Schedulability Analysis of Serial Transactions

Karim TRAORE
{karim.traore@ensma.fr}

GROLLEAU Emmanuel
{grolleau@ensma.fr}

COTTET Francis
{cottet@ensma.fr}

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

Abstract[1]

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 using known methods on transactions, so we present a less pessimistic 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.

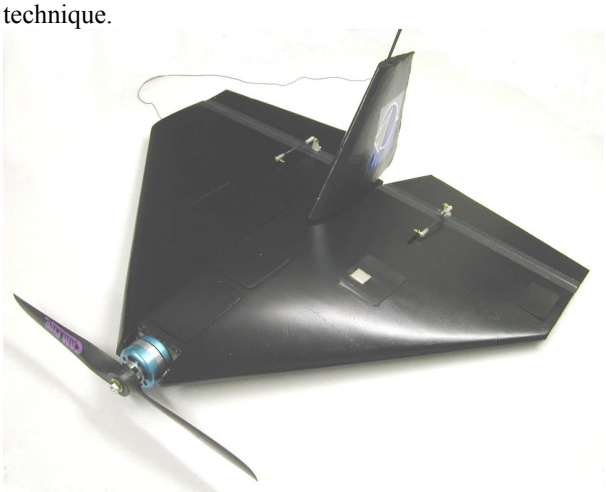


Figure 1: the AMADO

After the definition of the software architecture and the 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 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 bound of the worst-case response time which is less pessimistic than classic RTA (considering that a critical instant consists in a simultaneous release of all the tasks) in a context of tasks with offsets.

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 rely on calculating every combination of the tasks of the transactions; it thus remains exponential in time.

[1] This work was supported by ONERA/DGA

In order to validate the control system of the UAV, we had to deal with tasks with offset which 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 recalls some general results about transactions. Section 4 presents 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 the Application

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 Hardware architecture

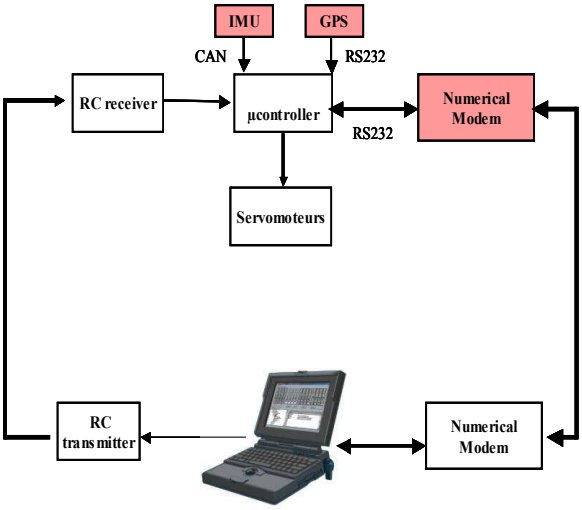


Figure 2: main architecture of the AMADO

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 [MPC1] connected to the actuators (3 servo-commands and the speed-variator, refreshed every 20 ms), an IMU [IMU1], 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.

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, since there is no possible memorisation of the blocks, each block must be read 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 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; 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).

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.

This kind of application can't be validated easily 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.

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

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. Moreover, the first release of the serial transaction is not known precisely because serial transaction is activated by an external peripheral.

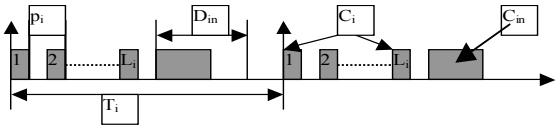


Figure 3 : pattern of serial transaction

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), in order to simplify the analysis, we will consider a reduced task offset Φ_{ij} which is always within 0 and T_i: $\Phi_{ij} = O_{ij}\%T_i$. 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 are ordered by increasing 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. In the table 3, we have represented all the transactions of the UAV application.

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 $P_{ij} > P_{ua}$.

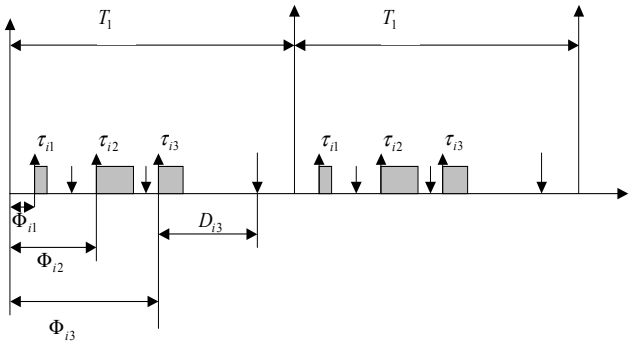


Figure 4: model of tasks with offsets

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 to evaluate the response time obtained by

Transactions	Period	tasks	WCET	Offset	deadline	Priority	"Release for execution" worst-case response time
1	200000	11	200000	0	200000	1	56156
2	20000	21	10000	0	10000	7	**11332**
3	50000	31	30000	0	30000	5	23784
4	20000	41	10000	0	10000	6	**11672**
5	250000	51	140000	0	140000	2	56096
6	60000	61	60000	0	60000	4	54636
7	250000	7i(i=1,…,120)	100	$O_{7i} = (i-1)*160$	160	11	124
		7121	3000	120*160	5000	9	3408
8	20000	81 - 82 - 83	96	0 – 720 - 2*720	720	10	468
		84	900	3*720	7500	8	**10720**
9	100000	9i(i=1..10)	12	$O_{9i} = (i-1)*80$	80	12	12
		911	900	10*80	70000	3	55416

Table 2 : representation of all the tasks of the configuration using the symbolism of transaction and values of worst-case response time with "release for execution" method

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 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.

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 $W_{ic}(\tau_{ua}, t)$ the interference of Γ_i on the response time of τ_{ua} during a time interval of length t.

$$W_{ic}(\tau_{ua}, t) = \sum_{j \in hp_i(\tau_{ua})} \left(\left\lceil \frac{t^*}{T_i} \right\rceil \cdot C_{ij} \right)$$

$$t^* = t - phase(\tau_{ij}, \tau_{ic})$$

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

t^* represents the time during which τ_{ij} can interfere with τ_{ua}.

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

The upper bound of the response time is obtained by iteration : $R_{ua}^0 = C_{ua}$

$$R_{ua}^{(n+1)} = C_{ua} + \sum_{\Gamma_i \in \Gamma} A(\tau_k, \Gamma_i, R_{ua}^n).$$

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

The interference that a transaction imposes on a 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 interference function of a transaction on a task of lower priority. 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).

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

Upper bound method based on the "imposed" interference

This method has been proposed in [TN04b]. It 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] subtracts a parameter x (see Figure 5) from the original interference formula:

$$W_{ic}(\tau_{ua}, t) = \sum_{j \in hp_i(\tau_{ua})} \left(\left(\left\lfloor \frac{t^*}{T_i} \right\rfloor + 1 \right) * C_{ij} - x_{icj}(t) \right)$$

$$t^* = t - phase(\tau_{ij}, \tau_{ic})$$

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

$$x_{icj}(t) = \begin{cases} 0 \\ \max\left(0, C_{ij} - (t^* \bmod T_i)\right) \end{cases} \quad \text{for } t^* \geq 0$$

$x_{icj}(t)$ 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: this transaction has 4 tasks with period $T_i = 50$

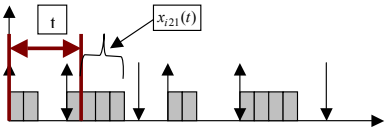


Figure 5: "imposed" interference

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

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

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

$$\tau_{i3} := < 2, 11, 5, 0, 0, 3 >$$

$$\tau_{i4} := < 4, 16, 15, 0, 0, 1 >$$

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

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

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

With the value of each $W_i(\tau_{ua}, t)$, the response time R_{ua} of τ_{ua} can be calculated. $R_{ua}^{(n+1)} = C_{ua} + \sum_{\Gamma_i \in \Gamma} W_i(\tau_{ua}, R_{ua}^n)$. R_{ua} is obtained by fix-point iteration starting with $R_{ua}^0 = C_{ua}$. Let us execute this method on the example (Figure6)

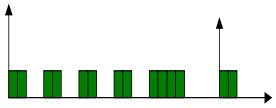


Figure 6. Example for imposed interference

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 response-time. We present at first the details of iteration number 2:

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 give the values obtained in the different iterations :

Iteration	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.

4- Contribution to RTA of transactions

4.1 Transactions without jitters

In this section, we first simplify the way to compute the interference [PG 98] for general transactions with no jitter.

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

$$W_{ic}(\tau_{ua}, t) = \sum_{\forall j \in hp_i(\tau_{ua})} \left(I_{ijc}^{Set1} + I_{ijc}^{Set2}(t) \right) \quad \text{with}$$

$$I_{ijc}^{Set1} = \left\lfloor \frac{J_{ij} + \Phi_{ijc}}{T_i} \right\rfloor C_{ij}, \quad I_{ijc}^{Set2} = \left\lceil \frac{t - \Phi_{ijc}}{T_i} \right\rceil C_{ij}, \text{ and}$$

$$\Phi_{ijc} = (T_i + O_{ij} - (O_{ic} + J_{ic}))\% T_i$$

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. $p<q \Rightarrow \Phi_{ikp} \leq \Phi_{ikq}$). 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 :

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

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 :

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

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 $\forall k_p \in hp_i(\tau_{ua}), k_p \neq k_1, W_{ik_1}(\tau_{ua},t) - W_{ik_p}(\tau_{ua},t)$:

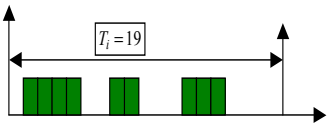
$$W_{ik_1}(\tau_{ua},t) - W_{ik_p}(\tau_{ua},t) = \sum_{\substack{k_j \in hp_i(\tau_{ua}) \\ k_j < k_p}} \left(\left\lceil \frac{t-(\Phi_{ik_j}-\Phi_{ik_1})}{T_i} \right\rceil - \left\lceil \frac{t-(T_i+\Phi_{ik_j}-\Phi_{ik_p})}{T_i} \right\rceil \right) C_{ik_j} + \sum_{\substack{k_j \in hp_i(\tau_{ua})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

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)

Example : transaction of period=19 with 3 tasks (Fig. 7)



$\tau_{i1} =< 4,1,6,0,0,1 >$
$\tau_{i2} =< 2,7,5,0,0,2 >$
$\tau_{i3} =< 3,12,6,0,0,3 >$

Figure 7.calculation with intervals

	$W_{i1}-W_{i2}$			$W_{i1}-W_{i3}$		
	]0;13]	]0;6]	]6;11]	]0;8]	]6;14]	]0;11]
If $t \in I$	C_{i1}	$-C_{i2}$	$-C_{i3}$	C_{i1}	C_{i2}	$-C_{i3}$
If $t \notin I$	0	0	0	0	0	0

Evaluation of $W_i(\tau_{ua},t)$ with t=14

$W_{i1}(\tau_{ua},14) - W_{i2}(\tau_{ua},14) = 0+0+0 = 0$

$W_{i1}(\tau_{ua},14) - W_{i3}(\tau_{ua},14) = 0+C_{i2}+0 = 2$

Thus $W_i(\tau_{ua},14) = W_{i1}(\tau_{ua},14) = 9$

With this method, it is sufficient to evaluate only one value of $W_{ic}(\tau_{ua},t)$

4.2 Serial transaction

Let us introduce the definition of a serial transaction:

Definition1: A serial transaction is a transaction with the following constraints:

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.

Example of serial transaction : (Figure 6)

Definition2 : 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 the treatment task of Γ_i.

Definition3 : 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.

The next result relies on the intervals defined in section 4.1, let us define E_{ikj} as the shift between two successive tasks of higher priority than the task under analysis (Figure 8). Let $k_1,k_2,.....,k_{|hp_i(\tau_{ua})|}$ the elements of $hp_i(\tau_{ua})$. We assume that $hp_i(\tau_{ua})$ is ordered by offsets values increasing i.e $\Phi_{ik_j}\le\Phi_{ik_{(j+1)}}$ for $j<|hp_i(\tau_{ua})|$

$E_{ij}=\Phi_{ik_{(j+1)}}-\Phi_{ik_j}$ for $j<|hp_i(\tau_{ua})|$ and

$E_{i|hp_i(\tau_{ua})|}=T_i+\Phi_{ik_1}-\Phi_{ik_{|hpi(\tau ua)|}}$

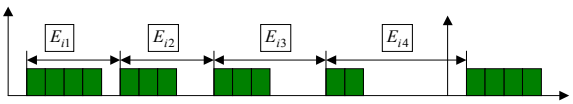


Figure 8. Illustration of E_{ij} and theorem 1

Theorem 1 shows that for specific patterns of transactions without jitters where the WCET of tasks are decreasing and the shifts between successive offsets are increasing, the critical instant of a task always coincides to the first instance of the transaction. 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 1 : 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 are decreasing, i.e. $C_{ij}\ge C_{ik}$ $\forall(j\le k)\in hp_i(\tau_{ua})$, and offset shifting are increasing i.e. $E_{ij}\le E_{i(j+1)}$ for $j<|hp_i(\tau_{ua})|$, 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}, $\sum E_{ij}=T_i$. For this proof, we use the method of calculation presented in section 4.1. In this section we have shown 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 $k_j\ge k_p$ 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} decreasing and E_{ij} increasing ; 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 $E_{ik1}\le E_{ik2}$

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

…

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

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

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 $t\%T_i \in]0..T_i-(E_{ik_1}+E_{ik_2}+\ldots+E_{ik_{p-1}})]$ since $T_i=\Sigma E_{ij}$ $t\%T_i \in]0..E_{ik_p}+E_{ik_{p+1}}+\ldots+E_{ik_n})]$ 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 $t\%T_i \in]E_{ik_1}..T_i-(E_{ik_2}+\ldots+E_{ik_{p-1}})]$ since $T_i=\Sigma E_{ij}$ $t\%T_i \in]E_{ik_1}..E_{ik_1}+E_{ik_p}+E_{ik_{p+1}}+\ldots+E_{ik_n}$ 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 $t\%T_i \in]0..E_{ik_1}+\ldots+E_{ik_{p-1}}]$, 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 $t\%T_i \in]E_{ik_p}..E_{ik_1}+\ldots+E_{ik_p}]$, 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 $t\%T_i \in]E_{ik_p}+E_{ik_{p+1}}+.$ $..+E_{ik_{n-1}}..E_{ik_1}+E_{ik_2}+E_{ik_3}+\ldots+E_{ik_{n-1}}]$, let un note I_{ikpkn} this interval

The proof uses the same way as before, except that for the general case, we show that there are always at least as many positive interval than negative intervals. Since the WCET cannot decrease, and since 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 $W_{i1}(\tau_{ua}, t\%T_i \in]0..E_{ik1}])- W_{ip}(\tau_{ua}, t\%T_i \in]0..E_{ik1}]) \geq 0$
- $t\%T_i \in]E_{ik1}..E_{ik1}+E_{ik2}]$: $t\%T_i \in I_{ikpk2}$ (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 $Eikj$.

□

5- Validation of the case study

Theroem 1 implies that in order to analyse an intermediate priority task, 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-case response time with a classic response time analysis. Note that this theorem cannot be applied to a lower priority task, because the condition "decreasing WCET" is not satisfied in this case.

Let S be a set of transactions.

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

Let us note $hp(\tau_{ua})$ the set of serial transactions in S such as τ_{ua} is a lower priority task. Let us note $it(\tau_{ua})$ the set of indices of serial transactions in S such as τ_{ua} is an intermediate priority task .

By applying Theorem 1, the interference applied by the serial transactions whose indices belong to $it(\tau_{ua})$ in a time interval of length t does not need any specific study related to transactions. It is given (tight upper bound) by:

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

In this formula, $\left\lfloor \frac{t}{T_j} \right\rfloor$ represents the number of periods T_j completed in the time interval of length t ; and $\min\left(\left\lceil \frac{t\%T_j}{p_j} \right\rceil, L_j \right)$ represents the number of acquisition tasks activated in the remaining time($t\%T_j$).

We still need to use the technique defined in [TN04b] in order to study the interference of the serial transactions whose indices belong to $hp(\tau_{ua})$, leading to a pessimistic upper bound, but allowing us to validate the case study (see Table 3). This application is valid

because in the table 3, we can see that for all the tasks, the worst-case response time is lower than the deadline.

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

Table 3: Worst-case response time calculated with serial transaction method

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 the validation of a task system.

After simplifying the way to evaluate the interference of a transaction and finding the critical instant candidate, 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 1) . This new result allows us to calculate an exact worst-case response time for intermediate priority tasks (usually most tasks of a system), while we still use the method proposed in [TN04b] for the tasks of lower priority than a whole serial transaction. Our future work is generalizing the theorem 1 to a larger case of transactions called monotonic transactions. Moreover, an extension of this theorem 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.

[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.osek-vdx.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

[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 exclusionrelations on nested or overlapping critical sections, Phoenix Conference on Computers and Communications (Phoenix, USA), Apr. 1992,pp. 6471-6479.