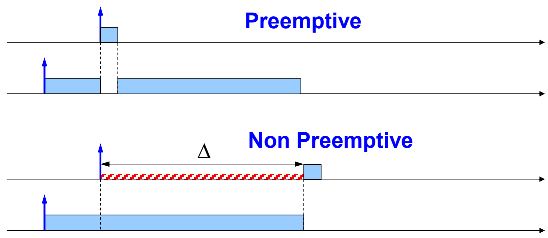
Limited Preemptive Scheduling

Preemptive scheduling

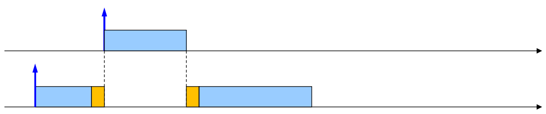
Most of work on scheduling has been focused on fully preemptive systems, because they allow higher responsiveness:



Disadvantages of preemptions

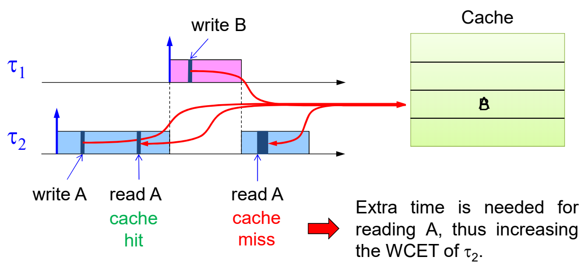
However, each preemption has a cost:

- Context switch cost: time taken by the scheduler to suspend the running task, switch the context, and dispatch the new incoming task.



Cache related preemption delay

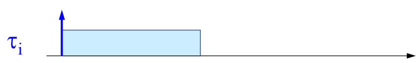
CRPD: delay introduced by high priority tasks that evict cache lines containing data used in the future:



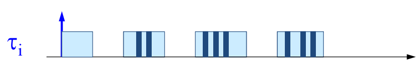
Extra time is needed for reading A, thus increasing the WCET of τ_2.

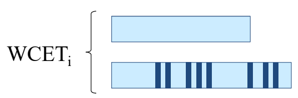
WCET

Task executing alone (or non preemptively) on a single CPU:



Task experiencing preemptions by higher priority tasks:



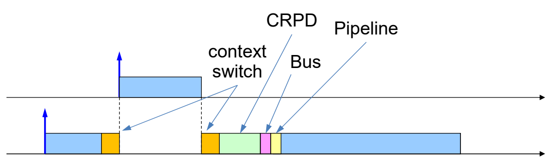


C_i^{NP}

$C_i = C_i^{NP} + \mathrm{CRPD}$

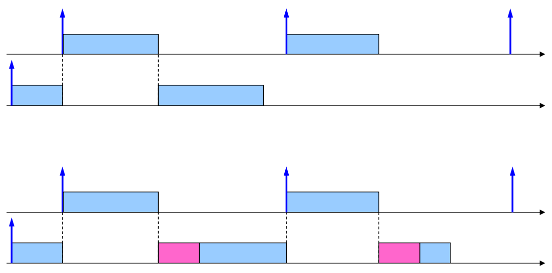
Preemption indirect costs

- Pipeline cost: time to flush the pipeline when a task is interrupted and to refill it when task is resumed.
- Bus cost: time spent waiting for the bus due to additional conflicts with I/O devices, caused by extra accesses to the RAM for the extra cache misses.



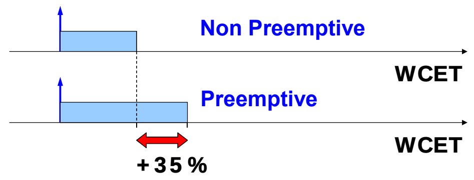
Preemption indirect costs

- Additional preemptions: the extra execution time also increases the number of preemptions:



Preemption cost

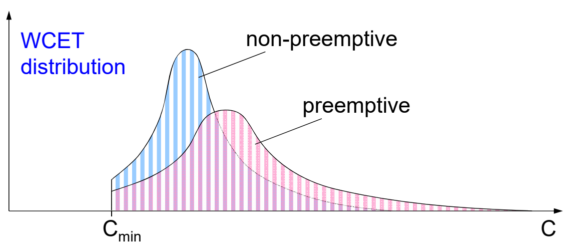
- WCETs may increase up to 35% in the presence of preemptions (less efficiency):



- WCETs become also more variable (less predictability):

Influence on WCETs

- As a consequence, WCETs estimations for preemptive tasks are
  - higher
  - less predictable (highly variable)



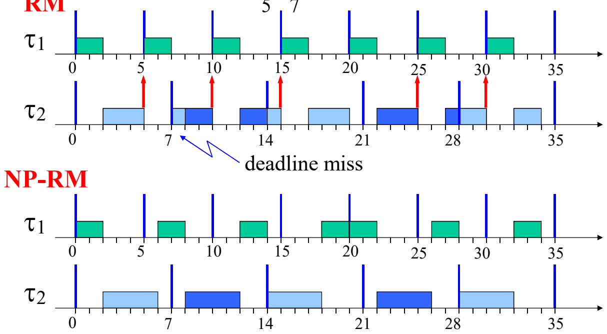
Advantages of NP scheduling

- It reduces context-switch overhead:
  - making WCETs smaller and more predictable.
- It simplifies the access to shared resources:
  - No semaphores are needed for critical sections
- It reduces stack size:
  - Task can share the same stack, since no more than one task can be in execution
- It allows achieving zero I/O Jitter:
  - finishing_time – start_time = C_i (constant)

Advantages of NP scheduling

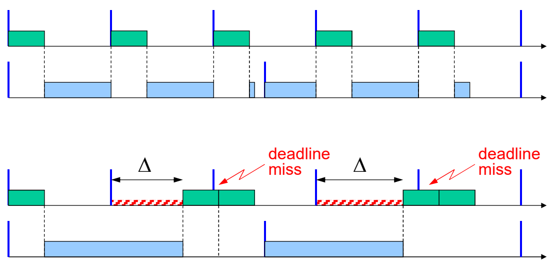
In fixed priority systems can improve schedulability:

$$U = \frac{2}{5} + \frac{4}{7} \cong 0.97$$



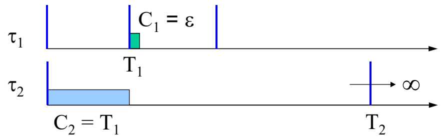
Disadvantages of NP scheduling

- In general, NP scheduling reduces schedulability introducing blocking delays in high priority tasks:



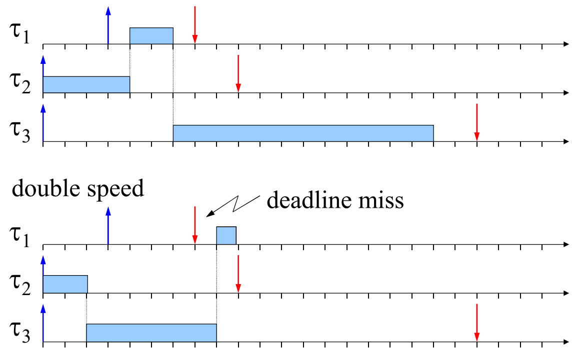
Disadvantages of NP scheduling

- The utilization bound under non preemptive scheduling drops to zero:



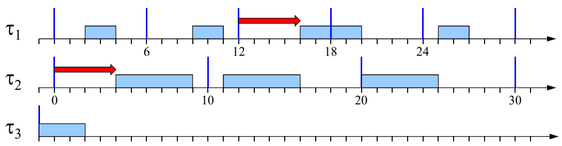
$$U = \frac{\varepsilon}{T_1} + \frac{C_2}{\infty} \longrightarrow 0$$

Non preemptive anomalies



Non-preemptive analysis

Analysis of non-preemptive systems is more complex, because the largest response time may not occur in the first job after the critical instant.

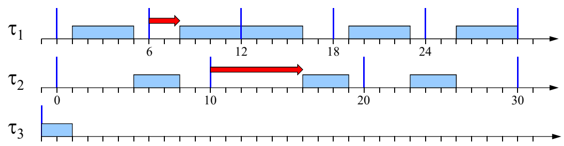


Self-pushing phenomenon

High priority jobs activated during non-preemptive execution of lower priority tasks are pushed ahead and introduce higher delays in subsequent jobs of the same task.

Non-preemptive analysis

Hence, the analysis of τ_i must be carried out for multiple jobs, until all tasks with priority $\geq P_i$ are completed.



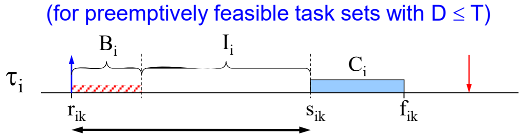
NOTE

Analysis can reduce to the first job of each task if and only if

1. the task set is feasible under preemptive scheduling;
2. All deadlines are less than or equal to periods.

Response time analysis

(for preemptively feasible task sets with D ≤ T)



$$WO_i = \max_k \{s_{ik} - r_{ik}\}$$

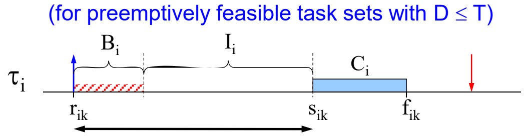
$$B_i = \max_{P_j < P_i} \{C_j\}$$

- ***hp(i)*** set of tasks with priority higher than P_i
- ***lp(i)*** set of tasks with priority lower than P_i

WO_i = Worst-case Occupied time: due to blocking B_i from ***lp(i)*** tasks and interference I_i from ***hp(i)*** tasks.

Response time analysis

(for preemptively feasible task sets with D ≤ T)



$$WO_i = \max_k \{s_{ik} - r_{ik}\}$$

NOTE: the end of I_i cannot coincide with the activation of a higher priority task, because it would increase I_i.

Hence $\lfloor x \rfloor + 1$ must be used instead of $\lceil x \rceil$

$$WO_i = B_i + \sum_{k=1}^{i-1} \left(\left\lfloor \frac{WO_i}{T_k} \right\rfloor + 1 \right) C_k$$

$$R_i = WO_i + C_i$$

Response time analysis

(for preemptively feasible task sets with D ≤ T)

$$\begin{cases} WO_i^{(0)} = B_i + \sum_{k=1}^{i-1} C_k \\ WO_i^{(s)} = B_i + \sum_{k=1}^{i-1} \left(\left\lfloor \frac{WO_i^{(s-1)}}{T_k} \right\rfloor + 1 \right) C_k \end{cases}$$

Stop when $WO_i^{(s)} = WO_i^{(s-1)}$

$$\boxed{R_i = WO_i + C_i}$$

19

Trade-off solutions

The following solutions can be adopted to balance between the two extreme approaches:

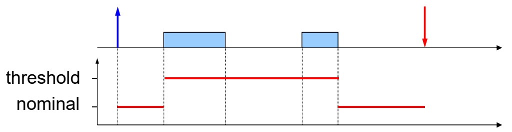
- ***Preemption Thresholds***
  - Allow preemption only to tasks with high importance
- ***Deferred Preemptions***
  - Allow preemption only after a given time interval
- ***Fixed Preemption Points***
  - Allow preemption only at given points in the task code

Preemption Thresholds (PT)

Each task has two priorities:

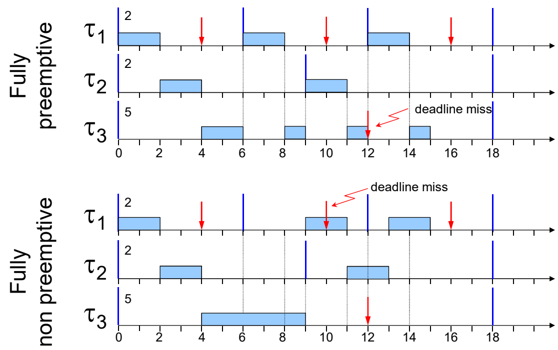
P_i nominal priority: used to enqueue the task in the ready queue and to preempt

θ_i threshold priority: used for task execution $(\theta_i \geq P_i)$



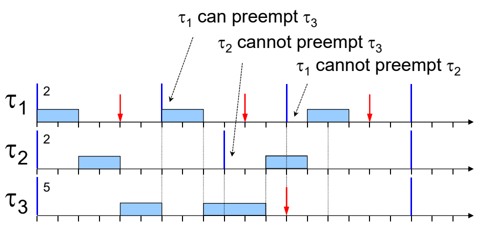
A task τ_i can be preempted by τ_h only if $P_h > \theta_i$

Unfeasible task set



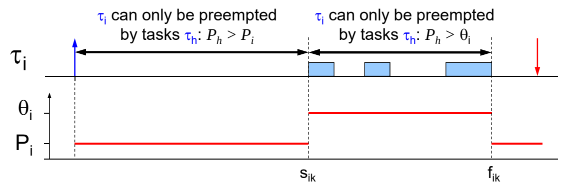
But feasible with PT

	P_i	θ_i
τ_1	3	3
τ_2	2	3
τ_3	1	2



NOTE:
The same feasible schedule is obtained by splitting τ_3 in two non preemptive chuncks: $q_{31} = 2$, $q_{32} = 3$

Response time analysis (PT)

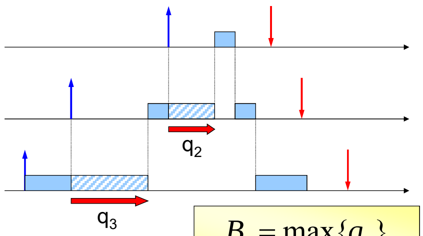


$$s_{ik} = B_i + (k-1)C_i + \sum_{h:P_h > P_i} \left(\left\lfloor \frac{s_{ik}}{T_h} \right\rfloor + 1 \right) C_h$$

$$f_{ik} = s_{ik} + C_i + \sum_{h:P_h > \theta_i} \left(\left\lceil \frac{f_{ik}}{T_h} \right\rceil - \left\lfloor \frac{s_{ik}}{T_h} \right\rfloor - 1 \right) C_h$$

Deferred Preemption

Each task can defer preemption up to q_i



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

Interesting problem

Given a preemptively feasible task set, find the longest non-preemptive interval Q_i for each task that still preserves schedulability.

- Under EDF ➡ Baruah - ECRTS 2005
- Under Fix. Pr. ➡ Gang-Buttazzo, RTCSA 2009

Often, high priority tasks have $Q_i = C_i$, meaning that they can execute fully non preemptively.

Blocking tolerance

To compute Q_i, we need to find the maximum blocking time that can be tolerated by a task, called blocking tolerance (β_i):

$$\left(\sum_{k=1}^{i} \frac{C_k}{T_k}\right) + \frac{B_i}{T_i} \le U_{\text{lub}}(i)$$

$$\beta_i = T_i \left[U_{\text{lub}}(i) - \sum_{k=1}^{i} U_k \right]$$

A simple bound for Q_i

The longest non preemptive interval Q_i is related with the maximum blocking time β_i that can be tolerated by higher priority tasks.

It must be $\quad \forall i \quad B_i \le \beta_i$

where $\quad B_i = \max_{P_j < P_i} \{q_j\}$

Hence: $\quad \forall i \quad \max_{P_j < P_i} \{q_j\} \le \beta_i$

A simple bound for Q_i

$$\forall i \quad \max_{P_j < P_i} \{q_j\} \le \beta_i$$

$$\begin{cases} i = 1 & \max\{q_2, q_3, q_4\} \le \beta_1 \\ i = 2 & \max\{q_3, q_4\} \le \beta_2 \\ i = 3 & q_4 \le \beta_3 \end{cases}$$

$$\begin{cases} i = 1 & q_2 \le \beta_1 \\ i = 2 & q_3 \le \min\{\beta_1, \beta_2\} \\ i = 3 & q_4 \le \min\{\beta_1, \beta_2, \beta_3\} \end{cases}$$

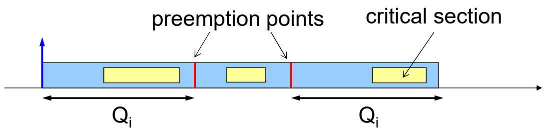
A simple bound for Q_i

$$\begin{aligned} &i = 1 \quad && Q_1 = \infty \\ &i \ge 2 \quad && Q_i = \min\{Q_{i-1}, \beta_{i-1}\} \end{aligned}$$

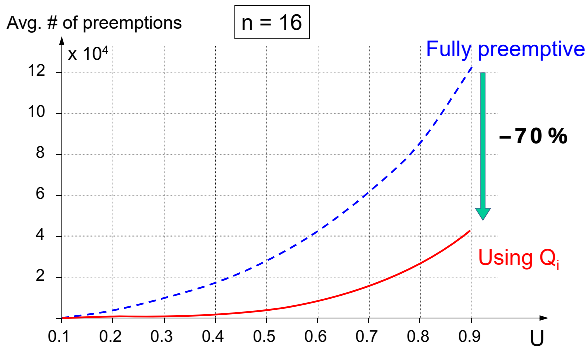
Using Q_i

Once Q_i is computed, it can be used as follows:

- Partition each task into a set of NP regions no larger than Q_i inserting suitable preemption points.
- Incapsulate critical sections into NP regions, avoiding complex concurrency control protocols.

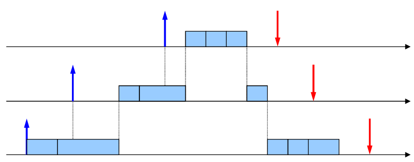


Experimental Results



Fixed Preemption Points (FPP)

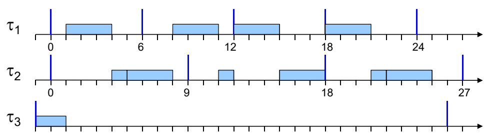
- Each task τ_i is divided in m_i chunks: $q_{i,1}$ ... q_{i,m_i}
- It can only be preempted between chunks



$$B_i = \max_{P_j < P_i} \{ q_j^{\max} \}$$

Example

Let: τ_1 be fully non preemptive: $q_{11} = C_1 = 3$
τ_2 consisting of 2 NP chunks: $q_{21} = 1$, $q_{22} = 3$, $C_2 = 4$
τ_3 be fully non preemptive: $q_{31} = C_3 = 2$

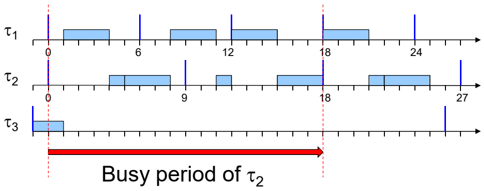


Note that:

- The worst case response time of τ_2 does not occur in the first instance.
- The interference on τ_2 is larger than $B_2 + C_1$.

Response Time Analysis (FPP)

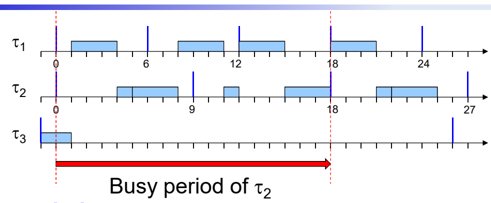
- Must be carry out up to the busy period of each task.



Level-i busy period

It is the interval in which the processor is busy executing tasks with priority higher than or equal to P_i, including blocking times.

Response Time Analysis (FPP)



Level-i busy period

It can be computed as the shortest interval that satisfies:

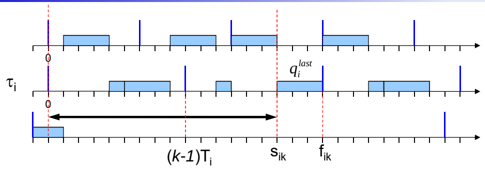
$$L_i = B_i + \sum_{h:P_h \geq P_i} \left\lceil \frac{L_i}{T_h} \right\rceil C_h$$

up to job N_i:

$$N_i = \left\lceil \frac{L_i}{T_i} \right\rceil$$

Initial value can be: $L_i^{(0)} = B_i + \sum_{h:P_h \geq P_i} C_h$

Response Time Analysis (FPP)



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

$$f_{ik} = s_{ik} + q_i^{last}$$

$$R_{ik} = f_{ik} - (k-1)T_i$$

$$R_i = \max_{k\in[1,N_i]}\{R_{ik}\}$$

NOTE:
$$\begin{cases} s_{ik}^{(0)} = (k-1)T_i + \left(C_i - q_i^{last}\right) \\ N_i = \left\lceil \dfrac{L_i}{T_i} \right\rceil \end{cases}$$

Response Time Analysis (FPP)

for $(i=1$ to $n)$ {
  $N_i = \lceil L_i / T_i \rceil$
  $s_{ik}^{(0)} = (k-1)T_i + \left(C_i - q_i^{last}\right)$
  $k = 1$
  do {
    $s_{ik} = B_i + (k-1)C_i + \left(C_i - q_i^{last}\right) + \sum_{h:P_h>P_i}\left(\left\lfloor\frac{s_{ik}}{T_h}\right\rfloor + 1\right)C_h$
    $f_{ik} = s_{ik} + q_i^{last}$
    $R_{ik} = f_{ik} - (k-1)T_i$
    if $(R_{ik} > R_i)$ **then** $R_i = R_{ik}$
    if $(R_i > D_i)$ **then** *return(UNFEASIBLE)*
    $k{+}{+}$
  } **while** $(k \le N_i)$
}
return(FEASIBLE)

Special cases

- **Fully non preemptive scheduling**

$$\begin{cases} q_i^{last} = C_i \\ B_i = \max\limits_{P_j<P_i}\{C_j\} \end{cases}$$

- **Deferred Preemption**

$$\begin{cases} q_i^{last} = 0 \\ B_i = \max\limits_{P_j<P_i}\{Q_j\} \end{cases}$$

Final remarks

- Preemption Thresholds are easy to specify, but it is difficult to predict the number of preemptions and where they occur $\Rightarrow$ large preemption overhead
- Deferred Preemption allows bounding the number of preemptions but it is difficult to predict where they occur. Note that the analysis assumes $q_i^{last} = 0$
- Fixed Preemption Points allow more control on preemptions and can be selected on purpose (e.g., to minimize overhead, stack size, and reduce WCETs).
  - A large final chunk in τ_i reduces the interference from hp-tasks (hence R_i), but creates more blocking to hp-tasks.