Efficient Schedulability Analysis of Serial Transactions

Karim TRAORE, Emmanuel GROLLEAU, Francis COTTET

LISI/ENSMA
Laboratoire d'Informatique Scientifique et Industrielle
École Nationale de Mécanique et d'Aérotechnique
Téléport 2 – BP 40109 F-86961 Chasseneuil Futuroscope Cedex, France
karim.traore@ensma.fr, grolleau@ensma.fr, cottet@ensma.fr

Abstract. On the basis of a concrete real-time application, we present in this article a new task model called "serial transaction". This model is a particular instance of the task model with offsets defined by Tindell and Palencia and al.. A serial transaction is typically a task reading serial information (RS232, CAN,…): several instances are identical and read a unitary part of a serial packet, these tasks have the same WCET, offset shifting, priority and relative deadline. In addition, the last task of a transaction has to deal with the packet, and is typically longer, but has a longer relative deadline, and a lower priority. The need for this task model appeared in a real application, that couldn't be validated easily using known methods on transactions, so we present a less pessimistic and simpler (to implement) real-time evaluation method dedicated on to this new model.

1 Introduction

Several laboratories of Poitiers (ENSMA and University) are developing together a mini UAV (Unmanned Air Vehicle) (see Figure 1). The LISI is in charge of developing and validating the system (embedded and ground station). The embedded processing unit is a microcontroller (Freescale/Motorola MPC555) connected via serial port to a GPS receiver and a modem used in order to communicate with the ground station. The measurement of the attitude of the UAV is done by an IMU (Inertial Measurement Unit) connected to the microcontroller via a CAN network.

In the development of a real-time application like this one, two techniques of scheduling can be used : the on-line scheduling, with a fixed [LL73, LW82, Aud91] or variable allocation of priorities of the tasks in the tasks set [Der74, Lab74, DM89] and off-line techniques which use a sequence whose correctness was proved [XP92, Gro99]. The real-time RTOS (Real-Time Operational System) OSEKTurbo OS/MPC5xx [OSM1, OSM2], in conformity with standard OSEK/VDX [Osek1, Osek2], selected for this application, allows only fixed priorities. We thus used an on-line approach with fixed priority technique.

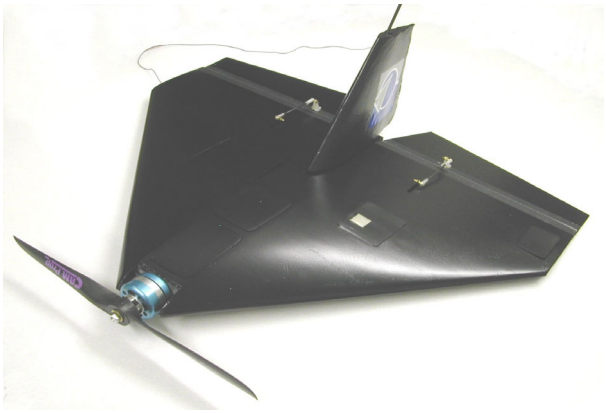


Fig.1. the AMADO

After the definition of software architecture and temporal parameters of the various tasks, one of the most important phases is the temporal validation which consists in proving that whatever happens, all the tasks will meet their temporal constraints. RTA (Response Time analysis) methods are used to bound the worst case response time of the tasks of an application. Tindell [Tin94] proposed a method for calculating an upper limit of the worst-case response time less pessimistic than classic RTA (considering a critical instant consisting of a simultaneous release of all the tasks).

Palencia and Harbour [PG98] extended Tindell's work with dynamic offsets, and formalized his work as transactions. Lastly, [TN04b][MS03] introduced the concept of "imposed" interference differing from "released for execution" interference used by Tindell. However, for now the exact calculation methods used to determinate the exact worst case response time relies on calculating every combination of the tasks of the transactions; it thus remains exponential in time.

In order to validate the control system of the UAV, we had to deal with tasks with offset witch are particular instances of transactions: these tasks are activated by peripherals connected on serial and CAN ports. Section 2 presents the case study. Section 3 reminds some general results about transactions. Section 4 present some new results obtained, allowing us to analyse the interference of a serial transaction in a pseudo polynomial time for a subset of the tasks of the task system. Section 5 applies these new results in order to validate our case study.

2 Presentation of serial transaction

The project, named AMADO, is a UAV with a wingspread of 55 cm, using a delta shaped wing with two symmetrical drifts for a total weight (including the control system) of 930 grams. The main objective is to create an autonomous plane embedding a camera, and to be able to follow dynamically defined waypoints. The

UAV is connected to a ground station thanks to a wireless modem, allowing it to receive high level orders during a mission. The critical parts of the flight control are embedded.

2.1 Description of the application

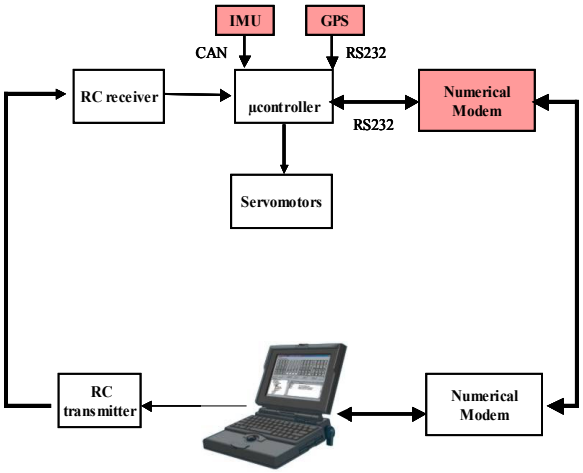


Fig.2. main architecture of the AMADO

The Figure 2 shows two parts: the ground station, and the embedded station. The ground station can communicate thanks to a half duplex modem with the embedded system, and the traditional radio emitter is kept as an emergency control in case of general failure of the embedded system. The main role of the ground station is video displaying recording, flight instruments, and high level commands (either waypoints flight, or assisted flight).

The embedded system heart is a Freescale/Motorola MPC555 connected to the actuators (3 servo-commands and the speed-variator, refreshed every 20 ms), an IMU [IMU1] (Inertial Measurement Unit), a GPS receiver [GPS1], a traditional radio receiver and a modem.. The MPC555 is a 32 bits PowerPC with a frequency of 40MHZ, 448KB of flash memory and 26KB of RAM; moreover, it has important units such as the MIOS (Modular Input Output System) for the capture and the generation of signals PPM (Pulse Position Modulation) necessary for activating the servomotors and to read the commands for the traditional radio, 2 SCI (Serial Communication Interfaces) for the serial asynchronous interfaces, 2 CAN interfaces network, 2TPU (Time Processors Units) that can be used to control 16 Inputs/outputs and 2 analog-to-digital converters per TPU. It contains also a floating point unit.

Two sensors are used in order to calculate the position and attitude of the UAV: the GPS receiver and the IMU. The Inertial Measurement Unit sends information about angular speed and accelerations, which, once treated, give the roll and the pitch of the UAV. This IMU is connected on a CAN port and delivers information at a frequency of 50Hz and a throughput of 1Mbps. A frame of the IMU is compound of 3 blocks of 6 bytes. In order for the system to get a complete frame, each block must be read and

stored before the next arrives. Once the system has 3 blocks, it can constitute the frame, and handle it to calculate the roll and the pitch.

The GPS receiver is used to get the speed (direction and module) and the absolute 3Dimensional position of the UAV. The GPS Receiver sends data to the controller at a frequency of 4Hz and delivers information with a throughput of 57600bps. As a RS232 communication, the information is sent byte after byte; the number of bytes sent during one period (frame) of the GPS can reach 120 bytes. As in the case of the IMU, the system must recover each byte and arrange it before the arrival of the next byte, under penalty of losing the complete frame.

Finally the modem [Modem1] connected to the microcontroller on the serial port is bi-directional and communicates with the microcontroller at a throughput of 115kbps. The length of the frame transmitted to the microcontroller by the modem can reach 10 bytes. The requirements are the same as in the case of the GPS receiver. In the presentation of this architecture, we omitted voluntarily the video circuit that does not have any impact on the real-time aspects of this application.

2.2 Software architecture of the application

We have chosen the real-time executive OSEKTurbo OS/MPC5xx of Metrowerks for our application. This RTOS is conforming to the standard OSEK/VDX [Osek1][Osek2]; standard defined for applications with limited resources [OSM3]. The OSEK/VDX executives are light because they are based on a static description of all the system using the OIL (OSEK Implementation Language). The system is described as OSEK objects:

- Tasks: each task has a static priority; a value of 0 is the lowest possible priority for a task. Two different kinds of tasks are provided by OSEK; the basic tasks (BT) and the extended tasks (ET). The extended tasks have 4 possible states ("running", "ready", "suspended", "waiting"); while the basic tasks have only 3 possible states ("running", "ready", "suspended").
- Resources: the resources correspond to mutex. The priority ceiling protocol [Sha90] is used to avoid the priority inversion, and deadlocks.
- Events: events are synchronization tools. It enables to initiate the transitions from or to the "waiting" state. The events are assigned to extended tasks. Each extended task has a definite number of events; this task is called "owner of the event". Events can be used to communicate information to the task to which they are assigned.
- Messages: the messages enable communication of type N transmitters and m receivers.

Apart the initialisation task, there are 12 tasks in the control system (see Table 1). The priorities of the tasks have been assigned following a Deadline Monotonic policy [LL73]. Note that the value L=120 (resp. L=3, L=10) corresponds to the number of times the task has to be activated in order to acquire a frame.

Tasks	Period	WCET		deadline	Priority
	(in microsecond)				
Monitoring (1)	200000	60		200000	1
Acq PWM (2)	20000	24		10000	7
Transmit Grd (3)	50000	3360		30000	5
Deliver Cmd (4)	20000	40		10000	6
Navigation (5)	250000	560		140000	2
ReguleAttitude (6)	60000	32400		60000	4
Acq GPS (7)	250000	100	L=120	160	11
Acq IMU (8)	20000	96	L=3	720	10
Acq Instruction(9)	100000	12	L=10	80	12
TreatGPS (10)	250000	3000		5000	9
TreatIMU (11)	20000	900		7500	8
TreatInstruction (12)	100000	900		70000	3

Table1: task system of the UAV

This kind of application can't be validated if the offsets are not taken into account. Indeed, it appears clearly that task TreatGPS is released when the whole GPS frame has been received; it cannot thus be released at the same time as the task Acq GPS; it is the same case for task TreatIMU and the task Acq IMU; the same situation occurs for the task TreatInstruction and the task Acq Instruction.

The Figure 3 presents a model of a serial transaction, L_i instances of the acquisition of a part of a frame are separated by a duration corresponding to the arrival rate of the packets (Acq GPS, Acq IMU, Acq Instruction), and a longer task is used to handle the whole frame (TreatGPS, TreatIMU, TreatInstruction). In a serial transaction, the acquisition tasks are usually short, because they only have to bufferize the packets until the whole frame is built, while the treatment tasks are longer since they have to deal with the full frame.

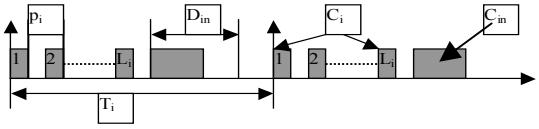


Fig. 3. pattern of serial transaction

In order to define a serial transaction as a particular case of a transaction, let us first give a survey of definitions and results found in [Tin94][TN04a][PG98].

3 Transactions

The model of tasks with offsets was proposed by Tindell in order to reduce existing pessimism of the schedulability analysis when the critical instant for a task occurs when it is released at the same time as all the tasks of higher priority. Indeed, certain tasks can for example have the same period and be bound by relations of offsets i.e. they can never be released at the same time. A set of tasks of the same period bounded by offset is called a transaction. A task system is compound of a set of transactions [PG98][TN04a]:

$\Gamma := \{\Gamma_1, \Gamma_2, \ldots \Gamma_k\}$

A transaction (see Figure 4) Γ_i contains $|\Gamma_i|$ tasks having the same period T_i : $\Gamma_i := <\{\tau_{i1}, \ldots, \tau_{i|\Gamma i|}\}, T_i>$.

A task is defined by $\tau_{ij} := <C_{ij}, O_{ij}, D_{ij}, J_{ij}, B_{ij}, P_{ij}>$ where C_{ij} is the worst-case execution time (WCET), O_{ij} is the offset (minimal time between the release of the transaction and the release of the task), but it is equivalent to consider $\Phi_{ij} = O_{ij} \% T_i$, D_{ij} is the relative deadline, J_{ij} the maximum jitter (giving t_0 the release date of an instance of the transaction Γ_i, then the task τ_{ij} is released between t_0+O_{ij} and $t_0+O_{ij}+J_{ij}$), B_{ij} maximum blocking due to lower priority tasks, and P_{ij} the priority. Without loss of generality, we consider that the tasks ordered by non decreasing offsets Φ_{ij}; in our case, we define the response time as being the time between the release of the task and the completion of the task.

Let us note also $hp_i(\tau_{ua})$ the set of indices of the tasks of Γ_i with a priority higher than the priority of a task τ_{ua} i.e. $j \in hp_i(\tau_{ua})$ if and only if $P_{ij} > P_{ua}$.

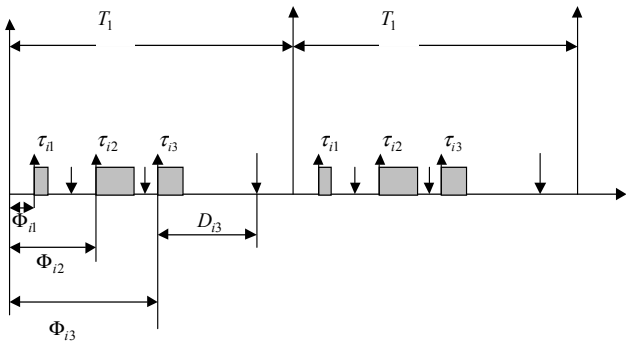


Fig. 4. model of tasks with offsets

The RTA method is to be applied on each task of the transactions. The task under analysis is usually noted τ_{ua}. Tindell showed that the critical instant of τ_{ua} is a particular instant when it is released at the same time as one task of higher priority in each transaction (its own transaction being handled separately). The main difficulty is to determine what is the critical instant candidate τ_{ic} of a transaction Γ_i that initiates the critical instant of τ_{ua}. An exact calculation method would require (until someone would proposes a better way) to evaluate the response time obtained by carrying out all the possible combinations of the tasks of priority higher than τ_{ua} in each transaction and to choose the task τ_{ic} in each transaction that leads to the worst-case response time. This exhaustive method has an exponential complexity and is intractable for realistic task systems; several approximation methods giving an upper bound of the worst-case response time have been proposed.

3.1 Upper bound method based on the interference "released for execution"

[Tin94][PG98] Let us note τ_{ic} the task of Γ_i that coincides with the critical instant of τ_{ua}. Let us note $I(\tau_{ua}, \tau_{ic}, t)$ the interference of Γ_i on the response time of τ_{ua} during a time interval of length t when τ_{ic} is released at the same instant as τ_{ua}. $I(\tau_{ua}, \tau_{ic}, t) = \sum_{\tau_{ij} \in hp_i(\tau_{ua})} \left(\left\lceil \frac{t'}{T_i} \right\rceil * C_{ij} \right)$

$t' = t - phase(\tau_{ij}, \tau_{ic})$

$phase(\tau_{ij}, \tau_{ic}) = (O_{ij} - O_{ic}) \% T_i$

t' represents the time during which τ_{ij} can interfere with τ_{ua}.

Let us note $A(\tau_{ua}, \Gamma_i, t) = \max_{\tau_{ic} \in \Gamma_i} I(\tau_{ua}, \tau_{ic}, t)$

The upper bound of the response time is

$R_{ua} = C_{ua} + \sum_{i \in \Gamma} A(\tau_k, \Gamma_i, R_{ua})$.

The value of R_{ua} is thus obtained by a classic iteration lookup.

The interference that a transaction imposes on a lower priority task can be represented by a periodic and static pattern. [TN04a] proposed an optimisation of the computation of the interference. This technique consists in storing in a table the parameters of the function of interference of a transaction on a task of lower priority. The logic of this approach is that the computation of the interference during a time interval of length t is related to two terms: the number of periods T_i of the transaction Γ_i during the time interval of length t and the remainder of the division of t per the period T_i (noted $t\%T_i$ in the sequel). This approach reduces the computation time but this method does not reduce the difference between the real worst-case response time

and the upper bound obtained. Therefore, we couldn't validate our system with the general method because the tasks (2), (4) and (11) have a worst-case response time greater than their relative deadline; while the real worst-case response time of all the tasks of the set could in fact be lower than their deadline. (see Table2).

Tasks	Period	deadline	Priority	"release for execution" method
1	200000	200000	1	56056
2	20000	10000	7	11332
3	50000	30000	5	23784
4	20000	10000	6	11672
5	250000	140000	2	56096
6	60000	60000	4	54636
7	250000	160	11	124
8	20000	720	10	468
9	100000	80	12	12
10	250000	5000	9	3408
11	20000	7500	8	10720
12	100000	70000	3	55416

Table2: upper bound on response times with "released for execution" interference

We thus present a method given in [TN04b] giving a tighter upper bound.

3.2 Upper bound method based on the "imposed" interference

[TN04b] This method removes the unnecessary overestimation taken into account in the computation of the interference created by a task on a lower priority one. This overestimation does not have any impact in the case of tasks without offset but has a considerable effect in the approximation of the worst-case response time when we are in the presence of tasks with offsets. This method consists in calculating the interference effectively imposed by a task τ_j on a task τ_{ua} with a lower priority during a time interval of length t; the idea is that the interference cannot exceed the interval of time t.

$$\frac{dInterférence_j(t)}{dt} \leq \frac{dt}{dt}$$

In order to calculate this "imposed" interference, [TN04b] substracts a parameter x (see Figure 5) from the original interference formula; let us note $W_{ic}(\tau_{ua}, t)$ the interference that Γ_i imposes on the response time of τ_{ua} during a time interval of length t when τ_{ic} is released at the same instant as τ_{ua} .

$$W_{ic}(\tau_{ua}, \mathrm{t}) = \sum_{j \in hp_i(\tau_{ua})} \left(\left(\left\lfloor \frac{t'}{T_i} \right\rfloor + 1 \right) * C_{ij} - \mathrm{x_{ijc}} \right)$$

$$t' = t - phase(\tau_{ij}, \tau_{ic})$$

$$phase(\tau_{ij}, \tau_{ic}) = (\mathrm{O_{ij}} - \mathrm{O_{ic}}) \,\%\, \mathrm{T_i}$$

$$x_{ijc} = \begin{cases} 0 & \text{for } t' < 0 \\ \max\left(0, \mathrm{C_{ij}} - (t' \,\%\, \mathrm{T_i})\right) & \text{for } t' \geq 0 \end{cases}$$

x_{ijc} corresponds to the part of the task τ_{ij} that cannot be executed in the time interval of length t; since this interference is not effectively imposed in this interval, it is not taken into account.

Example: transaction de 4 tâches avec période de 50

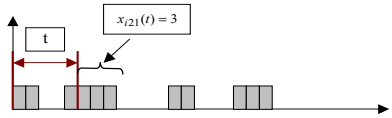


Fig.5. "imposed" interference

$$\Gamma_i := < \{\tau_{i1}, \tau_{i2}, \tau_{i3}, \tau_{i4}\}, 50 >$$

$$\tau_{i1} := < 2, 0, 4, 0, 0, 4 >$$

$$\tau_{i2} := < 4, 4, 8, 0, 0, 2 >$$

$$\tau_{i3} := < 2, 12, 5, 0, 0, 3 >$$

$$\tau_{i4} := < 3, 17, 15, 0, 0, 1 >$$

$$W_{i1}(\tau_{ua}, 5) = (2 - 0) + (4 - 3) + (0 - 0) + (0 - 0) = 3$$

For determining the upper bound of the response-time, we use this function :

$$W_i(\tau_{ua}, t) = \max_{c \in hp_i(\tau_{ua})} W_{ic}(\tau_{ua}, \mathrm{t})$$

With the value of each $W_i(\tau_{ua}, t)$, the response time R_{ua} of τ_{ua} can be calculated. $R_{ua} = C_{ua} + \sum_{i \in \Gamma} W_i(\tau_{ua}, R_{ua})$. R_{ua} is obtained by fix-point iteration starting with $R_{ua} = 0$. Let us execute this method on the example (Figure 6 (a))

Fig. 6.(a and b) Example for imposed interference (a) reverse transaction (b)

In the transaction Γ_i, we have five tasks. Let us consider a lower priority task τ_{ua} with $C_{ua} = 5$. Let us calculate the value of response-time.

Iteration 1:

$$W_{i1}(\tau_{ua}, 0) = 0 \quad W_{i2}(\tau_{ua}, 0) = 0 \quad W_{i3}(\tau_{ua}, 0) = 0$$

$$W_{i4}(\tau_{ua}, 0) = 0 \quad W_i(\tau_{ua}, 0) = 0 \quad R_{ua} = 5$$

Iteration 2:

$W_{i1}(\tau_{ua},5) = (2-0)+(2-1)+(0-0)+(0-0)+(0-0) = 3$

$W_{i2}(\tau_{ua},5) = (0-0)+(2-0)+(2-1)+(0-0)+(0-0) = 3$

$W_{i3}(\tau_{ua},5) = (0-0)+(0-0)+(2-0)+(2-1)+(0-0) = 3$

$W_{i4}(\tau_{ua},5) = (0-0)+(0-0)+(0-0)+(2-0)+(4-3) = 3$

$W_{i5}(\tau_{ua},5) = (0-0)+(0-0)+(0-0)+(0-0)+(4-0) = 4$

$W_i(\tau_{ua},0) = 4 \qquad R_{ua} = 9$

We resume in the table below the values obtained in the next iterations:

Iteration number	t	W_{i1}	W_{i2}	W_{i3}	W_{i4}	W_{i5}	W_i	R_{ua}
1	0	0	0	0	0	0	0	5
2	5	3	3	3	3	4	4	9
3	9	5	5	5	6	5	6	11
4	11	6	6	7	6	6	7	12
5	12	6	6	8	6	6	8	13
6	13	7	7	8	7	7	8	13

Consequently, the value of R_{ua} is equal to 13.

It is not simple to evaluate the value of imposed interference. Indeed, with this method it is necessary in each iteration to evaluate the value of "n" interferences with "n" as the number of tasks in the transaction. Moreover, it is necessary to evaluate the value of " x_{ijc} " "n" times in each iteration. In order to simplify the calculation of the value of imposed interference, we use the transaction presented in figure 6(b). We call this transaction the reverse transaction Γ_i^{-1} of the transaction Γ_i. With the reverse transaction, we show in the next section 4.3 that it is sufficient to calculate only the value of $W_{i^{-1}1}(\tau_{ua},t)$ at each iteration.

We present in the table below the values obtained in the different iterations:

Iteration number	t	$W_{i^{-1}1}$	R_{ua}
1	0	0	5
2	5	4	9
3	9	6	11
4	11	7	12
5	12	8	13
6	13	8	13

4 Contribution to RTA of transactions

4-1 Definitions

Now let us introduce the definition of a serial transaction:

Definition 1: A serial transaction is a transaction with the following constraints(Figure 3, figure 6(a), figure 8 and figure 9):

Let Γ_I be a serial transaction,

- null jitter: $\forall i/\tau_{ij} \in \Gamma_i$, $J_{ij}=0$
- regular arrival pattern p_i: $\forall j \in [1..|\Gamma_i|]$, $\Phi_{ij}=(j-1)p_i$.
- there are two kinds of tasks :
  - the $L_i=|\Gamma_i|-1$ acquisition tasks such that : $\tau_{ij,j\in[1..Li]} := <C_i, (j-1)p_i, p_i, 0, B_{ij}, P_i>$;
  - the treatment task $\tau_{i|\Gamma i|}:=<C_{in},L_ip_i,D_{in}, 0, B_{ij}, P_{in}>$
- with $C_{in}>C_i$, $D_{in}>p_i$,$P_{in}<P_i$ and $(T_i - L_i \cdot p_i) - C_{in} > p_i - C_{in}$. This means that the treatment task is longer than the acquisition tasks, but is provided a longer deadline and a lower priority.

Definition 2 : a task τ_{ua} is an intermediate priority task for a serial transaction Γ_i if the priority of τ_{ua} is lower than acquisition tasks of Γ_i but higher than treatment task of Γ_i.

Definition 3 : a task τ_{ua} is a lower priority task for a serial transaction Γ_i if the priority of τ_{ua} is lower than all the tasks of Γ_i.

Definition 4: Let Γ_i be a serial transaction, we call reverse transaction of the serial transaction Γ_i the transaction Γ_i^{-1} obtained by putting in first position the task of last position of Γ_i; the other parameters remain identical (period, offsets between tasks, etc.) (see figure 6 (a and b)). The tasks of Γ_i^{-1} are defined as:

$$\tau_{i^{-1}1} =< C_{in}, 0, C_{in} + (p_i - C_i), 0, B_{i|\Gamma_i|}, P_{i|\Gamma_i|} >$$

$$\tau_{i^{-1}j, j\in[2..|\Gamma_i|]} =< C_i, C_{in} - C_i + (j-1)\cdot p_i, p_i, 0, B_{i(j-1)}, P_{i(j-1)} >$$

4-2 schedulability of intermediate priority task

In this section, we first simplify the way to compute the interference [PG 98] for general transactions with no jitter. Then we use this simplification in order to show that a task τ_{ua} of intermediate priority has to suffer the biggest interference when it's released at the same time as the first acquisition task, whatever the length of the busy period is.

Theorem 1: with null jitters, it is possible to establish the task of a transaction Γ_i that leads to the worst "released for execution" interference during a time interval t in testing $|hp_i(\tau_{ua})|^2$ intervals.

Proof: according to [PG98] the interference of a transaction for a task τ_{ic} candidate to coincide with the critical instant is given by:

$$W_{ic}(\tau_{ua},t) = \sum_{\forall j \in hp_i(\tau_{ua})} \left(I_{ijc}^{Set1} + I_{ijc}^{Set2}(t) \right) \qquad \text{with} \qquad I_{ijc}^{Set1} = \left\lfloor \frac{J_{ij} + \Phi_{ijc}}{T_i} \right\rfloor C_{ij},$$

$$I_{ijc}^{Set2} = \left\lceil \frac{t - \Phi_{ijc}}{T_i} \right\rceil C_{ij}, \text{ and } \Phi_{ijc} = (T_i + O_{ij} - (O_{ic} + J_{ic}))\% T_i$$

By assumption, the jitter is null, so the interference is written :

$$W_{ic}(\tau_{ua},t) = \sum_{\forall j \in hp_i(\tau_{ua})} \left(\left\lfloor \frac{(T_i + O_{ij} - O_{ic})\% T_i}{T_i} \right\rfloor C_{ij} + \left\lceil \frac{t - (T_i + O_{ij} - O_{ic})\% T_i}{T_i} \right\rceil C_{ij} \right)$$

By definition, $(T_i + O_{ij} - O_{ic})\% T_i < T_i$ therefore

$$W_{ic}(\tau_{ua},t) = \sum_{\forall j \in hp_i(\tau_{ua})} \left\lceil \frac{t - (T_i + O_{ij} - O_{ic})\% T_i}{T_i} \right\rceil C_{ij}$$

Which is equivalent to

$$W_{ic}(\tau_{ua},t) = \sum_{\forall j \in hp_i(\tau_{ua})} \left\lceil \frac{t - (T_i + \Phi_{ij} - \Phi_{ic})\% T_i}{T_i} \right\rceil C_{ij}$$

Let us note $k_1, k_2, \ldots, k_{|hpi(\tau ua)|}$ the indices ordered by offset of $hp_i(\tau_{ua})$ (i.e. p<q => $\Phi_{ikp} \leq \Phi_{ikq}$). Since the offsets are assumed to be lower than the period, $(T_i+\Phi_{ij}-\Phi_{ic})\%T_i$ correspond to $\Phi_{ij}-\Phi_{ic}$ if $\Phi_{ic} \leq \Phi_{ij}$ and $(T_i+\Phi_{ij}-\Phi_{ic})$ if $\Phi_{ij} < \Phi_{ic}$. Hence, separating the formula between tasks released before and after the critical instant candidate τ_{ikp}, we have :

$$W_{ik_p}(\tau_{ua},t) = \sum_{\substack{k_j \in hp_i(\tau_{ua}) \\ k_j < k_p}} \left\lceil \frac{t - (T_i + \Phi_{ik_j} - \Phi_{ik_p})}{T_i} \right\rceil C_{ik_j} + \sum_{\substack{k_j \in hp_i(\tau_{ua}) \\ k_j \geq k_p}} \left\lceil \frac{t - (\Phi_{ik_j} - \Phi_{ik_p})}{T_i} \right\rceil C_{ik_j}$$

$$\text{so } W_{ik_1}(\tau_{ua},t) = \sum_{k_j \in hp_i(\tau_{ua})} \left\lceil \frac{t - (\Phi_{ik_j} - \Phi_{ik_1})}{T_i} \right\rceil C_{ik_j}$$

$$W_{ik_2}(\tau_{ua},t) = \left\lceil \frac{t - (T_i + \Phi_{ik_1} - \Phi_{ik_2})}{T_i} \right\rceil C_{ik_1} + \sum_{\substack{k_j \in hp_i(\tau_{ua}) \\ k_j \geq k_2}} \left\lceil \frac{t - (\Phi_{ik_j} - \Phi_{ik_2})}{T_i} \right\rceil C_{ik_j}$$

And so on. Therefore

$$W_{ik_1}(\tau_{ua},t)-W_{ik_2}(\tau_{ua},t)=\left(\left\lceil\frac{t-(\Phi_{ik_1}-\Phi_{ik_1})}{T_i}\right\rceil-\left\lceil\frac{t-(T_i+\Phi_{ik_1}-\Phi_{ik_2})}{T_i}\right\rceil\right)C_{ik_1}+$$
$$\sum_{\substack{k_j\in hp_i(\tau_{ua})\\ k_j\geq k_2}}\left(\left\lceil\frac{t-(\Phi_{ik_j}-\Phi_{ik_1})}{T_i}\right\rceil-\left\lceil\frac{t-(\Phi_{ik_j}-\Phi_{ik_2})}{T_i}\right\rceil\right)C_{ik_j}$$

Let us analyze now, how we can determine efficiently the differences between the interference function when comparing the first task as the critical instant candidate comparing to another task :

$\left(\left\lceil\frac{t}{T_i}\right\rceil-\left\lceil\frac{t-(T_i+\Phi_{ik_1}-\Phi_{ik_2})}{T_i}\right\rceil\right)C_{ik_1}$ is always equal to 0 or C_{ik1} because $\Phi_{ij}<T_i$.

The difference is C_{ik1} if and only if :

$t\%T_i>0\ and\ t\%T_i-(T_i+\Phi_{ik_1}-\Phi_{ik_2})\leq 0$,equivalently

$t\%T_i\in]0..T_i+\Phi_{ik_1}-\Phi_{ik_2}]$

For the other tasks interference (i.e. other part of the sum) :

$\left(\left\lceil\frac{t-(\Phi_{ik_j}-\Phi_{ik_1})}{T_i}\right\rceil-\left\lceil\frac{t-(\Phi_{ik_j}-\Phi_{ik_2})}{T_i}\right\rceil\right)C_{ik_j}$ is always equal to 0 or $-C_{ikj}$ because $\Phi_{ij}<T_i$.

The difference is equal to $-C_{ikj}$ if and only if :

$t\%T_i-(\Phi_{ik_j}-\Phi_{ik_1})\leq 0\ and\ t\%T_i-(\Phi_{ik_j}-\Phi_{ik_2})>0$

equivalently if : $t\%T_i\in]\Phi_{ik_j}-\Phi_{ik_2}..\Phi_{ik_j}-\Phi_{ik_1}]$

We can thus calculate $W_{ik_1}(\tau_{ua},t)-W_{ik_2}(\tau_{ua},t)$ testing $|hp_i(\tau_{ua})|$ intervals.

We will now calculate the difference $\forall k_p\in hp_i(\tau_{ua}),k_p\neq k_1,W_{ik_1}(\tau_{ua},t)-W_{ik_p}(\tau_{ua},t)$:

$$W_{ik_1}(\tau_{ua},t)-W_{ik_p}(\tau_{ua},t)=\sum_{\substack{k_j\in hp_i(\tau_{ua})\\ k_j<k_p}}\left(\left\lceil\frac{t-(\Phi_{ik_j}-\Phi_{ik_1})}{T_i}\right\rceil-\left\lceil\frac{t-(T_i+\Phi_{ik_j}-\Phi_{ik_p})}{T_i}\right\rceil\right)C_{ik_j}+$$
$$\sum_{\substack{k_j\in hp_i(\tau_{ua})\\ k_j\geq k_p}}\left(\left\lceil\frac{t-(\Phi_{ik_j}-\Phi_{ik_1})}{T_i}\right\rceil-\left\lceil\frac{t-(\Phi_{ik_j}-\Phi_{ik_p})}{T_i}\right\rceil\right)C_{ik_j}$$

The first sum has a value ≥ 0 whereas the second has a value ≤ 0. We have :

Difference of $+C_{ikj}$ for $k_j<k_p$ if $t\%T_i\in]\Phi_{ik_j}-\Phi_{ik_1}..T_i+\Phi_{ik_j}-\Phi_{ik_p}]$ (1)

Difference of $-C_{ikj}$ for $k_j\geq k_p$ if $t\%T_i\in]\Phi_{ik_j}-\Phi_{ik_p}..\Phi_{ik_j}-\Phi_{ik_1}]$ (2)

We can thus obtain the difference $W_{ik_1}(\tau_{ua},t) - W_{ik_p}(\tau_{ua},t)$ in testing $|hp_i(\tau_{ua})|$ intervals, therefore, testing $|hp_i(\tau_{ua})|^2$ intervals is enough to calculate the critical instant candidate. In fact, if the difference $W_{ik_1}(\tau_{ua},t) - W_{ik_p}(\tau_{ua},t)$ is always ≥0, then the release of τ_{ik1} leads to the worst interference, otherwise, it is the release of the task τ_{ikp} that leads to the worst interference.

The next result relies on the intervals defined in theorem 1, let us define E_{ikj} as the shift between too successive tasks of higher priority than the task under analysis:

$E_{ij} = \Phi_{ik} - \Phi_{ij}$ for $j \in hp_i(\tau_{ua})$ if j has a successor k in $hp_i(\tau_{ua})$ and

$E_{i|hp_i(\tau_{ua})|} = T_i + \Phi_{i1} - \Phi_{i|hp_i(\tau_{ua})|}$

(see figure 7)

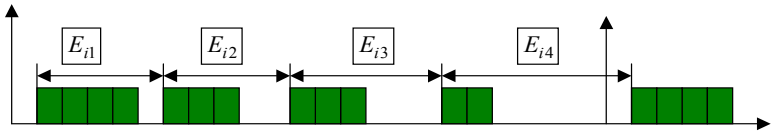


Fig.7. Illustration of E_{ij} and theorem 2

Theorem 2 shows that for specific patterns of transactions without offset where the WCET of tasks are non increasing and the shift between successive offsets is non decreasing, the critical instant of a task always coincides to the first instance of the transaction (Figure 7). The acquisition tasks of a serial transaction follow this kind of pattern, therefore the critical instant of a task of an intermediate priority (lower than acquisition tasks but higher than treatment task) always coincides with the first acquisition task.

Theorem 2 : let Γ_i be a transaction, τ_{ua} a task under analysis. If the jitters are null and if the tasks of Γ_i are such that their WCET is non-increasing, i.e. $C_{ij} \geq C_{ik}$ $\forall (j \leq k) \in hp_i(\tau_{ua})$, and offset shifting are non-decreasing i.e. $E_{ij} \leq E_{ik}$ $\forall (j \leq k) \in hp_i(\tau_{ua})$ then the critical instant of τ_{ua} coincide with the release of the first task of $hp_i(\tau_{ua})$.

Proof : the proof is based on the interferences. According to the definition of E_{ij}, $\sum E_{ij} = T_i$ .We have proved in Theorem1 that the difference of interference between a candidate k_p and the candidate k_1 was obtained for every $k_j \in hp_i(\tau_{ua})$ by :

Difference of $+C_{ikj}$ for $k_j < k_p$ if $t\%T_i \in]\Phi_{ik_j} - \Phi_{ik_1} .. T_i + \Phi_{ik_j} - \Phi_{ik_p}]$ (1)

Difference $-C_{ikj}$ for $k_j \geq k_p$ if $t\%T_i \in]\Phi_{ik_j} - \Phi_{ik_p} .. \Phi_{ik_j} - \Phi_{ik_1}]$ (2)

Let us analyze these intervals in the context C_{ij} non-increasing and E_{ij} non-decreasing ; let us compare the candidates k_1 and k_2 :

$k_j=k_1$	Difference $+C_{ik1}$ for $t\%T_i \in]0..T_i + \Phi_{ik_1} - \Phi_{ik_2}]$ i.e. for $t\%T_i \in]0..T_i - E_{ik_1}]$, let us note I_{ik2k1} this interval
$k_j=k_2$	Difference of $-C_{ik2}$ for $t\%T_i \in]0..\Phi_{ik_2} - \Phi_{ik_1}]$ i.e. for $t\%T_i \in]0..E_{ik_1}]$, let us note I_{ik2k2} this interval
$k_j=k_3$	Difference of $-C_{ik3}$ for $t\%T_i \in]\Phi_{ik_3} - \Phi_{ik_2}..\Phi_{ik_3} - \Phi_{ik_1}]$, $t\%T_i \in]E_{ik_2}..E_{ik_1} + E_{ik_2}]$, let us note I_{ik2k3} this interval
$k_j=k_n$	Difference of $-C_{ikn}$ for $t\%T_i \in]\Phi_{ik_n} - \Phi_{ik_2}..\Phi_{ik_n} - \Phi_{ik_1}]$ i.e. for $t\%T_i \in]E_{ik_2} + E_{ik_3} + ... + E_{ik_{n-1}}..E_{ik_1} + E_{ik_2} + E_{ik_3} + ... + E_{ik_{n-1}}]$, let us note I_{ik2kn} this interval

We will prove now that with our constraints, the intersection of the intervals giving a negative difference is empty, i.e. there is at most one negative value for any value of $t\%T_i$; and then if $t\%T_i$ is in an interval giving a negative value, in such a case we are in an interval giving a positive value. Therefore, we will prove that either there is not any difference of interference (neither negative nor positive) or there is at most one negative value but in this case there is a positive difference that is greater or equal to the negative difference (since its value is C_{ik1}). In the proof, an interval I is < (lower) than an interval J if any value of I is lower than any value of J.

$I_{ik2k2} < I_{ik2k3}$ because $E_{ik1} \leq E_{ik2}$

$I_{ik2k3} < I_{ik2k4}$ because $E_{ik1}+E_{ik2} \leq E_{ik2}+E_{ik3}$ because $E_{ik1} \leq E_{ik3}$

…

$I_{ik2kn-1} < I_{ik2kn}$ because

$E_{ik1}+E_{ik2}+\ldots+E_{ikn-2} \leq E_{ik2}+E_{ik3}+\ldots+E_{ikn-1}$ because $E_{ik1} \leq E_{ikn-1}$

Consequently, the intersection of the negative intervals is empty.

Finally, we will prove that if t is in one of the intervals $I_{ik2kp,\ p\in 2..kn}$, then it is in the interval I_{ik2k1} .

Let us suppose that $t\%T_i \notin I_{ik2k1}$, this means $t\%T_i \in]T_i-E_{ik1}..T_i[\cup \{0\}$.

If $t\%T_i=0$, then t is not element of any interval

In the case $t\%T_i \in]T_i-E_{ik1}..T_i[$, we will prove that T_i-E_{ik1} is greater than any other interval $I_{ik2kj,j=2..kn}$. It is sufficient for this proof, since the intervals are increasing, to prove that $T_i-E_{ik1} \geq E_{ik1}+E_{ik2}+\ldots+E_{ikn-1}$. So, we have to prove that $T_i \geq 2E_{ik1}+E_{ik2}+\ldots+E_{ikn-1}$; since by definition $T_i=E_{ik1}+E_{ik2}+\ldots+E_{ikn}$, therefore we have to prove that $E_{ik1}+E_{ik2}+\ldots+E_{ikn-1}+E_{ikn} \geq 2E_{ik1}+E_{ik2}+\ldots+E_{ikn-1}$, this is true because $E_{ikn} \geq E_{ik1}$.

Let us generalize to a task k_p of the serial transaction:

$k_j=k_1$	Difference $+C_{ik1}$ for $t\%T_i \in]0..T_i + \Phi_{ik_1} - \Phi_{ik_p}]$ i.e. for $t\%T_i \in]0..T_i - (E_{ik_1} + E_{ik_2} + ... + E_{ik_{p-1}})]$ since $T_i=\Sigma E_{ij}$, $t\%T_i \in]0..E_{ik_p} + E_{ik_{p+1}} + ... + E_{ik_n})]$ let us note I_{ikpk1} this interval
$k_j=k_2$	Difference $+C_{ik2}$ for $t\%T_i \in]\Phi_{ik_2} - \Phi_{ik_1} ..T_i + \Phi_{ik_2} - \Phi_{ik_p}]$ i.e. for $t\%T_i \in]E_{ik_1} ..T_i - (E_{ik_2} + ... + E_{ik_{p-1}})]$ since $T_i=\Sigma E_{ij}$, $t\%T_i \in]E_{ik_1} ..E_{ik_1} + E_{ik_p} + E_{ik_{p+1}} + ... + E_{ik_n})]$ let us note I_{ikpk2} this interval
$k_j=k_p$	Difference of $-C_{ikp}$ for $t\%T_i \in]0..\Phi_{ik_j} - \Phi_{ik_1}]$ i.e. for $t\%T_i \in]0..E_{ik_1} + ... + E_{ik_{p-1}}]$, let us note I_{ikpkp} this interval
$k_j=k_p+1$	Difference of $-C_{ikp+1}$ for $t\%T_i \in]\Phi_{ik_p+1} - \Phi_{ik_p} ..\Phi_{ik_p+1} - \Phi_{ik_1}]$ i.e. for $t\%T_i \in]E_{ik_p} ..E_{ik_1} + ... + E_{ik_p}]$, let us note $I_{ikpkp+1}$ this interval
$k_j=k_n$	Difference of $-C_{ikn}$ for $t\%T_i \in]\Phi_{ik_n} - \Phi_{ik_p} ..\Phi_{ik_n} - \Phi_{ik_1}]$ i.e. for $t\%T_i \in]E_{ik_p} + E_{ik_{p+1}} + ... + E_{ik_{n-1}} ..E_{ik_1} + E_{ik_2} + E_{ik_3} + ... + E_{ik_{n-1}}]$, let us note I_{ikpkn} this interval

The proof uses the same way as before, except that for the general case, what we show is that there are always at least as many positive interval than negative intervals. Since the WCET can't decrease, and that the positive intervals correspond to the first tasks of the transaction, the positive difference is always greater or equal than the negative difference.

- $t\%T_i \in]0..E_{ik1}]$: $t\%T_i \in I_{ikpk1}$ and $t\%T_i \in I_{ikpkp}$, and $\forall k_q>k_p$, $t\%T_i \notin I_{ikpkq}$ because the lower limit of these intervals is greater than $E_{ikp} \geq E_{ik1}$. So, there is at least one positive interval (giving a difference of C_{i1}) and at most one negative interval (giving a difference of C_{ikp}) and since $C_{i1} \geq C_{ikp}$, we obtain W_{i1} (τ_{ua}, $t\%T_i \in]0..E_{ik1}]$)- W_{ip} (τ_{ua}, $t\%T_i \in]0..E_{ik1}]$)≥ 0
- $t\%T_i \in]E_{ik1}.. E_{ik1}+E_{ik2}]$: $t\%T_i \in I_{ikpk2}$ (positive intervals), $t\%T_i \in I_{ikpkp}$ (negative interval). It is possible that $t\%T_i \in I_{ikpkp+1}$ (negative interval), but in this case, $t\%Ti \in I_{ikpk1}$ (positive interval). On the contrary, $\forall k_q>k_{p+1}$, $t\%T_i \notin I_{ikpkq}$ because $E_{ikp}+E_{ikp+1} \geq E_{ik1}+E_{ik2}$. Since the execution times are nonincreasing, we have $W_{i1}(\tau_{ua}, t\%T_i \in]E_{ik1}.. E_{ik1}+E_{ik2}])$- $W_{ip}(\tau_{ua}, t\%T_i \in]E_{ik1}.. E_{ik1}+E_{ik2}]) \geq 0$
- the same reasonning can be lead on the other possible intervals for $t\%T_i$ for every interval of length Eik_j.

Theroem 2 implies that in order to analyse a task of intermediate priority compared to a serial transaction, it is sufficient to test its response time when it's released at the same time as the first task of the serial transaction to obtain its tight worst response-time with a classic response time analysis. Note that it can't be applied to a task of lower priority than all the transaction, because the condition "non increasing WCET" is not satisfied in this case.

Let us note $it(\tau_{ua})$ the set of indices of the transactions to whom τ_{ua} is an intermediate priority task. By applying Theorem 2, the interference applied by the serial transactions belonging to $it(\tau_{ua})$ does not need any specific study related to transactions. It is given (tight upper bound) by:

$$\sum_{j \in it(\tau_{ua})} \left(\left\lfloor \frac{W_{ua}(t)}{p_j} \right\rfloor \cdot L_j + \min\left(\left\lceil \frac{W_{ua}(t) \% p_j}{D_j} \right\rceil, L_j \right) \right) \cdot C_j$$

4-3 schedulability of lower priority task

We present formally the concept of reverse transaction used in the example of section 3.2 (Figure 6 (a) and (b)).

Theorem 3: Let Γ_i be a serial transaction, let Γ_i^{-1} be its reverse transaction and τ_{ua} a task under analysis. If all the tasks of the serial transaction Γ_i have a priority higher than the priority of τ_{ua}, then the interference imposed by the serial transaction Γ_i on the task τ_{ua} when it is released at the same time as the task initiating the critical instant in Γ_i has exactly the same value as the interference imposed by Γ_i^{-1} on τ_{ua} when τ_{ua} is released at the same time as the first task in transaction Γ_i^{-1} i.e $W_{i^{-1}1}(\tau_{ua}, t) = W_i(\tau_{ua}, t)$ for any t.

Proof: : Let us note $f_i = T_i - L_i \cdot p_i$; and $W_{i^{-1}1}(\tau_{ua}, t)$ the imposed interference on the task τ_{ua} by the transaction Γ_i^{-1} in a time interval of length t.

We thus will calculate $W_{i^{-1}1}(\tau_{ua}, t) - W_{ic}(\tau_{ua}, t)$.

For any time interval of length t, we know that there is an integer k such that $t = k \cdot T_i + t \,\%\, T_i$.

According to [TN04a], $W_{i^{-1}1}(\tau_{ua}, t) = W_{i^{-1}1}(\tau_{ua}, k \cdot T_i) + W_{i^{-1}1}(\tau_{ua}, t\%T_i)$

$W_{ic}(\tau_{ua}, t) = W_{ic}(\tau_{ua}, k \cdot T_i) + W_{ic}(\tau_{ua}, t\%T_i)$

Since the value of interference imposed in any time interval of length T_i (Period) is the same whatever the beginning of this interval is, then $W_{i^{-1}1}(\tau_{ua}, k \cdot T_i) = W_{ic}(\tau_{ua}, k \cdot T_i) = k \cdot (L_i \cdot C_i + C_{in})$ consequently,

$W_{i^{-1}1}(\tau_{ua},t) - W_{ic}(\tau_{ua},t) = W_{i^{-1}1}(\tau_{ua},t\%T_i) - W_{ic}(\tau_{ua},t\%T_i)$ so we can suppose $0 \le t < T_i$; with this consideration, we have :

Interval	$W_{i^{-1}1}(\tau_{ua},t)$	$W_i(\tau_{ua},t)$
$t \le C_{in}$	$W_{i^{-1}1}(\tau_{ua},t) = t$	$W_{in}(\tau_{ua},t) = t$
$t \ge C_{in} + L_i \cdot p_i$	$W_{i^{-1}1}(\tau_{ua},t) = C_{in} + L_i \cdot C_i$	$W_{i1}(\tau_{ua},t) = C_{in} + L_i \cdot C_i$
$C_{in} < t$ and $t < C_{in} + L_i \cdot p_i$	$W_{i^{-1}1}(\tau_{ua},t) = C_{in} + \left\lceil \frac{t-(C_{in}+(p_i-C_i))}{p_i} \right\rceil \cdot C_i - x_{i^{-1}1}(t)$	$W_i(\tau_{ua},t) = \max \left\{ \begin{matrix} W_{ic}(\tau_{ua},t) \\ \forall\ c \in [1..L_i+1] \end{matrix} \right\}$

For $t \le C_{in}$ and $C_{in} + L_i \cdot p_i \le t < T_i$, we have already $W_{i^{-1}1}(\tau_{ua},t) = W_i(\tau_{ua},t)$
We have now to prove the equality $W_{i^{-1}1}(\tau_{ua},t) = W_i(\tau_{ua},t)$ for $C_{in} < t < C_{in} + L_i \cdot p_i$.

For $t \in]C_{in} ; C_{in} + (p_i - C_i)]$, $W_{i^{-1}1}(\tau_{ua},t) = W_{i(L_i+1)}(\tau_{ua},t)$; c is equal to $L_i + 1$; and for $t \in]C_{in} + (p_i - C_i) ; C_{in} + L_i \cdot p_i[$, $\exists c \in [1..L_i]$ such as $t \in]C_{in} + (p_i - C_i) + (L_i - c) \cdot p_i ; C_{in} + (p_i - C_i) + (L_i - c + 1) \cdot p_i]$ (Figure 8) .

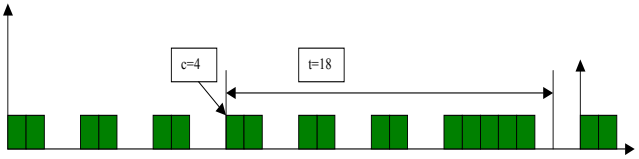


Fig.8. Illustration of determination of the value of c with t=18,
$p_i = 4$, $C_i = 2$, $C_{in} = 5$, $L_i = 6$; we find c=4

For these two cases, we have $W_{i^{-1}1}(\tau_{ua},t) = W_{ic}(\tau_{ua},t)$ **(1)**

Moreover, $W_{ic}(\tau_{ua},t) = (L_i - c + 1) \cdot C_i + (C_{in} - x_{icn}(t))$ because $f_i - C_{in} > p_i - C_i$ (according to the definition of serial transaction).

We will prove now that for all $p \in [1..L_i + 1]$, $W_{ip}(\tau_{ua},t) \le W_{ic}(\tau_{ua},t)$, i.e $W_i(\tau_{ua},t) = W_{ic}(\tau_{ua},t)$
We take the value of $W_{ic}(\tau_{ua},t)$ like reference (figure 9 (a))

1st case : p > c with c<L_i+1

It appears clearly on the figure 9 (A and B) that the shifting of the interval t from c to the position p decreases the value of the interference by $(p - c) \cdot C_i$ on the left side

whereas the increasing on the value of interference obtained on the right side is lower or equal to $(p-c)\cdot C_i$ because $f_i - C_{in} > p_i - C_i$. Therefore $W_{ip}(\tau_{ua},t) \le W_{ic}(\tau_{ua},t)$.

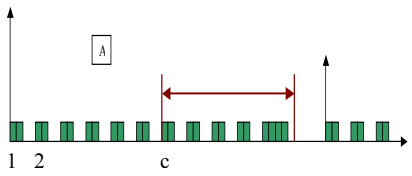
2nd case : p < c with c>1

On the figure 9 (A and C), we can see that every time the interval t is shifted the value p_i towards the left (until we reach the position p such as $t < (L_i + 1 - p)\cdot p_i$), we add C_i on the value of the interference on the left side; however, the decreasing on the value of the interference on the right side is in the interval $[C_i ; p_i]$; therefore the value of the interference after this shifting decreases. When t is lower than $(L_i + 1 - p)\cdot p_i$, a shifting of the interval t towards the left doesn't change the value of the interference (Figure 9 (D)).

Consequently, the value of $W_{ip}(\tau_{ua},t)$ is always lower or equal than the value of $W_{ic}(\tau_{ua},t)$; then

$W_i(\tau_{ua},t) = W_{ic}(\tau_{ua},t)$ **(2)**

(1) and **(2)** imply $W_{i^{-1}1}(\tau_{ua},t) = W_i(\tau_{ua},t)$



Example of Calculation of $W_{ic}(\tau_{ua},t)$ with c=7

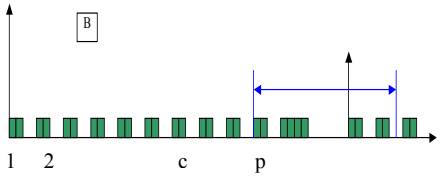
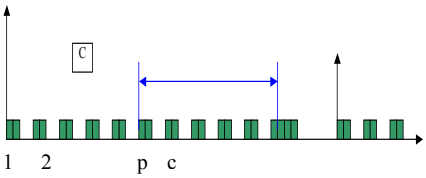
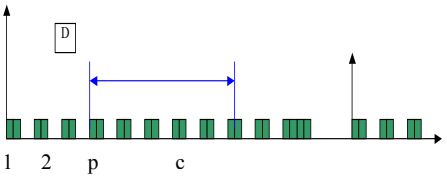


Illustration of $W_{ip}(\tau_{ua},t)$ with p>c



1st Illustration of $W_{ip}(\tau_{ua},t)$ with p<c



2nd Illustration of $W_{ip}(\tau_{ua},t)$ with p>c

Fig.9 : Illustration of the comparison between $W_{ip}(\tau_{ua},t)$ and $W_{ic}(\tau_{ua},t)$

□

Let us note $hp(\tau_{ua})$ the set of indices of serial transactions such that τ_{ua} has a lower priority than every task of the transaction.

By applying Theorem1, the interference applied by the serial transactions belonging to $hp(\tau_{ua})$ in a time interval of length t is :

$$\sum_{j\in hp(\tau_{ua})}\left(\left\lceil\frac{t}{T_j}\right\rceil\cdot C_{jn}+\left\lfloor\frac{t}{T_j}\right\rfloor\cdot L_j\cdot C_j+\min\left(\left\lceil\frac{(t\%T_j-C_{jn}-(p_j-C_j))}{p_j}\right\rceil,L_j\right)\cdot C_j\right)$$

This formula facilitates the calculation of the worst-case response-time.

5 Validation of the case study

Let S be a set of tasks that contains classical tasks and serial transactions.

Let τ_{ua} be a task under analysis with execution time equal to C_{ua}.

Let us note $hp(\tau_{ua})$ the set of indices of serial transactions such that τ_{ua} has a lower priority than every tasks of the transaction. Let us note $it(\tau_{ua})$ the set of indices of the transactions to whom τ_{ua} is an intermediate priority task, $Ch(\tau_{ua})$ the set of indices of classical tasks having a priority higher than the priority of τ_{ua} and $W_{ua}(t)$ the amount of interference that all the tasks in the set S with higher priority impose on the task τ_{ua} in a time interval of length t.

By applying theorem 2 and theorem 3, we obtain this formula :

$W_{ua}(0)=C_{ua}$

$$W_{ua}(t^{(n+1)})=C_{ua}+\sum_{j\in Ch(\tau_{ua})}\left\lceil\frac{W_{ua}(t^{(n)})}{p_j}\right\rceil\cdot C_j+$$

$$\sum_{j\in it(\tau_{ua})}\left(\left\lfloor\frac{W_{ua}(t^{(n)})}{T_j}\right\rfloor\cdot L_j+\min\left(\left\lceil\frac{W_{ua}(t^{(n)})\%T_j}{p_j}\right\rceil;L_j\right)\right)\cdot C_j+$$

$$\sum_{j\in hp(\tau_{ua})}\left(\left\lceil\frac{W_{ua}(t^{(n)})}{T_j}\right\rceil\cdot C_{jn}+\left\lfloor\frac{W_{ua}(t^{(n)}}{T_j}\right\rfloor\cdot L_j\cdot C_j\right.$$

$$\left.+\min\left(\left\lceil\frac{(W_{ua}(t^{(n)})\%T_j-C_{jn}-(p_j-C_j))}{p_j}\right\rceil,L_j\right)\cdot C_j\right)$$

This formula contains threee parts: the first part represent the interference of the classical tasks ($Ch(\tau_{ua})$) on the reponse-time of τ_{ua}; the second part is the application of theorem 2 ($it(\tau_{ua})$) and the last part is the application of theorem 3 ($hp(\tau_{ua})$).

By applying this formula with the case study, we obtain the table below:

Tasks	Period	deadline	Priority	Worst-case response time
1	200000	200000	1	56156
2	20000	10000	7	6532
3	50000	30000	5	15532
4	20000	10000	6	6572
5	250000	140000	2	56096
6	60000	60000	4	54636
7	250000	160	11	124
8	20000	720	10	468
9	100000	80	12	12
10	250000	5000	9	3408
11	20000	7500	8	5620
12	100000	70000	3	55416

Table3: Worst-case response time calculated with the formula

6 Conclusion

In this article, we have presented a new task model: the serial transaction. A serial transaction Γ_i is compound with L_i short but urgent acquisition tasks activated each time a serial packet is received, and a less urgent but longer treatment task activated when a whole frame is received.

The number of acquisition tasks can be important (more than 120 in a real case study) and makes the exact calculation of response time intractable. Moreover, overestimating the worst-case response time of the urgent acquisition tasks wouldn't allow validating a task system.

After simplifying the way to evaluate the interference of a transaction and to find the critical instant candidate (Theorem 1), we have shown that for tasks of intermediate priority, the critical instant always coincides with the release of the first task of the transaction (Theorem 2) . This new result allows us to calculate an exact worst-case response time for intermediate priority tasks (usually most tasks of a system). We have also presented the concept of reverse transaction deduced from serial transaction. Using this reverse transaction we have presented a method less pessimistic and simpler to implement than the best known approximation method [TN04b]. Our future work will focus on the determination of the real worst-case response time (complexity of the problem). An extension of Theorem 1 taking jitters into account is investigated.

References

[Aud91] N.C. Audsley, Optimal priority assignment and feasibility of static priority tasks with arbitrary start times, Tech. Report YCS-164, University of York, nov. 1991.

[Der74] M.L. Dertouzos, Control robotics : the procedural control of physical processors, Proc. of IFIP Congress, 1974, pp. 807-813.

[DM89] M.L. Dertouzos, A.K. Mok, Multiprocessor on-line scheduling of hard real-time tasks, IEEE Transactions on Software Engineering 15(12), Déc. 1989, 1497-1506.

[GPS1] TIM-LC, TIM-LF, TIM-LP System Integration Manual, http://www.u-blox.com

[Gro99] E. Grolleau, Ordonnancement temps réel hors-ligne optimal à l'aide de réseaux de pétri en environnement monoprocesseur et multiprocesseur, thèse, ENSMA - Université de Poitiers, nov. 1999.

[IMU1] Crista Inertial Measurement Unit (IMU) Interface / Operation Document, May 2004, http://www.cloudcaptech.com.

[Kai82] C. Kaiser, *Exclusion mutuelle et ordonnancement par priorité*, Technique et Science Informatiques, 1982.

[Lab74] J. Labetoulle, Un algorithme optimal pour la gestion des processus en temps réel, Revue Française d'Automatique, Informatique et Recherche Opérationnelle (Fév.1974), 11-17.

[LL73] C.L. Liu and J.W. Layland, Scheduling algorithms for mutltiprogramming in real-time environnement, Journal of the ACM 20(1) (1973), 46-61.

[LW82] J. Leung and J. Whitehead, On the complexity of fixed-priority scheduling of periodic, real-time tasks, Performance Evaluation (Netherland) 2(4) (1982), p.237-250.

[MPC1] MPC555/MPC556 User's Manual October 2000, http://e-www.motorola.com

[MS03] J. Mäki-Turja and M. Sjödin, Improved Analysis for Real-Time Tasks With Offsets – Advanced Model. Technical Report MRTC no. 101, Mälardalen Real-Time Research Centre (MRTC), May 2003.

[Osek1] OSEK/VDX operating system specification 2.2.2 July 2002, http://www.osekvdx.org.

[Osek2] OSEK/VDX System Generation OIL : Osek Implementation Langauge version 2.5, Juillet 2004, http://www.osek-vdx.org.

[OSM1] OSEKturbo OS/MPC5xx v2.2.1 Technical Reference, Juin 2003, http://www.metrowerks.com.

[OSM2] OSEKturbo OS/MPC5xx User's Manual, Juin 2003, http://www.metrowerks.com.

[OSM3] OSEKturbo performance information, http://www.metrowerks.com

[PG98] J.Palencia Gutierrez and M.Gonzalez Harbour. Schedulability Analysis for Tasks with Static and Dynamic Offsets. In Proc. 19th IEEE Real-Time System Symposium (RTSS), December 1998

[SHA90] L. Sha, R. Rajkumar, J. Lehockzy, Priority inheritance protocols : an approach to real time synchronisation, IEEE transaction computers, vol 39, N°9, pp 1175-1185, 1990

[Tin94] K. Tindell, Addind Time-Offsets to Schedulability Analysis, Technical Report YCS 221, Dept of Computer Science, University of York, England, January 1994

[TN04a] J.Mäki-Turja and M.Nolin. Faster Response Time Analysis of Tasks with Offsets. In Proc. 10th IEEE Real-Time Technology and Applications Symposium (RTAS), May 2004

[TN04b] J.Mäki-Turja and M.Nolin. Tighter Response Time Analysis of Tasks with Offsets. In Proc. 10th International conference on Real-Time Computing and Applications (RTCSA'04), August 2004

[XP92] J. Xu and D.L. Parnas, Pre-run-time scheduling of processes with exclusion relations on nested or overlapping critical sections, Phoenix Conference on Computers and Communications (Phoenix, USA), Apr. 1992,pp. 6471-6479