# Schedulability Analysis for Dual Priority Scheduling

## I. Dual Priority

Given a task system $\tau = \{\tau_1, \tau_2, \ldots, \tau_n\}$, we first make the following **assumptions**:

1) Each task $\tau_i$ has a original priority $n + i$ and a promotion priority $i$.
2) Each task has a fixed promotion point $p_i$. The concerned job $J_{i,k}$ has its promotion point $P_i = r_{i,k} + p_i$.

*Lemma 1 (Worst Case Pattern):* Suppose $\tau$ could experience deadline miss with the current priority and promotion point assignment, then it must be that there exists a busy time interval $[0, t]$ during which some job of task $\tau_i \in \tau$ would miss its deadline when all other tasks release their first job at time instant $0$, and all jobs including those released by $\tau_i$ are released as soon as possible with corresponding period.

*Proof 1:* Among all possible legal event sequences in which $\tau$ misses some deadline, let $S$ denote such a sequence where some job of $\tau_i \in \tau$ misses its deadline within the shortest busy interval $[0, t_F]$. As a result no deadline miss would happens earlier than $t_F$.

Assuming in the event sequence $S$, there exists some tasks, e.g., $\tau_j$ whose first release is not at time instant $0$ and the separation between some job release is greater than $T_j$. Then as we shift $r_{j,1}$ to $0$ and reduce time separation between each job releases to $T_j$, $\tau_i$ would consume no more resource than before.

This is because that all jobs of $\tau_j$ with deadline before $t$ would still meet its deadline (otherwise we can construct a new sequence $S$), while the last job of $\tau_j$ released before $t$ (e.g., $J_{j,m}$) is more likely to deny processor from $\tau_i$ since the promotion point of job $J_{j,m}$ will also decrease. Therefore after we modify the release pattern of $\tau_j$, $\tau_i$ would still misses its deadline. Finally by repeating the above steps for all such tasks, the deadline miss of $\tau_i$ would still happen at $t_F$. Therefore Lemma 1 is true.

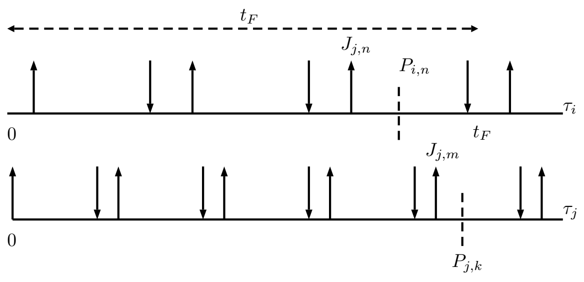

Fig. 1. Worst Case Pattern

From Lemma **??** we know that if $\tau$ would experience some deadline miss, then there exists a $\tau_i$ that will have its deadline miss with the worst case release pattern defined in Lemma **??**. Therefore the following corollary can tell us whether a task system $\tau$ is schedulable by the dual priority scheduling algorithm.

*Corollary 1:* If $\forall \tau_i \in \tau$ no deadline miss happens for all possible time interval length $t$ with the worst case release pattern: 1) some job of $\tau_i$ has deadline at $t$ and all jobs of $\tau_i$ are released as soon as possible with period $T_i$; 2) first job of $\tau_j \in \tau - \tau_i$ is released at $0$ and all jobs are released as soon as possible with period $T_j$, then $\tau$ is schedulable by the dual priority scheduling algorithm.

*Proof 2:* This corollary is a direct deduction from Lemma **??**.

From Corollary **??**, we know that as along as we can guarantee that $\forall\ \tau_i \in \tau : \ \forall\ t :$ no deadline miss happens with the worst case release pattern, then we can declare that the task system $\tau$ is schedulable by the dual priority scheduling algorithm. Therefore in the simplest case, an exact but computational expensive schedulability test for dual priority scheduling algorithm could be derived by simulating the behavior of the system with the worst case release pattern.

To reduce the complexity, we need a more efficient test to determine whether $\tau_i \in \tau$ is schedulable when jobs are released with the worst case release in Lemma **??**. Let $ibf_i(\tau_j, t')$ (ibf is short for interference bound function) denote maximum possible resource consumed by $\tau_j \in \tau - \tau_j$ in the system during the time interval $[0, t']$, and let $dbf(\tau_i, t)$ denotes the maximum execution requirement of $\tau_i$ during $[0, t]$, when all tasks are released with worst case pattern in Lemma **??**.[1] Therefore the following theorem can be used to determine whether $\tau_i \in \tau$ is schedulable when all jobs in the system is released in the worst case release pattern.

*Theorem 1:* $\tau_i$ would not experience any deadline miss with the worst case release pattern on a single processor by the dual priority scheduling algorithm if the following condition holds:

$$\forall\, t\ : \exists\, t' \in [t{-}D_i, t] : \ dbf(\tau_i, t){+}ibf_i(\tau{-}\tau_i, t') \leq t' \tag{1}$$

where

$$ibf_i(\tau - \tau_i, t') = \sum_{\tau_j \in \tau - \tau_i} ibf_i(\tau_j, t')$$

.

*Proof 3:* We can prove the statement that if $\tau_i$ is not schedulable with the worst case release pattern, then Equation **??** would not hold. Suppose that $\tau_i$ is not schedulable then there are legal event sequences in which $\tau_i$ misses some deadline when $\tau_i$ is assigned with the current priority and promotion point. Let $S'$ denote such a sequence where $\tau_i$ misses deadline at the earliest time at $t$.

It must be the cases that during $[0, t]$ some tasks are executing because otherwise the time interval between the last idle instant to $t$ could also construct such a sequence. Then it must be that during the time interval $[0, t']$, other tasks have consumed an amount of resource more than

$$ibf_i(\tau - \tau_i, t') > t' - dbf(\tau_i, t)$$

which contradicts the Equation **??**.
With Theorem **??**, we can use the following test to whether a task system $\tau$ is schedulable by the dual priority scheduling algorithm.

*Theorem 2:* A task system $\tau$ is schedulable by the dual priority scheduling on unit-speed uniprocessor if the following hold: $\forall \tau_i \in \tau : \ \forall\ t : \ \exists\ t' \in [t - D_i, t]$ :

$$dbf(\tau_i, t) + ibf_i(\tau - \tau_i, t') \leq t' \tag{2}$$

[1]Note that the two functions are specific for the worst case pattern.

## II. Interference Bound Function

In the worst case release pattern, we can easily calculate the exact value of $dbf(\tau_i, t)$. However the exact value of $ibf_i(\tau - \tau_i, t')$ is not trivial except we simulate the system. As a result, here we need to add some pessimism to bound the actual value of $ibf_i(\tau_j, t')$ instead. We will also apply an optimization technique to further improve the performance of the test because the resource consumed by $\tau$ before $r_{i,x}$ or $P_{i,x}$ can not be greater than $r_{i,x}$ or $P_{i,x}$, respectively. Thus here we also derive $ibf_i^1(\tau_j, t')$ and $ibf_i^2(\tau_j, t')$ which denote the maximum possible resource consumed by $\tau_j$ after $r_{i,x}$ and $P_{i,x}$, respectively.

In the worst case pattern, given a time interval length $t$, we can easily calculate the release time and promotion point of each job. Here we first present some notations that will be used later in the paper.

TABLE I
Notations

| | | |
|---|---|---|
| $P_{i,x} = t - D_i + p_i$ | $r_{i,x} = P_{i,x} - p_i$ | $n_j = \lfloor \frac{t'}{T_j} \rfloor$ |
| $r_{j,y} = n_j \times T_j$ | $P_{j,y} = r_{j,y} + p_j$ | $[a]_0 = \max(a, 0)$ |
| $n'_j = \lfloor \frac{r_i}{T_j} \rfloor$ | $r_{j,y'} = n'_j \times T_j$ | $[a,b]_s = \min(a, b)$ |
| $n''_j = \lfloor \frac{P_{i,x}}{T_j} \rfloor$ | $r_{j,y''} = n''_j \times T_j$ | $[a,b]_g = \max(a, b)$ |

Suppose $J_{j,y}$ is the job that has its release time $r_{i,x} \leq t' \leq r_{i,x} + T_j$, then we only need to figure out how much resource $J_{j,y}$ has consumed by $t'$. This is because that the $n_j$ previous jobs released before $r_{j,y}$ are assumed to meet their deadlines, and hence are supposed to consume $n_j.C_j$ time units of resource.

### A. $ibf_i(\tau_j, t')$ when $j < i$

We first consider the case when index $j$ is smaller than index $i$ which means $\tau_j$ has higher origin priority than $\tau_i$. Thus if $J_{j,y}$ is still active, $J_{i,x}$ would not execute unless it is during the interval $[P_{i,x}, P_{j,y}]$ (if $P_{i,x} < P_{j,y}$).

**Case 1** ( $P_{j,y} \leq P_{i,x}$) as shown in Figure **??**: Job $J_{i,x}$ always has lower priority than $J_{j,y}$. Thus the maximum possible resource consumed by $\tau_j$ before $t'$ is bounded by

$$ibf_i(\tau_j, t') = n_j.C_j + \min\left(C_j, t' - r_{j,y}\right)$$

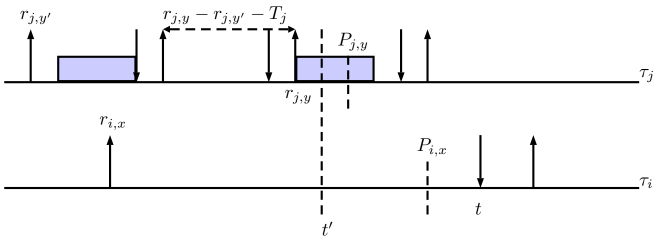

Fig. 2. $P_{j,y} \leq P_{i,x}$

The resource $\tau_j$ consumes after $r_{i,x}$ or $P_{i,x}$ is maximized when its execution happens as late as possible as shown in Figure **??**, and hence we have

$$ibf_i^1(\tau_j, t') = \begin{cases} \min(C_j, t' - r_{i,x}) & \text{if } r_{j,y} \leq r_{i,x} \\ \min(C_j, t' - r_{j,y}) + \min(C_j, [r_{j,y'} + D_j - r_{i,x}]_0) \\ + \frac{r_{j,y} - r_{j,y'} - T_j}{T_j} C_j & \text{if } r_{j,y} > r_{i,x} \end{cases}$$

Its maximum possible execution after $P_{i,x}$ is

$$ibf_i^2(\tau_j, t') = \begin{cases} \min\{C_j, [t' - P_{i,x}]_0\} & \text{if } r_{j,y} \leq P_{i,x} \\ \min\{C_j, t' - r_{j,y}\} + \frac{r_{j,y} - r_{j,y''} - T_j}{T_j} C_j + \\ \min(C_j, [r_{j,y''} + D_j - P_{i,x}]_0) & \text{if } r_{j,y} > P_{i,x} \end{cases}$$

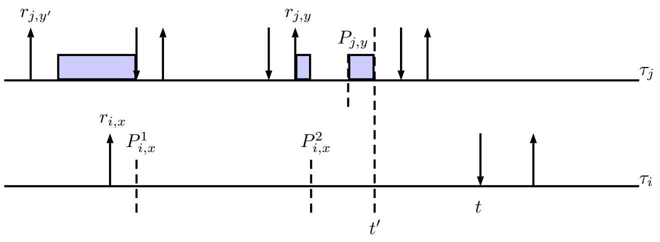

Fig. 3. $P_{j,y} > P_{i,x}$

**Case 2** ($P_{j,y} > P_{i,x}$): as shown in Figure **??**, $J_{i,x}$ has higher priority than $J_{j,y}$'s during $[\max(r_{j,y}, P_{i,x}), P_j]$ (if $P_i \leq P_j$). Thus $J_{j,y}$ can not execute during $[\max(r_{j,y}, P_{i,x}), P_j]$ unless $J_{i,x}$ has already finished. Thus we have

$$ibf_i(\tau_j, t') = n_j.C_j + \min(C_j, t' - r_{j,y} - (\min(t', P_{j,y}) - \max(r_{j,y}, P_{i,x})))$$

Its maximum possible execution after $r_{i,x}$ is

$$ibf_i^1(\tau_j, t') = \begin{cases} \min(C_j, \min(t', P_{i,x}) - r_{i,x}) \\ \quad \text{if } t' < P_{j,y} \wedge r_{j,y} \leq r_{i,x} \\ \min(C_j, \min(t', \max(r_{j,y}, P_{i,x})) - r_{j,y}) + \\ \frac{r_{j,y} - r_{j,y'} - T_j}{T_j} C_j + \min(C_j, [r_{j,y'} + D_j - r_i]_0) \\ \quad \text{if } t' < P_{j,y} \wedge r_{j,y} > r_{i,x} \\ \min\{C_j, t' - r_{i,x} - (P_{j,y} - P_{i,x})\} \\ \quad \text{if } t' \geq P_{j,y} \wedge r_{j,y} \leq r_{i,x} \\ \min\{C_j, t' - r_{j,y} - (P_{j,y} - \max(r_{j,y}, P_{i,x}))\} \\ + \frac{r_{j,y} - r_{j,y'} - T_j}{T_j} C_j + \min(C_j, [r_{j,y'} + D_j - r_{i,x}]_0) \\ \quad \text{if } t' \geq P_{j,y} \wedge r_{j,y} > r_{i,x} \end{cases}$$

Its maximum possible execution after $P_{i,x}$ is

$$ibf_i^2(\tau_j, t') = \begin{cases} 0 & \text{if } t' < P_{j,y} \wedge r_{j,y} \leq P_{i,x} \\ \frac{r_{j,y} - r_{j,y''} - T_j}{T_j} C_j + \min(C_j, [r_{j,y''} + D_j - P_{i,x}]_0) \\ & \text{if } t' < P_{j,y} \wedge r_{j,y} > P_{i,x} \\ \min\{t' - P_{j,y}, C_j\} & \text{if } t' \geq P_{j,y} \wedge r_{j,y} \leq P_{i,x} \\ \min\{t' - P_{j,y}, C_j\} + \min(C_j, [r_{j,y''} + D_j - P_{i,x}]_0) \\ + \frac{r_{j,y} - r_{j,y''} - T_j}{T_j} C_j & \text{if } t' \geq P_{j,y} \wedge r_{j,y} > P_{i,x} \end{cases}$$

### *B. $ibf_i(\tau_j, t')$ when $j > i$*

**Case 1.1** ($P_{i,x} \leq P_{j,y} \wedge r_{j,y} \leq r_{i,x}$): as shown in Figure **??**, $\tau_j$ may execute during $[r_{j,y}, r_{i,x}]$, but $\tau_j$ would not execute after $r_{i,x}$ unless $\tau_i$ has finished.

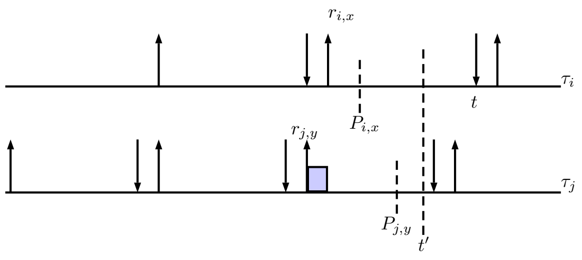

Fig. 4. $P_i \leq P_j \wedge r_j \leq r_i$

$$ibf_i(\tau_j, t') = n_j \times C_j + \min(C_j, r_{i,x} - r_{j,y})$$

However $J_{j,y}$ would never execute after $r_{i,x}$, and hence we have

$$ibf_i^1(\tau_j, t') = \min(C_j, r_{i,x} - r_{j,y})$$

$$ibf_i^2(\tau_j, t') = 0$$

**Case 1.2** ($P_{i,x} \leq P_{j,y} \wedge r_{j,y} > r_{i,x}$): as shown in Figure **??**, $J_{j,y}$ would not execute unless $\tau_i$ has completed. Thus we have

$$ibf_i(\tau_j, t') = n_j \times C_j$$

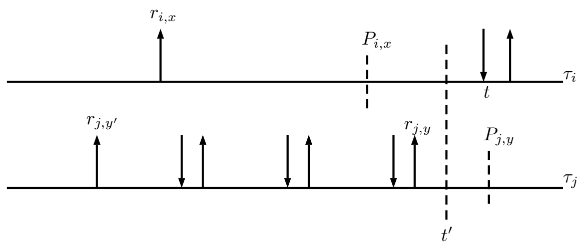

Fig. 5. $P_{i,x} \leq P_{j,y} \wedge r_{j,y} > r_{i,x}$

$\tau_j$ would not execute $P_{i,x}$ unless $J_{i,x}$ has already finished. Therefore we have

$$ibf_i^2(\tau_j, t') = 0$$

To maximize the possible execution of $\tau_j$ after $r_{i,x}$, we assume $\tau_j$ executes as late as possible. Meanwhile $J_{j,y-1}$ can not execute after $r_{j,y-1} + D_j$ or $P_{i,x}$ (if $P_{j,y-1} < P_{i,x}$) unless $J_{i,x}$ completes.

$$ibf_i^1(\tau_j, t') = \frac{r_{j,y} - r_{j,y'} - T_j}{T_j} C_j + \min(C_j, [r_{j,y'} + D_j - r_{i,x}]_0)$$

**Case 2.1** ($P_{i,x} > P_{j,y} \wedge r_{j,y} \leq r_{i,x}$): as shown in Figure **??**, the last job of $\tau_j$ can execute until $\min(t', P_i)$

$$ibf_i(\tau_j, t') = n_j \times C_j + \min\left(C_j, r_{i,x} - r_{j,y} + [\min(t', P_{i,x}) - \max(P_{j,y}, r_{i,x})]_0\right)$$

$$ibf_i^1(\tau_j, t') = \min\{C_j, [\min(t', P_{i,x}) - \max(P_{j,y}, r_{i,x})]_0\}$$

$$rbf_i^2(\tau_j, t') = 0$$

**Case 2.2** ($P_{i,x} > P_{j,y} \wedge r_{j,y} > r_{i,x}$): as shown in Figure **??**, $J_{j,y}$ would only execute during $[P_{j,y}, P_{i,x}]$ before $J_{i,x}$ finishes. Thus we have

$$ibf_i(\tau_j, t') = n_j.C_j + \min\left(C_j, [\min(t', P_{i,x}) - P_{j,y}]_0\right)$$

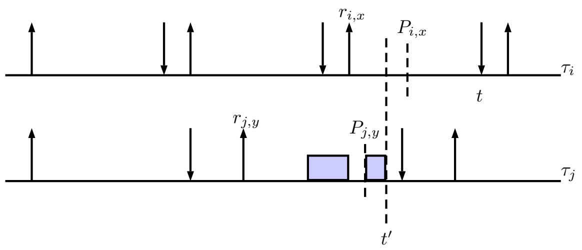

Fig. 6. $P_{i,x} > P_{j,y} \wedge r_{j,y} \leq r_{i,x}$

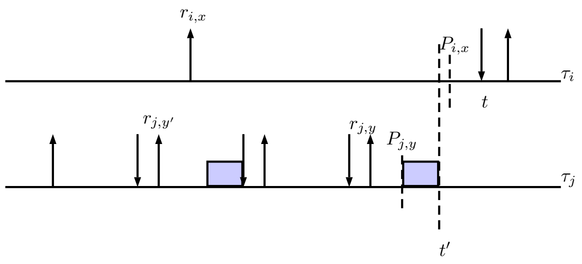

Fig. 7. $P_{i,x} > P_{j,y} \wedge r_{j,y} > r_{i,x}$

$$ibf_i^1(\tau_j, t') = \min(C_j, [r_{j,y'} + D_j - r_{i,x}]_0) + \min\left(C_j, [\min(t', P_{i,x}) - P_{j,y}]_0\right) + \frac{r_{j,y} - r_{j,y'} - T_j}{T_j} C_j$$

$$rbf_i^2(\tau_j, t') = 0$$

## III. Optimization Technique

We can bound the total execution before $r_i$ or $P_i$ (if $t' > P_i$) by $r_i$ or $P_i$, respectively. For $\tau_i$ itself, we simply assume its execution demand after $r_i$ and $P_i$ is $C_j$.

$$F_1(\tau_i, t, t') = \min\Bigg(r_i, n_i \times C_i + \sum_{\tau_j \in \{\tau - \tau_i\}} rbf_i(\tau_j, t') - rbf_i^1( \tag{3}$$

$$F_2(\tau_i, t, t') = \min\Bigg(P_i, n_i \times C_i + \sum_{\tau_j \in \{\tau - \tau_i\}} rbf_i(\tau_j, t') - rbf_i^2( \tag{4}$$

$$F(\tau_i, t, t') = \begin{cases} F_1(\tau_i, t, t') & \text{if } t' <= P_i \\ \min\{F_1(\tau_i, t, t'), F_2(\tau_i, t, t')\} & \text{otherwise} \end{cases} \tag{5}$$

We can derive a simple upper bound of $t$. Suppose that

$$\forall\, t' \in (k.T_i, k.T_i{+}D_i] : \; F(\tau_i, t, t') > t' \Rightarrow \min_{t' \in (k.T_i, k.T_i+D_i]} \frac{F(\tau_i, t, t')}{t'} > 1$$

, and let

$$H_i(t) = U{\times}t{+}\sum_{\tau_j \in \{\tau - \tau_i\}} C_j \geq t{\times}u_i{+}(\lfloor \frac{t'}{T_j} \rfloor{+}1)C_j \geq F(\tau_i, t, t')$$

then it must be that

$$\frac{H_i(t)}{t - D_i} > 1 \Rightarrow t - D_i < U \times t + \sum_{\tau_j \in \{\tau - \tau_i\}} C_j \Rightarrow t < \frac{D_i + \sum_{\tau_j \in \{\tau - \tau_i\}} C_j}{1 - U}$$