Schedulability Analysis for Dual Priority Scheduling

According to Baruah (2003, Dynamic- and Static-priority Scheduling of Recurring Real-time Tasks, Theorem 3), τ_i is schedulable on a single processor using static priority if and only if for each absolute deadline of a job $d_{i,k}$ where $k \in N$, there exists an interval $d_{i,k-1} \le t' \le d_{i,k}$ for which the following condition holds:

$$dbf(\tau_i, t) + \sum_{\tau_j \in \{hp_i\}} rbf_i(\tau_j, t') \le t' \tag{1}$$

Here the demand bound function captures the maximum execution demand of τ_i for a time interval length t if it is to meet all deadlines.

$$dbf(\tau_i, t) = \left(\lfloor \frac{t - D_i}{T_i} \rfloor + 1 \right) \times C_i \tag{2}$$

On the other hand, the request bound function $rbf_i(\tau_j, t')$ denotes the maximum amount of time for which τ_j could **deny the processor** to lower priority task τ_i over some interval length of t'.

I. Dual Priority

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

1) Each task τ_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 $P_i = r_{i,k} + p_i$.

Existing request bound function $rbf_i(\tau_j, t')$ denotes the maximum cumulative execution requirement by jobs of τ_j by t' that have ready times within any time interval of duration t'. However this is not the case in dual priority scheduling as shown in the following example.

Example 1: As shown in the Figure 1, when $r_i \le t' \le P_i$, the execution requirement of $J_{j,k}$ is C_j because τ_i would execute only after $J_{j,k}$ finishes. However, when $P_i \le t' \le P_j$, the time $J_{j,k}$ deny processor from τ_i is equal the time that $J_{j,k}$ has already executed, because τ_i has higher priority than τ_j during $[P_i, P_j]$. This is a significantly different from the static priority scheduling.

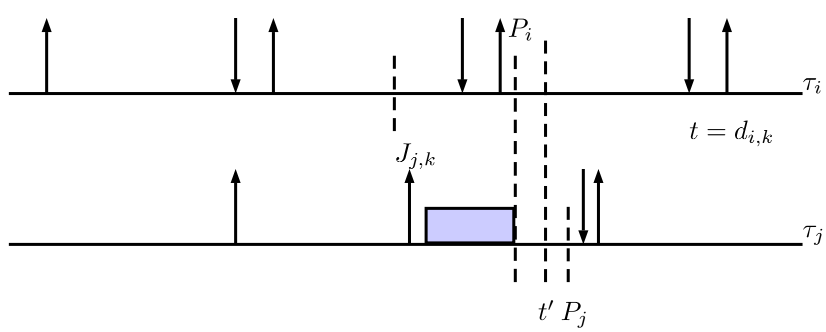

Fig. 1. Example One

Unfortunately, it is hard to calculate the exact time that $J_{j,k}$ has executed, and therefore here instead, we would use an upper bound of the resource $J_{i,k}$ has consumed to derive the test.

Lemma 1 (Worst Case Pattern): For each τ_j, it can consume maximum resource during $[0, t']$, and hence τ_i can receive minimal resource by t' if it is first released at 0 and all jobs are released as soon as possible with period T_i.

Proof 1: Suppose there is no deadline miss happens earlier, and as we shift the release pattern of τ_j left, $J_{j,1}$ will no longer request for any resource while the increment of the resource consumed by other jobs of τ_j is bounded by C_j. On the other hand, as we shift the pattern right, the promotion point of $J_{j,k}$ increases and hence $J_{j,k}$ is less likely to has higher priority than τ_i (and hence consume the resource).

For dual priority scheduling, then we can have the following theorem.

Theorem 1: In dual priority scheduling, τ_i is schedulable on a single processor if and only if for each absolute deadline of a job $d_{i,k}$ where $k \in N$, there exists an interval $r_{i,k} \leq t' \leq d_{i,k}$ for which the following condition holds:

$$dbf(\tau_i, t) + \sum_{\tau_j \in \{\tau - \tau_j\}} rbf_i(\tau_j, t') \leq t' \tag{3}$$

where $rbf_i(\tau_j, t')$ upper bounds the resource that τ_j has consumed by t' with corresponding priority and promotion point.

Proof 2: We can prove the statement that if τ_i is not schedulable then Equation 3 would not hold. Suppose that τ_i is not schedulable then there are legal event sequences in which τ_i misses some deadline when τ_i is assigned with the current priority and promotion point. Let S' denote such a sequence where τ_i misses deadline at the earliest time at t (no deadline misses happens earlier).

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 $[0, t']$ $(\forall t')$, other tasks have consumed an amount of resource more than

$$\sum_{\tau_j \in \{\tau - \tau_j\}} rbf_i(\tau_j, t') > t' - dbf(\tau_i, t) \Rightarrow dbf(\tau_i, t) > \max_{t'}(t' - \sum_{\tau_j \in \{\tau - \tau_j\}} rbf_i(\tau_j, t'))$$

which contradicts the Equation 3.

TABLE I

NOTATIONS

$n_j = \lfloor \frac{t'}{T_j} \rfloor$	$r_j = n_j \times T_j$	$P_j = n_j \times T_j + p_j$
$P_i = t - (D_i - p_i)$	$r_i = P_i - p_i$	$[a]_0 = \max(a, 0)$
$n_j^s = \lfloor \frac{r_i}{T_j} \rfloor$	$r_j^s = n_j^s \times T_j$	
$r_j^w = n_j^w \times T_j$	$n_j^w = \lfloor \frac{P_i}{T_j} \rfloor$	

II. SCHEDULABILITY TEST

Theorem 2: A task system τ is schedulable by dual priority scheduling if the following hold:

$$\forall \tau_i \in \tau, t \text{ where } k \in \{1, 2, \ldots\} : \exists t' : dbf(\tau_i, t) + \sum_{\tau_j \in \{\tau - \tau_j\}} rbf_i(\tau_j, t') \leq t' \tag{4}$$

Here we also derive $rbf_i^1(\tau_j, t')$ which denotes maximum possible execution that τ_j has received after r_i, and $rbf_i^2(\tau_j, t')$ which denotes maximum possible execution that τ_j has received after P_i (if $t' > P_i$). Assuming no deadline miss happens earlier, the total execution before r_i and P_i should not exceed r_i and P_i, respectively. As a result, we are able to tighten the schedulability test by bounding the total execution before r_i and P_i by r_i and P_i, respectively.

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

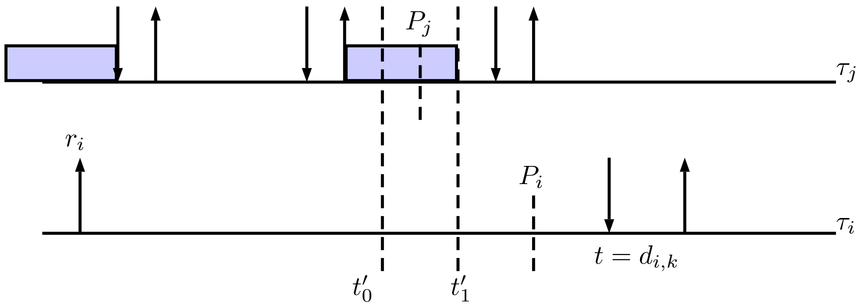

Fig. 2. $P_j \leq P_i$

Case 1 ($P_j \leq P_i$) as shown in Figure 2:

The promotion point of P_i is greater than P_j, and hence τ_i always has lower priority than τ_i. Thus its maximum possible execution units by t' is

$$rbf_i(\tau_j, t') = n_j.C_j + \min(C_j, t' - r_j)$$

Its maximum possible execution after r_i is

$$rbf_i^1(\tau_j, t') = \begin{cases} \min\{C_j, t' - r_i\} & \text{if } r_j \leq r_i \\ \min\{C_j, t' - r_j\} + \frac{r_j - r_j^s - T_j}{T_j} C_j + \min(C_j, [r_j^s + D_j - r_i]_0) & \text{if } r_j > r_i \end{cases}$$

Its maximum possible execution after P_i is

$$rbf_i^2(\tau_j, t') = \begin{cases} \min\{C_j, [t' - P_i]_0\} & \text{if } r_j \leq P_i \\ \min\{C_j, t' - r_j\} + \frac{r_j - r_j^w - T_j}{T_j} C_j + \min(C_j, [r_j^w + D_j - P_i]_0) & \text{if } r_j > P_i \end{cases}$$

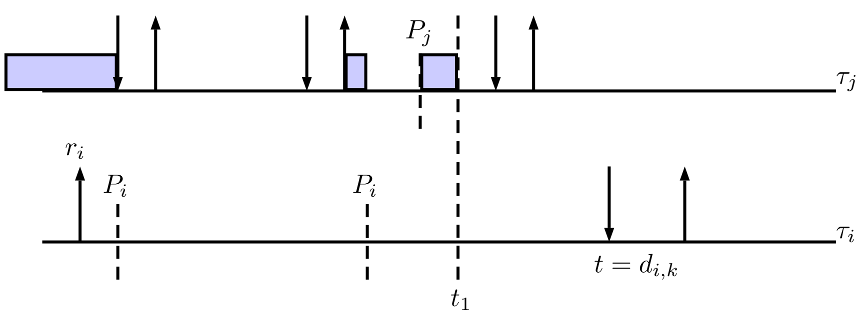

Fig. 3. $P_j > P_i$

Case 2 ($P_j > P_i$) as shown in Figure 3 τ_i has higher priority than τ_j's last job during $[\max(r_j, P_i), P_j]$ if $P_i \leq P_j$:

$$rbf_i(\tau_j, t') = \lfloor \frac{t'}{T_j} \rfloor C_j + \min\left(C_j, t' - r_j - (\min\{t', P_j\} - \max\{r_j, P_i\})\right)$$

Its maximum possible execution after r_i is

$$rbf_i^1(\tau_j, t') = \begin{cases} \min\{C_j, \min\{t', P_i\} - r_i\} & \text{if } t' < P_j \wedge r_j \le r_i \\ \min\{C_j, \min\{t', \max(r_j, P_i)\} - r_j\} + \frac{r_j - r_j^s - T_j}{T_j} C_j & \\ + \min(C_j, [r_j^s + D_j - r_i]_0) & \text{if } t' < P_j \wedge r_j > r_i \\ \min\{C_j, t' - r_i - (P_j - P_i)\} & \text{if } t' \ge P_j \wedge r_j \le r_i \\ \min\{C_j, t' - r_j - (P_j - \max(r_j, P_i))\} + \frac{r_j - r_j^s - T_j}{T_j} C_j & \\ + \min(C_j, [r_j^s + D_j - r_i]_0) & \text{if } t' \ge P_j \wedge r_j > r_i \end{cases}$$

Its maximum possible execution after P_i is

$$rbf_i^2(\tau_j, t') = \begin{cases} 0 & \text{if } t' < P_j \wedge r_j \le P_i \\ \frac{r_j - r_j^w - T_j}{T_j} C_j + \min(C_j, [r_j^w + D_j - P_i]_0) & \text{if } t' < P_j \wedge r_j > P_i \\ \min\{t' - P_j, C_j\} & \text{if } t' \ge P_j \wedge r_j \le P_i \\ \min\{t' - P_j, C_j\} + \frac{r_j - r_j^w - T_j}{T_j} C_j + \min(C_j, [r_j^w + D_j - P_i]_0) & \text{if } t' \ge P_j \wedge r_j > P_i \end{cases}$$

B. $rbf_i(\tau_j, t')$ when $j > i$

Case 1.1 ($P_i \le P_j \wedge r_j \le r_i$) as shown in Figure 4: τ_j may execute during $[r_j, r_i]$, and τ_j would not execute after r_j or P_i unless τ_i has finished.

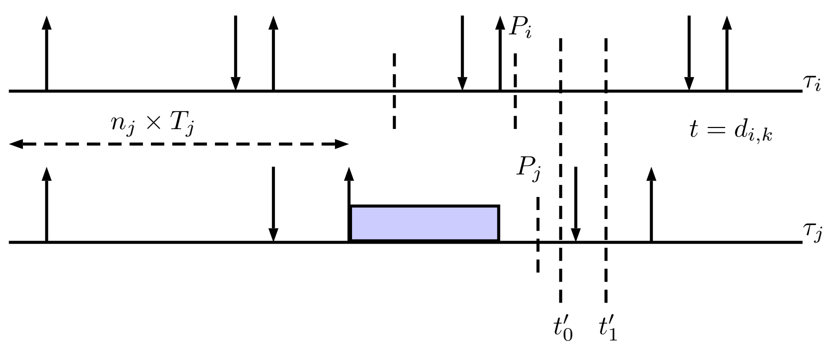

Fig. 4. $P_i \le P_j \wedge r_j \le r_i$

$$rbf_i(\tau_j, t') = (\lfloor \frac{t'}{T_j} \rfloor) \times C_j + \min(C_j, r_i - r_j)$$

$$rbf_i^1(\tau_j, t') = rbf_i^2(\tau_j, t') = 0$$

Case 1.2 ($P_i \le P_j \wedge r_j > r_i$) as shown in Figure 5: the last job of τ_j would not execute unless τ_i finishes. However all previous jobs are assumed to finish because otherwise the deadline miss should be already found. The $n_j - 1$ job must already finish by $\min(P_i, r_j - T_j + D_j)$.

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

$$rbf_i^1(\tau_j, t') = \min(C_j, [\min(P_i, r_j - T_j + D_j) - r_i]_0)$$
$$rbf_i^2(\tau_j, t') = 0$$

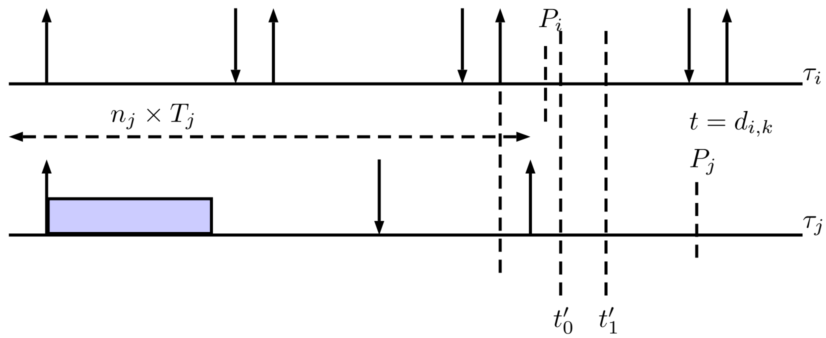

Fig. 5. $P_i \leq P_j \wedge r_j > r_i$

Case 2.1 ($P_i > P_j \wedge r_j \leq r_i$) as shown in Figure 6:the last job of τ_j can execute until $\min(t', P_i)$

$$rbf_i(\tau_j, t') = (n_j)C_j + \min\left(C_j, r_i - r_j + [\min(t', P_i) - \max(P_j, r_i)]_0\right)$$

$$rbf_i^1(\tau_j, t') = \min\{C_j, [\min(t', P_i) - \max(P_j, r_i)]_0\}$$

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

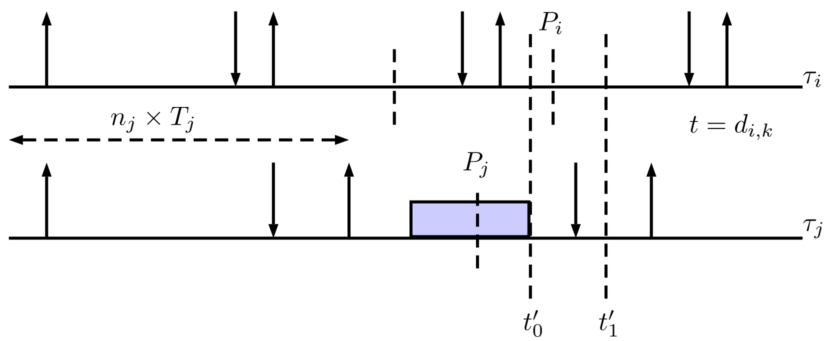

Fig. 6. $P_i > P_j \wedge r_j \leq r_i$

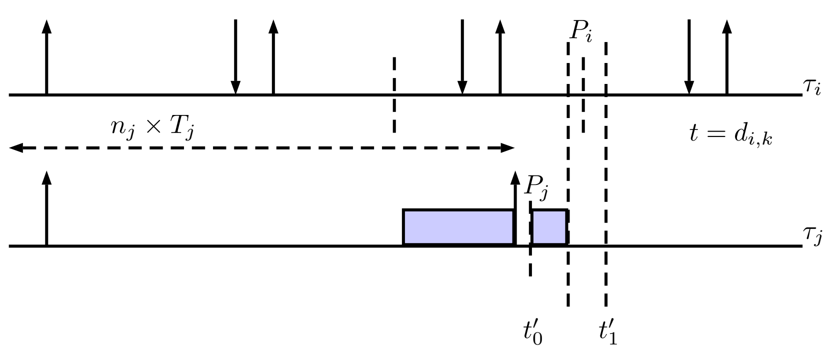

Fig. 7. $P_i > P_j \wedge r_j > r_i$

Case 2.2 ($P_i > P_j \wedge r_j > r_i$) as shown in Figure 7:

$$rbf_i(\tau_j, t') = (n_j)C_j + \min\left(C_j, [\min(t', P_i) - P_j]_0\right)$$

$$rbf_i^1(\tau_j, t') = \min\left(C_j, [\min(t', P_i) - P_j]_0\right) + \min(C_j, [r_j - T_j + D_j - r_i]_0)$$

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

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 τ_i itself, we simply assume its execution demand after r_i and P_i is C_j.

$$F_1(\tau_i, t, t') = \min\left(r_i, n_i \times C_i + \sum_{\tau_j \in \{\tau - \tau_i\}} rbf_i(\tau_j, t') - rbf_i^1(\tau_j, t')\right) + \sum_{\tau_j \in \{\tau - \tau_i\}} rbf_i^1(\tau_j, t') + C_i \quad (5)$$

$$F_2(\tau_i, t, t') = \min\left(P_i, n_i \times C_i + \sum_{\tau_j \in \{\tau - \tau_i\}} rbf_i(\tau_j, t') - rbf_i^2(\tau_j, t')\right) + \sum_{\tau_j \in \{\tau - \tau_i\}} rbf_i^2(\tau_j, t') + C_i \quad (6)$$

$$F(\tau_i, t, t') = \begin{cases} F_1(\tau_i, t, t') & \text{if } t' <= P_i \\ \min\{F_1(\tau_i, t, t'), F_2(\tau_i, t, t')\} & \text{otherwise} \end{cases} \quad (7)$$

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

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

IV. Draft Simulation Results

TABLE II
NOTATIONS

Uniform Distribution	[0.88,0.9]	[0.93,0.95]	[0.95,0.97]	[0.97,0.99]
Acceptance Ratio	1	1	0.995	0.982