Semantic Importance Dual-Priority Server: Properties

David R. Donari
Universidad Nacional del Sur - CONICET, Dpto. de Ing. Eléctrica y Computadoras,
Bahía Blanca, Argentina, 8000
ddonari@uns.edu.ar

and

Martin L. Duval
Universidad Nacional del Sur, Dpto. de Ing. Eléctrica y Computadoras,
Bahía Blanca, Argentina, 8000
mduval@uns.edu.ar

and

Leo D. Ordinez
Universidad Nacional del Sur - CONICET, Dpto. de Ing. Eléctrica y Computadoras,
Bahía Blanca, Argentina, 8000
lordinez@uns.edu.ar

and

Diana G. Sanchez
Universidad Nacional del Sur, Dpto. de Ing. Eléctrica y Computadoras,
Bahía Blanca, Argentina, 8000
sanchezd@criba.edu.ar

1 System Model

Most of the real-time tasks that interact with the environment through sensors are assumed to receive a value from those sensors and produce a result based on them. With the objective of taking account of this semantic aspect of a task, a parameter μ, named the *threshold*, is introduced. The μ threshold is established by the system developer at design time. So, it is based on the expected values of that result. If the result obtained exceeds the threshold, the task associated to that sensor is said to be IMPORTANT and it is associated, at least for one instance, to the set of $IMPORTANT$ tasks. On the contrary, when the result is below that threshold the task is NOT IMPORTANT and, analogously to the previous case, it will belong to the $NOT\,IMPORTANT$ set. It is worth mentioning, that the classification of a task is done once an execution is completed. In a formal way:

$$J_{i,j} \in \begin{cases} IMPORTANT & \text{if } \delta_{i,j-1} \geq \mu_i \\ NOT\,IMPORTANT & \text{if } \delta_{i,j-1} < \mu_i \end{cases}$$

where $\delta_{i,j}$ is a magnitude whose domain establishes an order relationship and it is based on the results produced by the task.

As this tasks are soft, their temporal characterization is not exact and is usually described by probability functions. In this sense, a task τ_i is composed by a series of jobs, being $J_{i,j}$ the j-Th job of task i, which arrives at a rate T_i, where this value is the *minimum interarrival time*. At the same time, each task is also characterized by a *worst case execution time* C_i. The time at which a job arrives to the system is known as

the activation time $a_{i,j}$ and the deadline of that job is obtained as follows: $d_{i,j} = a_{i,j} + T_i$. In addition to temporal features, the new introduced μ_i parameter is had.

A server is a software abstraction where, in a general case, several tasks are encapsulated. This is also known as the *isolation property*. Each server has a portion of the actual processor available bandwidth, so if a task tries to use more bandwidth than the one of its associated server, then the server is delayed and consequently the task is also delayed. As a consequence, by using a resource reservation mechanism based on servers, the schedulability analysis problem of the entire system is reduced to the one of estimating the schedulability of each server alone.

In a temporal characterization a server s has a budget Q_s, which is the maximum time available for execution of its tasks; an actual available budget c_s; a period P_s; a deadline d_s and a postponement factor α_s, which is used to impose a delay on the NOT IMPORTANT tasks.

2 The Algorithm

In this section, the algorithm SIDS will be formally presented, along with a series of properties that will be stated and proved. The main idea behind the algorithm is to postpone the execution of not important tasks, so that portion of the bandwidth can be used by other important tasks that belong to the same or to another server.

2.1 Definition and Functioning

In a simplified but general case, a SIDS is used to encapsulate a task whose available portion of the processor is bounded to the bandwidth of its SIDS. In the same line of reasoning, a system is composed by a certain number of SIDS, whose access to the processor is given by a higher level scheduling policy. If the chosen policy is Earliest Deadline First (EDF) [1], the SIDS with the closer deadline to the actual time is the one with the highest priority. At this point is where the newly introduced *postponement factor* plays a fundamental role. The fact of postponing the deadline of a SIDS with only a NOT IMPORTANT task makes it lose priority among the others.

On the other side, the imposition of a hard reservation makes that dynamic bandwidth distribution among the servers even more fair. In the case of SIDS, the hard reservation is introduced by means of differential waiting for replenishment of the SIDS' budget. With this in mind, a SIDS can be in one of four states at each moment of time:

Active: There is at least one job ready to be executed and $c_s > 0$.

Idle: There are no pending jobs to be executed.

Short_Wait: The execution budget was exhausted and there is at least one IMPORTANT job waiting to complete its execution.

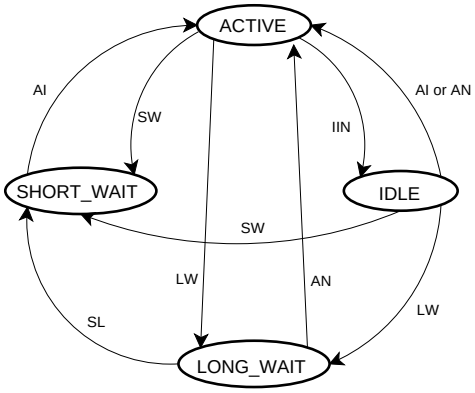
Long_Wait: Identical to the previous case, but there are no IMPORTANT pending jobs and there is at least one NOT IMPORTANT job waiting to execute.

In Figure 1(a) the different possible transitions between states is shown.

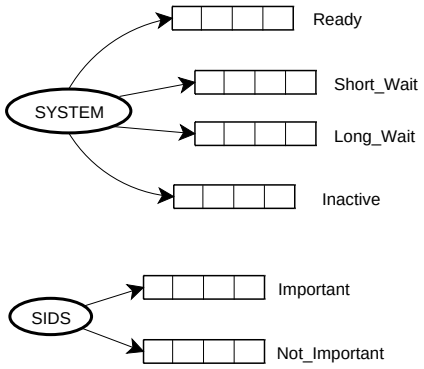
As was mentioned before, SIDS proposes a hierarchical scheduling architecture. In this sense, there are two levels of queues: first, the system queues; and second, the ones internal to a SIDS. Having this in mind and from the previous state model, in Figure 1(b) the different queues necessary in each part of the system are shown.

SIDS is based on a simple set of rules, which are described following this convention: **AI** is for Active Important; **AN** is for Active Not Important; **WS** is for Wait Short; **WL** is for Wait Long; **SL** is for Stop Long Wait; **IIN** is for Inactive Important/Not Important and **DB** is for Decrement Budget. In this sense, the rules are also numbered to distinguish the situation in which they are applied; for example, in the case of rule AI, there are three different moments in which it is applied keeping in all cases the same spirit. With this in mind, the rules previously described can be thought like a family of rules, where, despite the situation, each instance of the family performs the same task each time.

AI: SIDS has enough budget to execute jobs and there are IMPORTANT pending ones. A transition to ACTIVE state is performed.



(a) State model of SIDS.



(b) Different levels of queues in the SIDS approach.

Figure 1: Logical aspects of SIDS.

AN: SIDS has enough budget to execute jobs and there are NOT IMPORTANT pending ones. A transition to ACTIVE state is performed.

SW: When the SIDS' budget is exhausted and there are IMPORTANT pending jobs it waits for at most one period for its replenishment. A transition to WAIT_SHORT state is performed.

LW: When the SIDS' budget is exhausted and there are NOT IMPORTANT pending jobs it waits for a multiple α_s of its period for replenishment. A transition to WAIT_LONG state is performed.

SL: If a SIDS is in WAIT_LONG state and an IMPORTANT job arrives, it cuts down the waiting to, at most, one period from the activation time of that job. A transition to WAIT_SHORT state is performed.

DB: When a SIDS executes a job for one time unit, it decrements it budget accordingly.

IIN: When a job finishes and there are not pending ones, the SIDS goes to IDLE state.

With all, a more formal scheme than the rules previously shown is presented in Algorithm 1. Auxiliary functions used in the algorithm are grouped in Table 1.

update_SIDS_IMPORTANT(){ $c_s \leftarrow Q_s$ $d_k \leftarrow a_k + P_s$ $k \leftarrow k + 1$}	update_SIDS_NOT_IMPORTANT(){ $c_s \leftarrow Q_s$ $d_k \leftarrow a_k + \alpha P_s$ $k \leftarrow k + 1$}
postpone_IMPORTANT(){ $r_s \leftarrow d_k + P_s$}	postpone_NOT_IMPORTANT(){ $r_s \leftarrow d_k + \alpha P_s$}

Table 1: Auxiliar functions used in Algorithm 1

2.2 Properties

In general, the execution time demanded by a task τ_i in the interval $[t_1, t_2]$ is given by:

$$D_i(t_1, t_2) = \sum_{(t_1 \leq a_{i,j}) \wedge (t_2 \geq d_{i,j})} C_{i,j} \tag{2.1}$$

where $a_{i,j}$, $d_{i,j}$ and $C_{i,j}$ are the activation time, the deadline and the worst case execution time of the j-Th job of τ_i, respectively.

From equation 2.1 and the definition of SIDS, three possible relations are deduced between the intervals $[t_1, t_2]$ and $[a_k, d_k]$:

Algorithm 1 Algorithm SIDS

```
When a job J_j arrives in t = a_k and the SIDS is IDLE do
    Enqueue it
    if J_j ∈ IMPORTANTS then
        if t ≥ d_k − c_s P_s/Q_s then {Become ACTIVE}
            update_SIDS_IMPORTANT()      ⟶ Rule AI.1
        else if (d_k ≥ t) and (c_s = 0) then {Go to WAIT_SHORT}
            postpone_IMPORTANT()      ⟶ Rule SW.1
        else {Become ACTIVE}
            The job is served with the current budget and deadline      ⟶ Rule AI.2
        end if
    else
        if t ≥ d_k − c_s α P_s/Q_s then {Become ACTIVE}
            update_SIDS_NOT_IMPORTANT()      ⟶ Rule AN.1
        else if (d_k ≥ t) and (c_s = 0) then {Go to LONG_WAIT}
            postpone_NOT_IMPORTANT()      ⟶ Rule LW.1
        else {Become ACTIVE}
            The job is served with the current budget and deadline      ⟶ Rule AN.2
        end if
    end if
end When
When a job J_j arrives in t and the SIDS is either ACTIVE or in WAIT_SHORT do
    Enqueue it
end When
When a job J_j arrives in t = a_k and the SIDS is in WAIT_LONG do
    Enqueue it
    if J_j ∈ IMPORTANTS then {Go to SHORT_WAIT}
        r_s ← min{a_k + P_s, r_s}      ⟶ Rule SL
    end if
end When
When a job J_j served by SIDS S_s executes for 1 unit of time do
    c_s ← c_s − 1      ⟶ Rule DB
end When
When SIDS S_s is executing J_j and c_s = 0 do
    if J_j ∈ IMPORTANTES then {Go to SHORT_WAIT}
        postpone_IMPORTANT()      ⟶ Rule SW.2
    else {Go to LONG_WAIT}
        postpone_NOT_IMPORTANT()      ⟶ Rule LW.2
    end if
end When
When (there are IMPORTANT pending jobs) and (t ≥ r_s) do {Become ACTIVE}
    a_k ← t      ⟶ Rule AI.3
    update_SIDS_IMPORTANT()
end When
When (there are NOT IMPORTANT pending jobs) and (t ≥ r_s) do {Become ACTIVE}
    a_k ← t      ⟶ Rule AN.3
    update_SIDS_NOT_IMPORTANT()
end When
When a job J_j finishes do
    if (There is at least one pending job) and (there is enough budget) then {Remain ACTIVE}
        Depending on the kind of pending jobs      ⟶ Rule AI.2 or Rule AN.2
    else {Go IDLE}
            ⟶ Rule IIN
    end if
end When
```

(1) $t_2 - t_1 < d_k - a_k$

(2) $t_2 - t_1 = d_k - a_k$

(3) $t_2 - t_1 > d_k - a_k$

According to the definition of SIDS and due to the *hard reservation* condition, it can be stated that there can be just one interval $[a_k, d_k]$ for each period P_s of the server. Then, from the relations between the interval $[a_k, d_k]$ and the period P_s and between the intervals $[t_1, t_2]$ and $[a_k, d_k]$, case (1) can not be given and from (2) and (3) come out the following definition for a SIDS s.

Definition 2.1 (Maximum demand bound function)**.** The maximum demand bound function of a SIDS with only IMPORTANT tasks is given by:

$$D_{s_{MAX}}(t_1, t_2) = \left(\left\lfloor \frac{t_2}{P_s} \right\rfloor - \left\lfloor \frac{t_1}{P_s} \right\rfloor\right) Q_s \tag{2.2}$$

Based on the distinction done by the algorithm between IMPORTANT and NOT IMPORTANT tasks, analogously to the previous definition, the following is obtained.

Definition 2.2 (Minimum demand bound function)**.** The minimum demand bound function of a SIDS with only NOT IMPORTANT tasks is given by:

$$D_{s_{MIN}}(t_1, t_2) = \left\lfloor \frac{t_2 - t_1}{\alpha_s P_s} \right\rfloor Q_s \tag{2.3}$$

Theorem 2.1 (Isolation Theorem)**.** *A SIDS with parameters* (Q_s, P_s, α_s) *uses a bandwidth* U_s *of, at least,* $\frac{Q_s}{\alpha P_s}$ *and, at most,* $\frac{Q_s}{P_s}$

Proof. The execution of a SIDS can be thought as composed by a number of execution chunks, namely k in an increasing order, being e_k the execution time demanded by chunk k. With this in mind, Equation 2.1 would be:

$$\forall t_1, t_2 \, \exists k_1, k_2 : D_s(t_1, t_2) = \sum_{k: a_k \geq t_1 \wedge d_k \leq t_2} e_k = \sum_{k=k_1}^{k_2} e_k$$

Being f_k the time at which a SIDS finishes the execution of chunk k and c_s its actual budget, it follows:

$$c_s(f_k) = c_s(a_k) - e_k$$

and

$$c_s(a_{k+1}) = \begin{cases} c_s(f_k) & \text{if } d_{k+1} \text{ was generated by rules AI.2 or AN.2} \\ Q_s & \text{if } d_{k+1} \text{ was generated by rules AI.1 or AI.3 or AN.1 or AN.3} \end{cases}$$

So, the demonstration consists on showing that

$$(d_{k_2} - a_{k_1}) \frac{Q_s}{\alpha P_s} \leq D_s(a_{k_1}, d_{k_2}) + c_s(f_{k_2}) \leq (d_{k_2} - a_{k_1}) \frac{Q_s}{P_s}$$

by application of the algorithm and induction over $k_2 - k_1$, this is, the amount of chunks generated.

Base case: In the interval $[t_1, t_2]$ there is only one active chunk, so $k_1 = k_2 = k$. The following cases can be given:

1. $d_k < a_k + P_s$
2. $d_k = a_k + P_s$
3. $d_k = a_k + \alpha P_s$

1) $\mathbf{d_k < a_k + P_s}$

If $d_k < a_k + P_s$ then d_k was generated by rules AI.2 or AN.2. Both cases are the same because this is the situation in which the SIDS is idle and a job arrives, but it is served with the budget remaining of the las execution of the SIDS.

$$a_k + \frac{c_s(a_k)}{Q_s} P_s < d_k$$

as $c_s(f_k) = c_s(a_k) - e_k = c_s(a_k) - D_s(a_k, d_k)$ then

$$a_k + \frac{D_s(a_k, d_k) + c_s(f_k)}{Q_s} P_s < d_k$$

reordering

$$D_s(a_k, d_k) + c_s(f_k) < (d_k - a_k) \frac{Q_s}{P_s} \tag{2.4}$$

2) $\mathbf{d_k = a_k + P_s}$

If $d_k = a_k + P_s$ then d_k was generated by rules AI.1 or AI.3 then

$$c_s(f_k) + D_s(a_k, d_k) = c_s(a_k) = Q_s$$

and we have

$$c_s(f_k) = c_s(a_k) - e_k = c_s(a_k) - D_s(a_k, d_k)$$

reordering the equation of rule AI.1

$$d_k \leq a_k + c_s(a_k)\frac{P_s}{Q_s}$$

$$d_k \leq a_k + c_s(f_k) + D_s(a_k, d_k)\frac{P_s}{Q_s}$$

reordering again

$$(d_k - a_k)\frac{Q_s}{P_s} \leq c_s(f_k) + D_s(a_k, d_k) = Q_s$$

in general

$$(d_k - a_k)\frac{Q_s}{P_s} \leq c_s(f_k) + D_s(a_k, d_k) \tag{2.5}$$

3) $\mathbf{d_k = a_k + \alpha P_s}$

If $d_k = a_k + \alpha P_s$ then d_k was generated by rules AN.1 or AN.3 then

$$c_s(f_k) + D_s(a_k, d_k) = c_s(a_k) = Q_s$$

and we have

$$c_s(f_k) = c_s(a_k) - e_k = c_s(a_k) - D_s(a_k, d_k)$$

reordering the equation of rule AN.1

$$d_k \leq a_k + c_s(a_k)\frac{\alpha P_s}{Q_s}$$

$$d_k \leq a_k + c_s(f_k) + D_s(a_k, d_k)\frac{\alpha P_s}{Q_s}$$

reordering again

$$(d_k - a_k)\frac{Q_s}{\alpha P_s} \leq c_s(f_k) + D_s(a_k, d_k) = Q_s$$

in general

$$(d_k - a_k)\frac{Q_s}{\alpha P_s} \leq c_s(f_k) + D_s(a_k, d_k) \tag{2.6}$$

From equations 2.4, 2.5 and 2.6 we have that for an only one active chunk in the system:

$$(d_k - a_k)\frac{Q_s}{\alpha P_s} \leq c_s(f_k) + D_s(a_k, d_k) \leq (d_k - a_k)\frac{Q_s}{P_s} \tag{2.7}$$

Inductive step:

$$(d_{k_2-1} - a_{k_1})\frac{Q_s}{\alpha P_s} \leq D_s(a_{k_1}, d_{k_2-1}) + c_s(f_{k_2-1}) \leq (d_{k_2-1} - a_{k_1})\frac{Q_s}{P_s} \tag{2.8}$$

Given the possible relations between d_k and d_{k-1} the following cases are to be considered:

1. $d_k \geq d_{k-1} + P_s$, with $a_k \geq d_{k-1}$. The possible rules applied are: 1) AI.1 2) LW.1 or LW.2, SL and AI.3.
2. $d_k = d_{k-1}$, with $a_k < d_{k-1}$. The possible rules applied are: 1) AI.2 2) AN.2 3) AIN.
3. $d_k = d_{k-1} + P_s$, with $a_k \leq d_{k-1}$. The possible rules applied are: SW.1 or SW.2 and AI.3.
4. $d_k \geq d_{k-1} + \alpha P_s$, with $a_k \geq d_{k-1}$. The possible rules applied are: AN.1.
5. $d_k \geq d_{k-1} + \alpha P_s$, with $a_k \leq d_{k-1}$. The possible rules applied are: LW.1 or LW.2 and AN.3.

1) $\mathbf{d_{k-1} + P_s \leq d_k < d_{k-1} + \alpha P_s}$
The only part of the inductive hypothesis that is affected by the rules applied is:

$$D_s(a_{k_1}, d_{k_2-1}) + c_s(f_{k_2-1}) \leq (d_{k_2-1} - a_{k_1})\frac{Q_s}{P_s}$$

then the previous expression can be expressed as

$$\sum_{k=k_1}^{k_2-1} e_k + c_s(f_{k_2-1}) \leq (d_{k_2-1} - a_{k_1})\frac{Q_s}{P_s}$$

adding e_{k_2} to both sides

$$\sum_{k=k_1}^{k_2-1} e_k + e_{k_2} \leq (d_{k_2-1} - a_{k_1})\frac{Q_s}{P_s} - c_s(f_{k_2-1}) + e_{k_2}$$

$$\sum_{k=k_1}^{k_2} e_k \leq (d_{k_2-1} - a_{k_1})\frac{Q_s}{P_s} - c_s(f_{k_2-1}) + e_{k_2}$$

taking that $c_s(f_k) = c_s(a_k) - e_k$

$$\sum_{k=k_1}^{k_2} e_k \leq (d_{k_2-1} - a_{k_1})\frac{Q_s}{P_s} - c_s(f_{k_2-1}) + c_s(a_{k_2}) - c_s(f_{k_2})$$

applying Rule AI.1 or AI.3 we have that $c_s(a_{k_2}) = Q_s$

$$\sum_{k=k_1}^{k_2} e_k + c_s(f_{k_2}) \leq (d_{k_2-1} - a_{k_1})\frac{Q_s}{P_s} + Q_s - c_s(f_{k_2-1})$$

and

$$\sum_{k=k_1}^{k_2} e_k + c_s(f_{k_2}) \leq (d_{k_2-1} - a_{k_1})\frac{Q_s}{P_s} + Q_s - c_s(f_{k_2-1}) \leq (d_{k_2-1} - a_{k_1})\frac{Q_s}{P_s} + Q_s$$

discarding the term in the middle and reordering we get

$$\sum_{k=k_1}^{k_2} e_k + c_s(f_{k_2}) \leq (d_{k_2-1} + P_s - a_{k_1})\frac{Q_s}{P_s}$$

knowing that $d_{k_2} \geq d_{k_2-1} + P_s$ we finally get

$$D_s(a_{k_1}, d_{k_2}) + c_s(f_{k_2}) \leq (d_{k_2} - a_{k_1})\frac{Q_s}{P_s} \tag{2.9}$$

2) $\mathbf{d_k = d_{k-1}}$
We start with the upper half of the inductive hypothesis.

$$D_s(a_{k_1}, d_{k_2-1}) + c_s(f_{k_2-1}) \leq (d_{k_2-1} - a_{k_1})\frac{Q_s}{P_s}$$

adding e_{k_2} to both sides

$$\sum_{k=k_1}^{k_2-1} e_k + e_{k_2} \leq (d_{k_2-1} - a_{k_1})\frac{Q_s}{P_s} - c_s(f_{k_2-1}) + e_{k_2}$$

being $d_{k_2-1} = d_{k_2}$ we have

$$\sum_{k=k_1}^{k_2} e_k \leq (d_{k_2} - a_{k_1})\frac{Q_s}{P_s} - c_s(f_{k_2-1}) + e_{k_2}$$

considering $c_s(f_k) = c_s(a_k) - e_k$ and $c_s(f_{k_2-1}) = c_s(a_{k_2})$ because of rules AI.2 or AIN

$$\sum_{k=k_1}^{k_2} e_k \leq (d_{k_2} - a_{k_1})\frac{Q_s}{P_s} - c_s(a_{k_2}) + c_s(a_{k_2}) - c_s(f_{k_2})$$

and finally

$$D_s(a_{k_1}, d_{k_2}) + c_s(f_{k_2}) \leq (d_{k_2} - a_{k_1})\frac{Q_s}{P_s} \tag{2.10}$$

To the lower half of the proof we proceed in the same fashion

$$(d_{k_2-1} - a_{k_1})\frac{Q_s}{\alpha P_s} \leq D_s(a_{k_1}, d_{k_2-1}) + c_s(f_{k_2-1})$$

adding e_{k_2} to both sides

$$(d_{k_2-1} - a_{k_1})\frac{Q_s}{\alpha P_s} - c_s(f_{k_2-1}) + e_{k_2} \leq \sum_{k=k_1}^{k_2-1} e_k + e_{k_2}$$

being $d_{k_2-1} = d_{k_2}$ we have

$$(d_{k_2} - a_{k_1})\frac{Q_s}{\alpha P_s} - c_s(f_{k_2-1}) + e_{k_2} \leq \sum_{k=k_1}^{k_2} e_k$$

considering $c_s(f_k) = c_s(a_k) - e_k$ and $c_s(f_{k_2-1}) = c_s(a_{k_2})$ because of rules AN.2 or AIN

$$(d_{k_2} - a_{k_1})\frac{Q_s}{\alpha P_s} - c_s(a_{k_2}) + c_s(a_{k_2}) - c_s(f_{k_2}) \leq \sum_{k=k_1}^{k_2} e_k$$

and finally

$$(d_{k_2} - a_{k_1})\frac{Q_s}{\alpha P_s} \leq D_s(a_{k_1}, d_{k_2}) + c_s(f_{k_2}) \tag{2.11}$$

3) $\mathbf{d_k = d_{k-1} + P_s}$

This case is similar to 1), in fact it can be seen as a special case of that one. The only part of the inductive hypothesis that is affected by the rules applied is:

$$D_s(a_{k_1}, d_{k_2-1}) + c_s(f_{k_2-1}) \leq (d_{k_2-1} - a_{k_1})\frac{Q_s}{P_s}$$

adding e_{k_2} to both sides

$$\sum_{k=k_1}^{k_2-1} e_k + e_{k_2} \leq (d_{k_2-1} - a_{k_1})\frac{Q_s}{P_s} - c_s(f_{k_2-1}) + e_{k_2}$$

$$\sum_{k=k_1}^{k_2} e_k \leq (d_{k_2-1} - a_{k_1})\frac{Q_s}{P_s} - c_s(f_{k_2-1}) + e_{k_2}$$

taking that $c_s(f_k) = c_s(a_k) - e_k$

$$\sum_{k=k_1}^{k_2} e_k \leq (d_{k_2-1} - a_{k_1})\frac{Q_s}{P_s} - c_s(f_{k_2-1}) + c_s(a_{k_2}) - c_s(f_{k_2})$$

applying Rule AI.3 we have that $c_s(a_{k_2}) = Q_s$

$$\sum_{k=k_1}^{k_2} e_k + c_s(f_{k_2}) \leq (d_{k_2-1} - a_{k_1})\frac{Q_s}{P_s} + Q_s - c_s(f_{k_2-1})$$

and

$$\sum_{k=k_1}^{k_2} e_k + c_s(f_{k_2}) \leq (d_{k_2-1} - a_{k_1})\frac{Q_s}{P_s} + Q_s - c_s(f_{k_2-1}) \leq (d_{k_2-1} - a_{k_1})\frac{Q_s}{P_s} + Q_s$$

discarding the term in the middle and reordering we get

$$\sum_{k=k_1}^{k_2} e_k + c_s(f_{k_2}) \leq (d_{k_2-1} + P_s - a_{k_1})\frac{Q_s}{P_s}$$

knowing that $d_{k_2} = d_{k_2-1} + P_s$ we finally get

$$D_s(a_{k_1}, d_{k_2}) + c_s(f_{k_2}) \leq (d_{k_2} - a_{k_1})\frac{Q_s}{P_s} \tag{2.12}$$

4) $\mathbf{d_k \geq d_{k-1} + \alpha P_s}$
The only part of the inductive hypothesis affected by the rules applied is the lower one.

$$(d_{k_2-1} - a_{k_1})\frac{Q_s}{\alpha P_s} \leq D_s(a_{k_1}, d_{k_2-1}) + c_s(f_{k_2-1})$$

adding e_{k_2} to both sides

$$(d_{k_2-1} - a_{k_1})\frac{Q_s}{\alpha P_s} + e_{k_2} \leq \sum_{k=k_1}^{k_2-1} e_k + e_{k_2} + c_s(f_{k_2-1})$$

being $e_k = c_s(a_k) - c_s(f_k)$

$$(d_{k_2-1} - a_{k_1})\frac{Q_s}{\alpha P_s} + c_s(a_{k_2}) - c_s(f_{k_2}) - c_s(f_{k_2-1}) \leq \sum_{k=k_1}^{k_2} e_k$$

by rule AN.1 $c_s(a_{k_2}) = Q_s$

$$(d_{k_2-1} - a_{k_1})\frac{Q_s}{\alpha P_s} + Q_s - c_s(f_{k_2-1}) \leq \sum_{k=k_1}^{k_2} e_k + c_s(f_{k_2})$$

$$\frac{Q_s}{\alpha P_s}(d_{k_2-1} + \alpha P_s - a_{k_1}) - c_s(f_{k_2-1}) \leq \sum_{k=k_1}^{k_2} e_k + c_s(f_{k_2})$$

like $d_k \geq d_{k-1} + \alpha P_s$

$$\frac{Q_s}{\alpha P_s}(d_{k_2} - a_{k_1}) - c_s(f_{k_2-1}) \leq \sum_{k=k_1}^{k_2} e_k + c_s(f_{k_2})$$

$$\frac{Q_s}{\alpha P_s}(d_{k_2} - a_{k_1}) \leq \sum_{k=k_1}^{k_2} e_k + c_s(f_{k_2}) + c_s(f_{k_2-1})$$

being $c_s(a_{k_2} = Q_s \geq c_s(f_{k_2-1}))$ we get

$$\frac{Q_s}{\alpha P_s}(d_{k_2} - a_{k_1}) \leq \sum_{k=k_1}^{k_2} e_k + c_s(f_{k_2}) + c_s(f_{k_2-1}) \leq \sum_{k=k_1}^{k_2} e_k + c_s(f_{k_2}) + c_s(a_{k_2})$$

then

$$\frac{Q_s}{\alpha P_s}(d_{k_2} - a_{k_1}) \leq \sum_{k=k_1}^{k_2} e_k + c_s(f_{k_2}) + c_s(a_{k_2})$$

reordering we get

$$\frac{Q_s}{\alpha P_s}(d_{k_2} - a_{k_1}) - Q_s \leq \sum_{k=k_1}^{k_2} e_k + c_s(f_{k_2})$$

This last inequation means that the processor's demand plus the budget at the end of the execution of the last chunk is great or equal than the budget required between the activation of the first chunk and the finishing of the last one minus a full replenishment (Q_s). This situation, however uncommon in practice can harm the execution of the last chunk.

$$\left\lfloor \frac{(d_{k_2} - a_{k_1})}{\alpha P_s} \right\rfloor Q_s \leq D_s(a_{k_1}, d_{k_2}) + c_s(f_{k_2}) \tag{2.13}$$

5) $\mathbf{d_k \geq d_{k-1} + \alpha P_s}$

The proof is analogous to case 4), the rules applied change but they are complementary in execution. For instance, a possible execution sequence of the rules could be: AN.1 , LW.2 and AN.3.

Then, by inequations 2.9, 2.10, 2.11, 2.12 and 2.13 can be stated that

$$\boxed{(d_{k_2} - a_{k_1})\frac{Q_s}{\alpha P_s} \leq D_s(a_{k_1}, d_{k_2}) + c_s(f_{k_2}) \leq (d_{k_2} - a_{k_1})\frac{Q_s}{P_s}} \tag{2.14}$$

□

Theorem 2.2 (Schedulability Property)**.** *Given a set of tasks with total utilization factor U_T and a set of SIDS servers with total utilization factor U_{SIDS} (considering only the upper bound limit), then the whole set is schedulable by Earliest Deadline First (EDF) if and only if*

$$U_T + U_{SIDS} \leq 1$$

Proof. The proof follows directly from the isolation theorem. □

Theorem 2.3 (Hard Schedulability Property)**.** *Given a hard important real-time task τ_i with parameters C_i, d_i and T_i, then it is schedulable by a SIDS with parameters Q_s and P_s, such that $C_i \leq Q_s$ and $T_i = P_s$, if and only if it is schedulable by EDF.*

Proof. Since task τ_i is hard, the difference between its job's activations is given by its period (or minimum interarrival time), which is equal to the period of the SIDS. In particular, $a_{k+1} - a_k \geq P_s$ considering jitter or the case that the task is aperiodic. As a consequence of this and because $\tau_i \in IMPORTANTS$, the deadline generated by the SIDS algorithm is $d_k = a_k + P_s$; which is, in fact, the same deadline of the task (according to [1]). Besides, the restriction of $C_i \leq Q_s$ gives the server enough budget to complete the execution of every job without postponing its deadline. Moreover, the SIDS will never go to a wait state because each time a job arrives is served by rule AI.1. This can be easily proved arguing that $P_s \geq Q_s$ and considering $d_k = a_k + P_s$. □

Property 2.1 (Compatibility Property)**.** *In the absense of NOT IMPORTANT tasks the algorithm behaves like IRIS-HR.*

Proof. If there are only IMPORTANT tasks, the rules that can actually be applied are: AI.1 SW.1, AI.2, DB, SW.2, AI.3 (related to important jobs) and IIN, which correspond directly to 1.i, 1.ii, 1.iii, 2, 3, 4 and 5 from the IRIS-HR presented in [2]. □

Property 2.2 (Maximum Deadline Value)**.** *The highest value that can be assigned to a SIDS deadline is given by:*

$$d_{MAX} = d_{s-1} + 2\alpha P_s$$

Proof. This property follows directly from the application of rules related to NOT IMPORTANT tasks and without any interruption due to IMPORTANT ones. Particularly, there are two possible combinations of rules:

1. Rules: AN.1, LW.2 and AN.3.

2. Rules: LW.1 and AN.3.

In both cases, there is a long wait involved, which takes up to αP_s units of time from the deadline; and then a deadline postponement of the same amount. Hence, the new deadline is $2\alpha P_s$ units of time from the previous one. □

References

[1] Liu, C. L., and Layland, J. W. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM 20*, 1 (1973), 46–61.

[2] Marzario, L., Lipari, G., Balbastre, P., and Crespo, A. Iris: A new reclaiming algorithm for server-based real-time systems. In *Proceedings of the 10th IEEE RTAS* (Toronto, Canada, 2004), IEEE Computer Society.