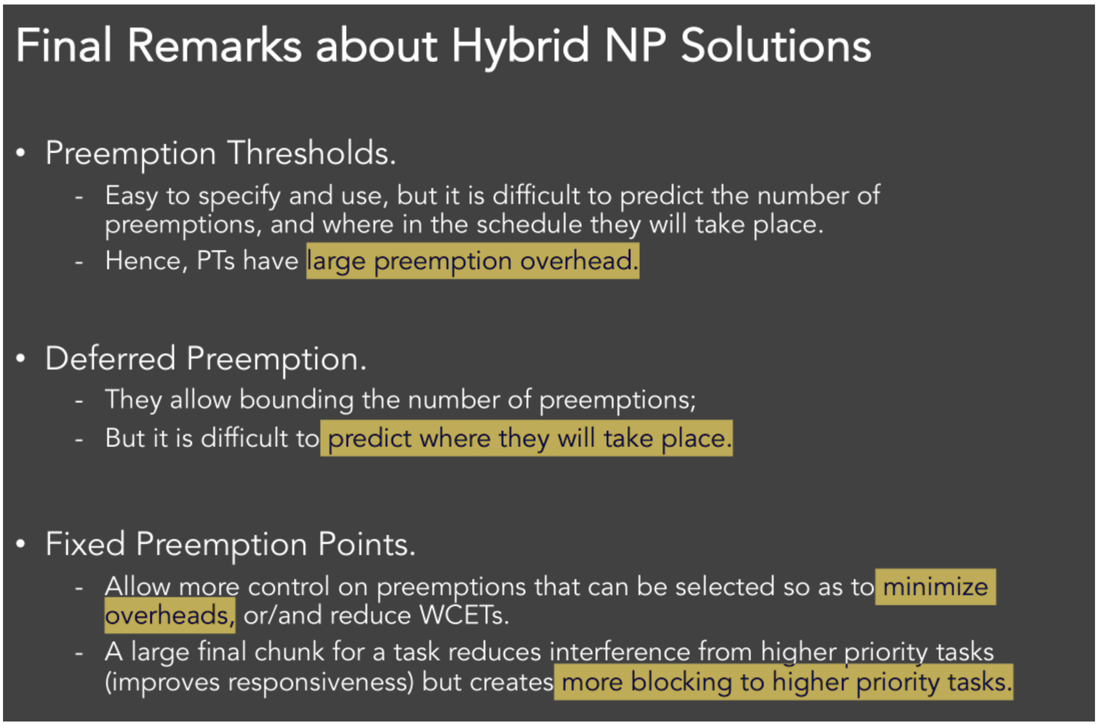
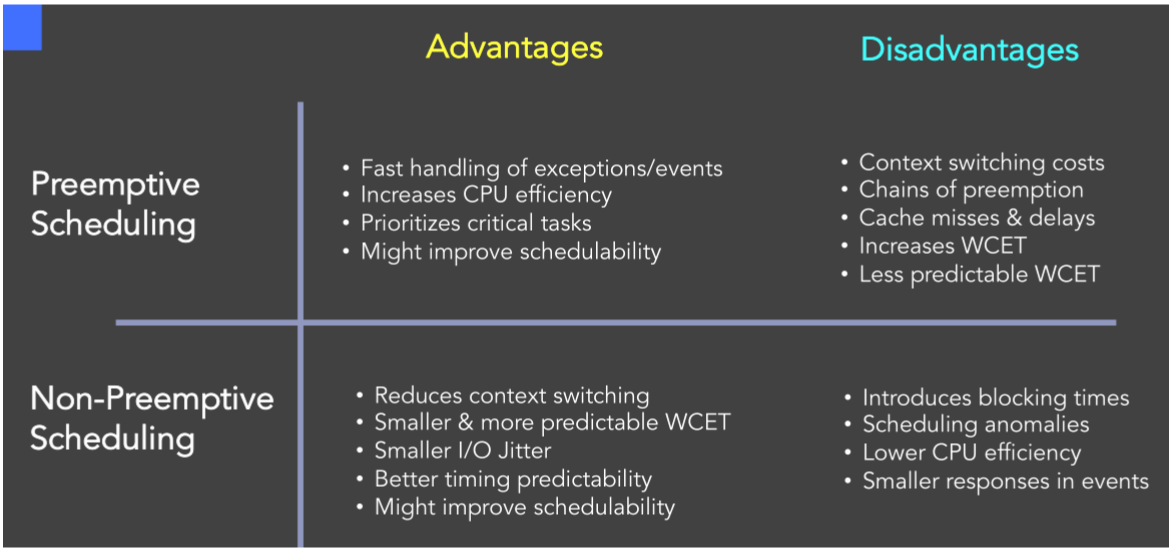
03_Non-Preemptive Scheduler



1. Preemption vs Non-Preemption



1.1. Disadvantages of Preemption

- introduces **context switching cost**
- One preemption might induce **more preemptions**
- ntroduces **cache-related preemption delays** (CPRD).
- increase the WCET (Due to the catche-related preemption delays)

1.2. Advantages of Non-preemption

- Reduce contex-swiching overhead
- Simplifies the access to shared resources
- Reduces the required stack size
- Allows achieving small I/O jitter
- in some cases, improves schedulability in fixed-priority systems

1.3. Disadvantages of Non-preemption

- Might reduce schedulability due to blocking
    - A task with higher priority needs to wait lower-priority tasks
- Give rise to scheduling anomalies
    - Unexpected and counter-intuitive system behaviour. (e.g. processor speed up)

2. Non-Preemptive (NP) scheduling

2.1. Assumtptions

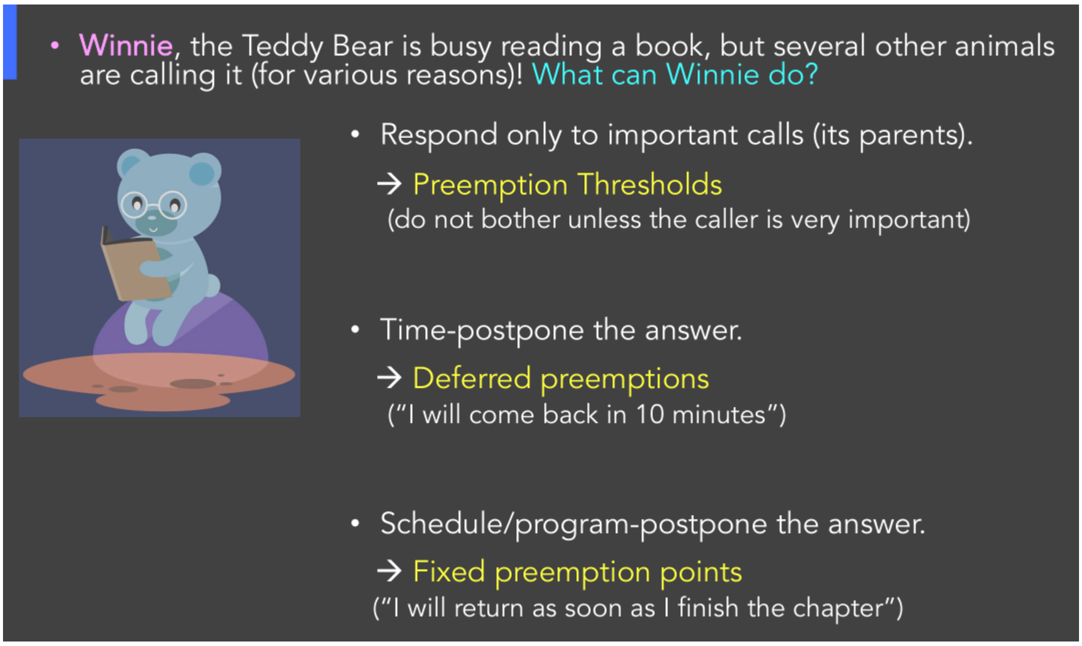
- consider a set of **hard RT periodic tasks**
- Constrained deadline model:

$$D_i \leq T_i, \forall i$$

- Tasks are assigned **fixed priorities** (RM,DM), and we always use

$$P_1 > P_2 > \cdots > P_n$$

2.2. Hybrid NP Solutions



Preemption Thresholds

Only tasks of **very high priority** can preempt low priority tasks.
For each task define a **different minimum priority threshold** that can preempt it.

Deferred Preemptions

Define for each task **the longest no-preemption time interval** for its execution.

Task Splitting Approach

Split the task to **non-preemptable code chunks**;
Preemption can take place only at specified points between those chunks.

2.3. Overal Property of NP-Scheduling Policies

- Different Critical Case:
    - Critiacal instant start in syn with higher priority jobs
- Self-pushing phenomenon
- The largest reponse time is not in the first execution after the critical instant
    - Response time analysis for Task i must be done for multiple periods

2.4. Analysis of NP Scheduling Policies

Level-i active period

The time interval in which the CPU is **busy executing Task i or other tasks with higher priority,** including also blocking times from lower priority tasks.

Formally, it's a timer interval $\Delta = [t_s, t_e)$, such that

$$P_i\left(t_s\right) = 0; \quad P_i\left(t_e\right) = 0; \quad P_i(t) > 0 \text{ for all } t \text{ in } \Delta$$

$P_i(t)$ is the **Processing load** that is pending at t, from all tasks with priority higher than i, including task i.

- no lower priority task is taken into consideration (because a higher priority pending means after blocked (if exist), no time for lower priority execute)

Calculate Level-i active period

Estimate the **worst-case blocking time** of Task i from lower-priority jobs.

$$B_i = \max_{j:P_j<P_i} \{C_j - 1\}$$

The busy period is **at least as large as** B_i plus the WCET of Task i:

$$L_i^{(0)} = B_i + C_i$$

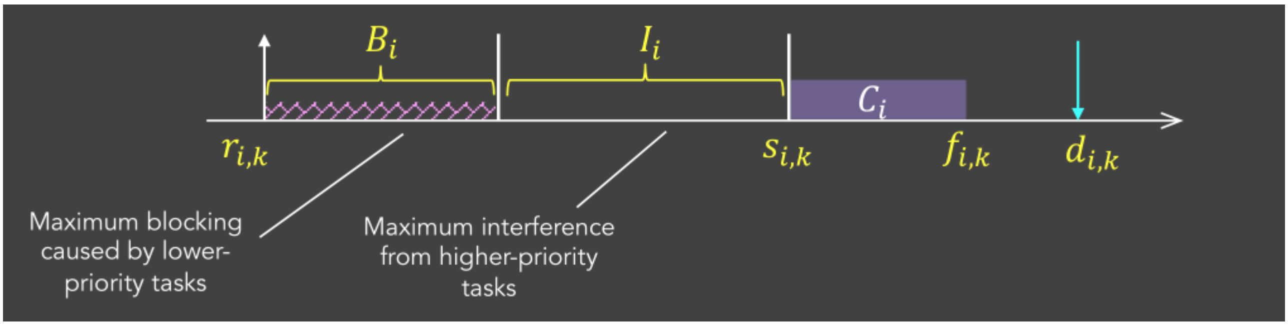
Find how many higher-priority jobs run in this interval

$$L_i^{(s)} = B_i + \sum_{h:P_h \geq P_i} \lceil \frac{L_i^{(s-1)}}{T_h} \rceil C_h \quad \text{until:} \quad L_i^{(s)} = L_i^{(s-1)}$$

the response time of Task i must be computed for k jobs:

$$k = 1, 2, \ldots, K = \lceil \frac{L_i}{T_i} \rceil$$

Analylsis of Schedulability



$$
\begin{aligned}
s_{i,k}^{(0)} &= B_i + \sum\nolimits_{h:P_h>P_i} C_h \\
s_{i,k}^{(l)} &= B_i + (k-1)C_i + \sum\nolimits_{h:P_h>P_i} \left(\left\lfloor \frac{s_{i,k}^{(l-1)}}{T_h} \right\rfloor + 1 \right) C_h
\end{aligned}
$$

B_i is the blocking

$(k-1)C_i$ is preceding jobs of the same task

$\sum\nolimits_{h:P_h>P_i} \left(\left\lfloor \frac{s_{i,k}^{(l-1)}}{T_h} \right\rfloor + 1 \right) C_h$ is the interference from higher priority tasks

$$
\begin{aligned}
f_{i,k} &= s_{i,k} + C_i \\
R_i &= \max\nolimits_{k\in[1,K_i]} \{f_{i,k} - (k-1)T_i\}
\end{aligned}
$$

The set is **schedulable** if and only if

$$R_i \leq D_i, i = 1, 2, \ldots, n$$

Special Case

It is sufficient to **consider only the first job if and only if** the task set is **feasible under preemptive scheduling** and all **deadlines are less (or equal) to the periods.**

$$
\begin{aligned}
&s_{i,1} = S_i = B_i + \sum\nolimits_{h:P_h>P_i} \left(\left\lfloor \frac{S_i}{T_h} \right\rfloor + 1 \right) C_h \\
&R_i = S_i + C_i
\end{aligned}
$$

3. Hybrid NP Solutions

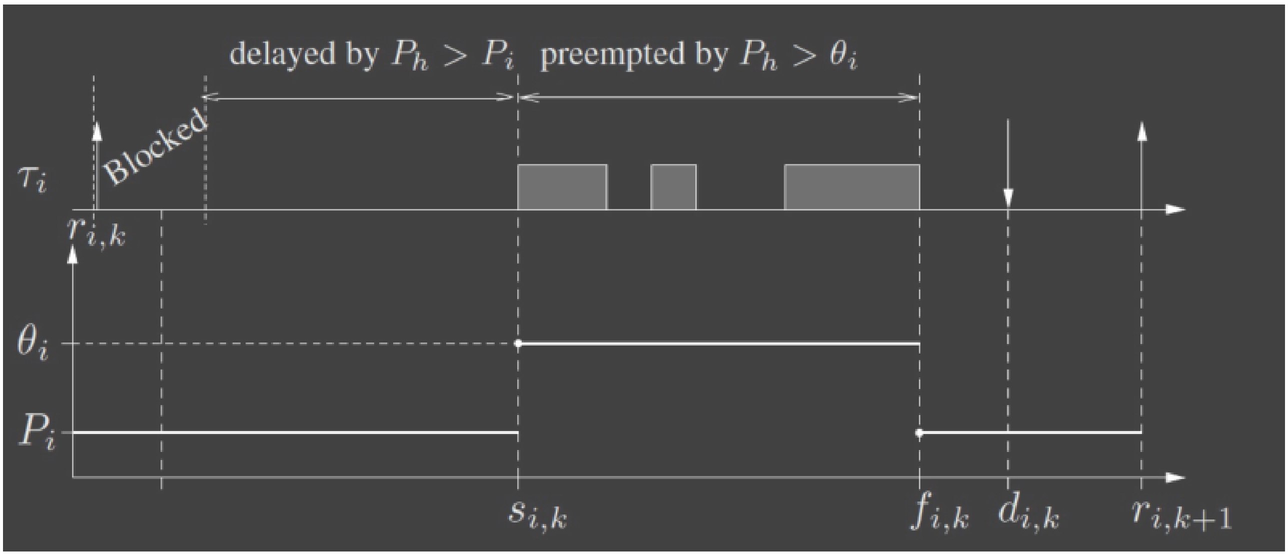
3.1. Preemption Thresholds

Each task i has two priorities:

- **Nominal Priority** P_i : used as long as the task is in the queue. Based on this value it **preempts (or not) other** tasks.
- **Threshold Priority** θ_i : used when the task runs. Only tasks with priority higher than this threshold can preempt task i.

In general $\theta_i > P_i$ (personal understanding)

Feasibility Analysis



Blocking time

$$B_i = \max_{j:P_j<P_i} \{C_j - 1 \mid P_j < P_i \leq \theta_j\}$$

Level-i active period and K

$$L_i^{(s)} = B_i + \sum_{h:P_h \geq P_i} \left\lceil \frac{L_i^{(s-1)}}{T_h} \right\rceil C_h$$
$$K = \left\lceil \frac{L_i}{T_i} \right\rceil$$

Start time of job k

$$s_{i,k}^{(l)} = B_i + (k-1)C_i + \sum_{h:P_h>P_i} \left(\left\lfloor \frac{s_{i,k}^{(l-1)}}{T_h} \right\rfloor + 1 \right) C_h$$

Finish time Analysis

$$f_{i,k}^{(0)} = s_{i,k} + C_i$$

$$f_{i,k}^{(l)} = s_{i,k} + C_i + \sum_{h:P_h>\theta_i} \left(\left\lceil \frac{f_{i,k}^{(l-1)}}{T_h} \right\rceil - \left(\left\lfloor \frac{S_{i,k}}{T_h} \right\rfloor + 1 \right) \right) C_h{}^{F}$$

Feasibility Check

$$R_i = \max_{k \in [1, K_i]} \{ f_{i,k} - (k-1) T_i \}$$

$$R_i \le D_i, i = 1, 2, \ldots, n$$

Notes:

For equation 13,14,15, use $P_h \ge P_i$ is because, these equations consider the situation that our task is not running. So when all in the queue, they are sorted by P not θ

Threshold Assign

Try from $\theta_i = P_i$, if not feasible, add one.

- The algorithm is **optimal**: If there is a feasible schedule, it will find it.

Algorithm: Assign Minimum Preemption Thresholds
Input: A task set $\mathcal{T}$ with $\{C_i, T_i, D_i, P_i\}, \forall \tau_i \in \mathcal{T}$
Output: Task set feasibility and $\theta_i, \forall \tau_i \in \mathcal{T}$
// Assumes tasks are ordered by decreasing priorities

$P_1 > P_2 > \cdots > P_n$

begin
    for ($i := n$ **to** 1) **do** *// from the lowest priority task*
        $\theta_i := P_i$;
        Compute R_i by Equation (8.15);
        while ($R_i > D_i$) **do** *// while not schedulable*
            $\theta_i := \theta_i + 1$; *// increase threshold*
            if ($\theta_i > P_1$) **then** *// system infeasible*
                return (INFEASIBLE);
            end
            Compute R_i by Equation (8.15);
        end
    end
    return (FEASIBLE);
end

$R_i = \max_{k \in [1, K_i]} \{ f_{i,k} - (k-1) T_i \}$

We reduce the tasks that can preempt Task i

The algorithm is optimal: If there is a feasible schedule, it will find it.

3.2. Deferred Preemptions

we define for each task an interval of time qi during which
it cannot be preempted

Modes

Floating Model

The non-preemption interval is explicitly defined in the code by the programmer.

Activation-triggered

The non-preemption interval is **triggered by the arrival of a higher priority task** -- which normally would preempt the current job -- and is enforced for q_i time slots

Analysis of DP scheduling

Blocking

Each Task i can be blocked by the non-preemptive subjobs of any lower-priority task.

- If floating model:

$$B_i = \max_{j:P_j<P_i} \{q_j - 1\}$$

- if activation-triggered model:

$$B_i = \max_{j:P_j<P_i} \{q_j\}$$

Others are the same with **NP scheduling**

Open-Discussions

Given a set of periodic tasks that is feasible under preemptive scheduling, what is the longest non-preemptive interval Qi for each task i, so that it can continue to execute for Qi time in non-preemptive mode, without violating the schedulability of the original task set?

3.3. Fixed Preemption Points

Each task i is split into m_i **chunks**.

- During those chunks (or, subjobs), the task cannot be preempted;
- Preemption can only happen in between these subjobs.

Assuming known WCET for each chunk, we can calculate:

$$C_i = \sum_{k=1}^{m_i} q_{ik}$$

Two important parameters

$$q_i^{\max} = \max_{k \in [1, m_i]} \{q_{ik}\}$$

Affects the feasibility of higher priority tasks (those that this subjob can preempt)

$$q_i^{\text{last}} = q_{i,m_i}$$

Affects the response time of Task i

Analylsis of TS scheduling

Blocking time:

$$B_i = \max_{j:P_j<P_i} \left\{q_j^{\max} - 1\right\} \quad q_i^{\text{last}} = q_{i,m_i}$$

Level-i active period

$$L_i^{(0)} = B_i + C_i$$

$$L_i^{(s)} = B_i + \sum_{n:P_h \geq P_i} \left\lceil \frac{L_i^{(s-1)}}{T_h} \right\rceil C_h$$

$$k = 1, \ldots, K = \left\lceil \frac{L_i}{T_i} \right\rceil$$

the starting times of the **last subjob of each job of each task** are:

$$s_{i,k}^{(0)} = B_i + C_i - q_i^{last} + \sum_{h:P_h > P_i} C_h$$

$$s_{i,k}^{(l)} = B_i + kC_i - q_i^{last} + \sum_{n:P_h > P_i} \left(\left\lfloor \frac{s_{i,k}^{(l-1)}}{T_h} \right\rfloor + 1 \right) C_h$$

Finish time and response time

$$f_{i,k} = s_{i,k} + q_i^{\text{last}}$$

$$R_i = \max_{k \in [1, K_i]} \{ f_{i,k} - (k-1)T_i \}$$

Feasibility Check

$$R_i \leq D_i, \quad i = 1, 2, \ldots, n$$

Special Case

If the task set is feasible under preemptive execution, then the analysis can be done by using only the first job of each task