calculation of $R_{2,1}$

$\tau_1 = (70, 26), \tau_2 = (100, 62)$.

$$R_{2,1} = \text{minimal } t \text{ satisfying:}$$

$$t = w_{2,1}(t) = C_2 + \sum_{k=1}^{2-1} \left\lfloor \frac{t}{T_k} \right\rfloor \cdot C_k$$

$$= 62 + \left\lfloor \frac{t}{70} \right\rfloor \cdot 26$$

Try substitute 114 for $t$:

$$114 = 62 + \left\lceil \frac{114}{70} \right\rceil \cdot 26 = 62 + 2 \cdot 26 = = 114 \text{ OK!}$$

What if we don't know what to substitute?
Example – calculation of $R_{2,2}$

$\tau_1 = (70, 26), \tau_2 = (100, 62)$.

$R_{2,2} =$ minimal $t$ satisfying:

$$t = w_{2,2}(t + T_2) - T_2 =$$

$$= 2C_2 + \sum_{k=1}^{2-1} \left\lfloor \frac{t + 100}{T_k} \right\rfloor \cdot C_k - 100 =$$

$$= 124 + \left\lfloor \frac{(t + 100)}{70} \right\rfloor \cdot 26 - 100$$

Try substitute 102 for $t$:

$$102 = 124 + \left\lfloor \frac{202}{70} \right\rfloor \cdot 26 - 100 =$$

$$= 124 + 3 \cdot 26 - 100 =$$

$$= 102 \text{ OK!}$$
Example – calculation of $R_{2,3}$

$\tau_1 = (70, 26), \tau_2 = (100, 62)$.

\[ R_{2,3} = \text{minimal } t \text{ satisfying:} \]
\[ t = w_{2,3}(t + 2T_2) - 2T_2 = \]
\[ = 3C_2 + \sum_{k=1}^{2-1} \left\lfloor \frac{t + 200}{T_k} \right\rfloor \cdot C_k - 200 = \]
\[ = 186 + \left\lfloor (t + 200)/70 \right\rfloor \cdot 26 - 200 \]

Try substitute 116 for $t$:
\[ 116 = 186 + \left\lfloor 316/70 \right\rfloor \cdot 26 - 200 = \]
\[ = 186 + 5 \cdot 26 - 200 = \]
\[ = 116 \text{ OK!} \]
1 Introduction

2 RM and DM scheduling and their optimality

3 Utilization-based schedulability tests

4 Time demand analysis and variants
   - Time demand analysis
   - Response-time analysis
   - Tasks with arbitrary deadlines

5 Summary
RM and DM are optimal among fixed priority scheduling algorithms.

Utilization based test is simple but only a sufficient condition.

Response time analysis is both sufficient and necessary.

Generic time demand analysis can handle many cases not covered by response time analysis.