Traffic Shaping in Real-Time Distributed Systems : a Low-Complexity Approach

Bruno Gaujal and Nicolas Navet
LORIA-INPL
ENSEM - 2, Avenue de la forêt de Haye
F-54516 Vandoeuvre-lès-Nancy
Tel: 33.3.83.59.57.46 Fax: 33.3.83.44.07.63
{gaujal,nnavet}@loria.fr

February 9, 2001

Abstract

In real-time systems, one generally identifies two types of timing requirements, hard and soft constraints. In this study, it is assumed that the Hard Real-Time traffic (HRT) is periodic with deadlines that must be guaranteed, while the Soft Real-Time traffic (SRT) is aperiodic with timing constraints that could occasionally be missed without major consequences. In this paper, the problem of scheduling these 2 types of traffic with different performance objectives will be addressed : (1) ensure that the timing requirements of HRT traffic are met, (2) minimize as much as possible the response time of SRT traffic while satisfying (1). For this purpose, we propose an easily implementable and low-complexity traffic shaping policy which preserves *feasibility* and improves response times for SRT traffic. The underlying idea is that it is possible to diminish the response time of SRT traffic if the busy periods induced by the HRT traffic are "harmoniously" distributed over time, creating time intervals during which the resource (i.e. the processor or the medium) can be used by SRT traffic with minimum delay. A computer-implementable algorithm that has to be executed independently on each node of the bus is also provided, as well as several extensions of the original model.

keywords : Real-Time Systems, Traffic Shaping, Scheduling Algorithm, Local Area Network.

1 Introduction

Limitations of existing solutions.
The problem of jointly scheduling both HRT traffic and SRT traffic is an important issue in

real-time systems and many effective approaches have appeared in the literature aimed at improving the responsiveness of SRT tasks [1, 16, 10, 17] but as pointed out in [2] their complexity induce high overheads when used on-line. In addition, most presuppose restrictive hypotheses such as the knowledge of the first release time of all periodic sources, making them generally hard to use for message scheduling where nodes are not synchronized. The Dual-Priority policy (DP), originally introduced for the preemptive scheduling of tasks in [2, 3], can also be applied to message scheduling as proposed in [22] even if nodes are not synchronized and it has proved to be very efficient [11] in the context of a CAN (Controller Area Network [7]) based in-vehicle multiplexing system. In essence, this policy facilitates the responsive execution of SRT frames by transmitting all HRT frames immediately, when there are no SRT traffic ready, or as late as possible where SRT frames are ready to be transmitted. Under dual-priority scheduling, the priority range must be partitioned into three bands: "low hard", "soft", "high hard" in increasing level of priority. An HRT frame is first queued with a priority within the "low hard" band and later, when it becomes urgent, it will be promoted to the "high hard" range. Let t be the release time of a HRT frame m, D_m be its deadline and $\mathcal{R}_m$ be its worst-case response time. The latest priority promotion time (from the "low-hard" to the "high-hard" range) such that the frame will meet its deadline, is $\mathcal{R}_m$ before its deadline, or otherwise expressed $(t + D_m - \mathcal{R}_m)$ [22]. In practice the drawback of the DP policy is that it requires modifications at the Data Link Layer (DLL) level which is generally implemented in hardware. For instance, in the context of CAN, Tindell and Hansson in [22] proposed that the frame priority be boosted at the communication controller level using a priority "step-up" timer for each transmission buffer but, to the best of our knowledge, such a controller is not available yet.

Goal of this paper.

In this paper, the problem of giving an effective and widely applicable solution (in the sense that it does not require any hardware modification unlike DP) for the scheduling of 2 types of traffic with different performance objectives will be addressed :

1. minimize as much as possible the average response time of SRT traffic.

2. ensure that the timing requirements of HRT traffic are met.

We show a first theorem that provides the sequence of emission times for HRT traffic that minimizes the average response time of SRT messages. However, this does not give any guarantees on the respect of HRT constraints. From this sequence, we derive a new traffic shaping policy and we prove that a set of HRT messages that are guaranteed to respect their deadlines with the "send As Soon As Possible" (ASAP) strategy will also meet their deadlines when scheduled under the proposed policy.

Organization of the paper.

The structure of the paper is as follows. In Section 2 we introduce the framework and the formulation of the problem. We also the optimal shape of the HRT traffic when its constraints are not taken into account. In Section 3, we propose a strategy to postpone the arrivals of the periodic messages in order to reduce the response time for the aperiodic traffic. An algorithm for the on-line computation of this traffic shaping is also provided. In

Section 4, we relax some assumptions used previously to show that this technique is rather general and can be applied in a large variety of cases given minor modifications. Section 5 gives some experiments that show the improvement of traffic shaping over the classical ASAP strategy.

2 Description of the problem

2.1 Framework

The framework of this study is the same as in [11] : a HRT periodic message with identifier m is characterized by (C_m, T_m, D_m) where C_m is the transmission time, T_m is the period, D_m the deadline. We denote by $\mathcal{M}$ the set of all the HRT messages of cardinality M. Due to the complexity of the general problem, following assumptions are placed :

- The communication protocol at the Data Link Layer level is a CSMA priority bus. In such buses (e.g. VAN [4], CAN [7]), all messages of the system have an unique priority that will serve to decide which message gains the bus : when two or more messages are competing for the bus, the one with the highest priority gains access, delaying the lower priority message(s). This priority based arbitration enable us to compute for each HRT frame of the application the worst-case response time, denoted by $\mathcal{R}_m$ and defined as the maximum delay between the transmission request and the complete reception of the message. The calculation of the worst-case response time of such buses has been extensively studied in the literature [21, 20, 19]. The principles of this calculation are shortly recalled in Appendix.

- The worst-case response time $\mathcal{R}_m$ must be bounded for each frame by D_m otherwise one can not guarantee that the message has been sent before the next one is queued and the set of message is non-schedulable. We denote R_m the quantity $D_m - \mathcal{R}_m$ which is the latest time, relatively to the arrival of the frame, at which the frame has to be sent in order to guarantee its deadline.

- All nodes on the network have the knowledge of the periods of all HRT frames.

- Time is slotted in the sense that HRT frames can only be sent at multiples of the chosen time granularity. For instance, for in-vehicle multiplexing systems such as described in [14, 13], a granularity of 1 millisecond is adequate while for most multimedia systems 10ms will be sufficient. From now on, we will assume that all used variables (i.e. T_m, R_m, $\mathcal{R}_m$, D_m) are multiples of the time granularity.

In a first step, we will also make the following assumptions which will be all removed in Section 4 :

- The deadline of a frame m is equal to its period ($D_m = T_m$).

- There is no jitter in emission, this means that all HRT are exactly released at multiples of their periods.

- The nodes are synchronized : they are all assumed to begin transmitting from the start of the system.

- The characteristics of the set of HRT messages is fixed for the whole lifetime of the application (no "mode changes" [9] inducing traffic characteristic modifications) and all nodes on the bus have full knowledge of these characteristics.

The ASAP policy consists in queuing a message m for emission as soon as possible. That is, the nth instance of message m is queued at instant nT_m. This policy has several advantages. It is very simple to implement and it is always feasible as long as R_m is positive. However, one may wonder if the slackness between instants nT_m and $nT_m + R_m$ may be used in order to optimize characteristics over the other traffic. This is what is done in the rest of the paper where emission of HRT messages may be delayed in order to improve response times of SRT messages.

2.2 Formulation of the problem

As mentioned before, we deal with two types of traffic. HRT traffic is supposed to be periodic and has hard real time constraints; SRT traffic on the other hand, is aperiodic and does not have real-time constraints. In the following, the arrival process of the SRT traffic will be a stochastic process, with a stationary distribution over time with finite intensity.

In this paper, unlike with DP [2, 3], we will restrict ourselves to static priorities which is very convenient from a practical point of view. Each message is thus given a priority level before run-time. All HRT message must be given a higher priority than SRT traffic, otherwise, with our assumption concerning the arrivals of SRT messages, no absolute guarantees on the real-time constraints can be given for HRT messages.

The objective is to shape the HRT traffic in order to make sure that they meet their constraints while reducing the response time of the aperiodic traffic.

- The first objective (respecting the real-time constraints) is met as long as the n-th periodic message of type m asks for the bus before time $nT_m + R_m$. A scheduling policy that meets this condition, provided that the set of messages is *schedulable*, for all instances of all periodic messages is termed *feasible*.

- The second objective is to reduce some functional of the response time of the aperiodic traffic. Let us consider the average value of the response time of all the SRT traffic over the infinite horizon.

2.3 Minimizing the response time of the aperiodic traffic

In this section, we focus on the second objective (shaping the periodic traffic in order to minimize the response time of the aperiodic traffic), and we forget about the constraint given by the first objective (assuring the schedulability of the periodic traffic).

Let $T = \text{lcm}_m T_m$ be the global period of the periodic traffic. Periodic message m occurs q_m times over the interval $[0, T]$ and its transmission time is supposed to be one.

The whole system can be seen as a slotted G/G/1 queue, where the input traffic is the aperiodic traffic and the periodic traffic corresponds to vacations of the server. **meme priorite et FIFO** More precisely,

- Aperiodic messages arrive to a slotted G/G/1 queue according to some given stationary process.
- The server may be either active in providing service, or may be absent for vacation periods. At each time slot, the server can decide whether to take a vacation or not (this corresponds to the arrival of some periodic message at this time slot in the original problem).
- At the nth slot, the controller chooses a control a_n that determines the number of periodic messages queued at slot n. Let a be the control sequence $(a_1, a_2, ...)$.
- Performance measures and objectives:
  – Let $h : \mathbb{R} \to \mathbb{R}$ be a convex increasing function.
  – We consider the class Π of all policies that satisfy the constraint:

$$\sum_{n=1}^{T} a_n = \sum_{m=1}^{M} q_m. \tag{1}$$

Consider the new problem: let W_n be the response time of an aperiodic message that would arrive at slot n.
Define the average expectation of the function h of the waiting time:

$$g(a) = \overline{\lim_{s\to\infty}} \frac{1}{s} \sum_{n=1}^{s} Eh(W_n(a_1, ..., a_n)), \tag{2}$$

The objective is to minimize $g(a)$ over $a \in \Pi$.

This kind of systems was studied in [5], where a more general version of Theorem 1 was proved.

Now, we construct a sequence a in the following way. Let $u = \sum_{m=1}^{M} q_m/T$, be the total average density of the periodic messages to be allocated over one period. We define $a_n^* = \lceil nu \rceil - \lceil (n-1)u \rceil$. Note in particular that $a^* \in \Pi$.

An example of the construction of a^* is given in Figure 1

Theorem 1. *The function $g(a)$ is minimized over Π at point a^*.*

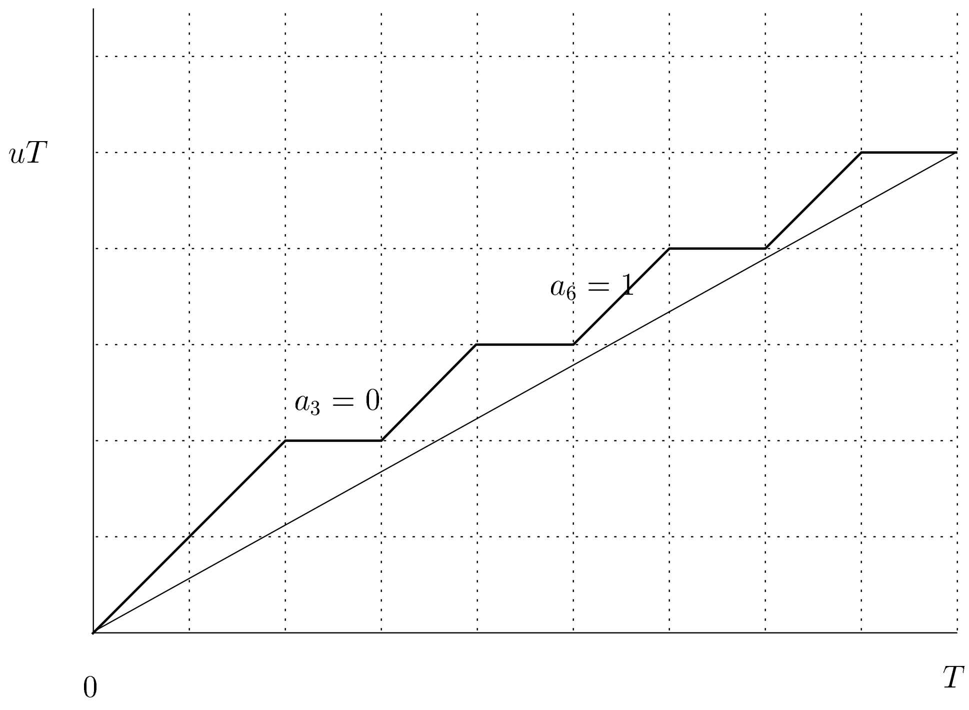

Figure 1: Construction of a^* when $T = 9$ and $u = 5/9$.

Proof: This is a sketch of the proof, which uses Theorem 6.1 in [5]. Indeed, the queue described here is a slotted version of the general setting of Theorem 6.1 in [5]. Here, the potential inter-potential vacation times are constant and hence i.i.d. As for the aperiodic traffic, we assumed that it forms a stationary slotted sequence. Therefore, it is a point process over the slots. As for the duration of the vacations, they are constant (equal to one) in our case, and the service times of the aperiodic traffic are assumed to be stationary, and independent of the vacations. We also assume that time 0 is the instant of the first vacation, which satisfies the sixth point in Theorem 6.1 in [5]. ■

The general rule that this analysis provides is that, in order to reduce the expected response time of the aperiodic traffic, one had to distribute the load due to periodic arrivals over time as regularly as possible. A bursty periodic load will cause large expected response times for aperiodic messages while a smooth periodic load will reduce this response time. However, the optimal allocation of the periodic traffic given in Theorem 1 has two drawbacks: (1) the assumption on the HRT transmission time is inconvenient in practice and most of all, (2) the policy may not be feasible.

The challenge is now to try to distribute the periodic messages evenly over time while preserving the feasibility property. For that, the definition of the density u of the periodic traffic is refined to take into account the feasibility constraint.

3 Traffic shaping with an on-line procedure

In this section, we provide a fast method that behaves better than the assignment As Soon As Possible (ASAP) with respect to the aperiodic traffic. Again, the focus is on the shaping of the periodic traffic and the aperiodic traffic is considered to be uncontrolled. If periodic messages have similar lengths, then an even distribution of the periodic load is similar to an even distribution of the arrival instants of the period messages which is achieved in the following construction. The method consists in two steps :

- The first step is global, which means that it considers the total traffic without distinguishing the different messages. During this step, we select instants (slots) of emission of messages.
- The second step of the algorithm consists in assigning a message to each slot, selected for emission, while satisfying feasibility.

3.1 Density and emission sequences

We describe the first step of the procedure in full details, which can be seen as a slot selection process.

As assumed earlier, time for the start of transmission of HRT frames is slotted. For each slot i and each message m, we define the pair (p_i^m, q_i^m), defined uniquely by the Euclidean division of i by T_m, $i = p_i^m T_m + q_i^m, q_i^m < T_m$.

For each message m, a density sequence is defined $\{u_i^m\}_{i \in \mathbb{N}}$.

$$u_i^m = \begin{cases} \frac{1}{T_m - \mathcal{R}_m + 1} & \text{if } q_i^m \leq R_m, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

Intuitively, the density sequence represents the "quantity" of message to be allocated per slot.

The following assertion is a simple consequence of the definition of u_i^m. For all integer p,

$$\sum_{i=pT_m}^{pT_m+R_m} u_i^m = \sum_{i=pT_m}^{(p+1)T_m-1} u_i^m = 1. \tag{4}$$

The global density at slot i is defined by

$$u_i = \sum_{m=1}^{M} u_i^m. \tag{5}$$

The sum of all densities up to slot i is

$$U_i = \sum_{j=0}^{i} u_j. \tag{6}$$

Finally, the purpose of the selection is to transform the density sequence u_i with values in $\mathbb{R}$ into a selection sequence v_i with values in $\mathbb{N}$ and such that v_i majorizes u_i ($\sum_{i=1}^{T} v_i \geq \sum_{i=1}^{T} u_i$) while being as close as possible to u_i.

$$v_0 = \lceil u_0 \rceil, \quad v_i = \lceil U_i \rceil - \lceil U_{i-1} \rceil. \tag{7}$$

The construction of the v sequence is illustrated in Figure 2. This construction is rather classical in admission control theory, see for example [6].

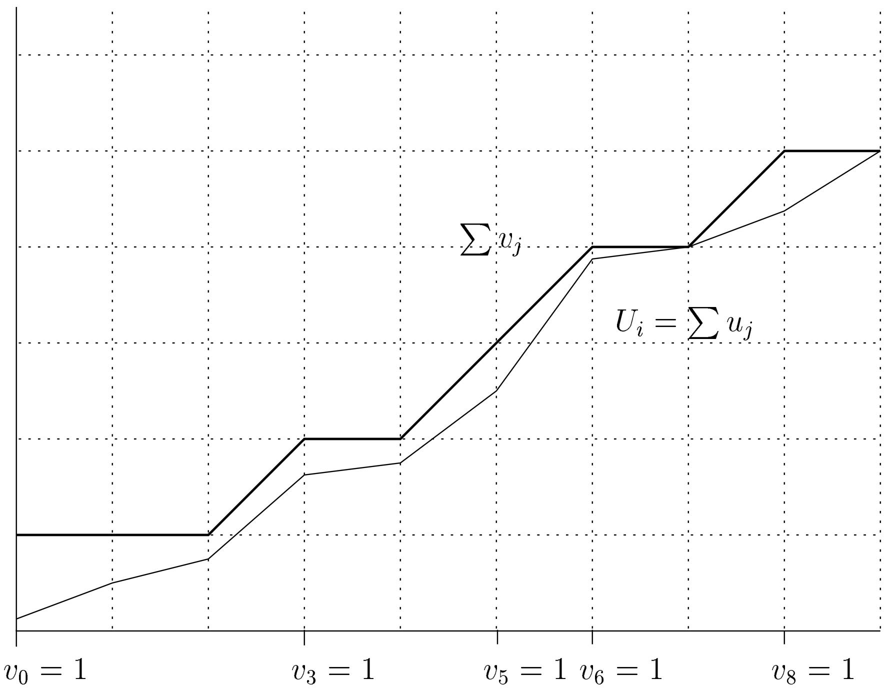

Figure 2: Selection procedure, slots 0,3,5,6 and 8 are selected for a message emission.

Result 1. *The complexity of step 1 is linear in the number of slots considered and in the number of messages.*

Proof: The construction of the quantities u_i^m, u_i, U_i and v_i are all linear in i and in m. ■

Remark 1. *It is sufficient to do step 1 up to $T = lcm(T_m)$, the selection being periodic with period T.*

3.2 Deadline first allocation

Once the selection is done, one has to choose which message to assign to each selected slot. This section is devoted to the choice of a message at each emission slot ($v_i = 1$). This is an allocation process.

This allocation can be done according to refined criteria that take into account the size of messages. However to stick with the low-complexity approach, we will use the deadline first allocation (DF) and prove its feasibility.

In general an allocation rule consists in constructing a sequence $\{a_i\}$, with values in $\mathcal{M}$, such that a_i is defined only when $v_i = 1$.

An allocation is feasible is for each message m and for each integer p, $\#\{a_j = m, pT_m \leq j \leq pT_m + R_m\} = 1$.

Recall that $i = p_i^m T_m + q_i^m, q_i^m < T_m$ (Euclidean division of i by T_m). Message m is *pending* at slot i if $\#\{a_j = m, j < i\} < p_i^m$. The set of all pending messages at slot i is denoted by P_i.

Definition 1. *Allocation rule DF: at slot i (with $v_i = 1$), allocate the message m satisfying*

$$m = argmin_{m \in P_i}\{p_i^m T_m + R_m\}, \tag{8}$$

where ties are broken arbitrarily.

Theorem 2. *If for all m, $R_m \geq 0$, then the allocation rule DF is feasible.*

First note that when $R_m < 0$ for some m, there is no hope to find an allocation which is feasible under all possible phases between the messages. This theorem says that whenever the set of messages is schedulable, then the allocation rule DF is feasible.

Proof: The proof holds by a double induction on the number of messages and on the slot.

With a single message ($\mathcal{M} = \{1\}$), the selection step sets $v_i = 1$ iff $i = pT_1$, for all $p \in \mathbb{N}$. The pending set $P_i = \{1\}$ if $i = pT_1$ and is empty otherwise. The allocation rule DF is clearly feasible, since $a_i = 1$ for each $i = pT_1$ satisfies the feasibility condition.

Now, we assume that we have $M > 1$ messages. For the first $M - 1$ messages, we assume that DF was feasible and we want to prove that is remains feasible with one more message (message M). All quantities with respect to the system with $M - 1$ messages (called system 1 in the following) will be preceded by prefix 1, while all quantities with respect to the full system with M messages (called system 2) will have prefix 2.

The first assertion is: for all p,

$$\begin{aligned} {}^2U_{pT_M} &= {}^1U_{pT_M} + p, & (9) \\ {}^2U_{pT_M+R_M} &= {}^1U_{pT_M+R_M} + p + 1, & (10) \end{aligned}$$

Indeed, Equality (9) clearly holds when $p = 0$. Equation (4) shows that Equalities (9) and (10) hold for all p by a straightforward induction.

This also means that with M messages, there is exactly one more selection point in each interval of the type $[pT_M, pT_M + R_M]$ (Equation (10)), and the same number of selection slots in intervals of the type $]pT_M + R_M, (p+1)T_M[$ (Equation (9)).

The number of selection points but also their location in both systems play an important part in this proof. This is the object of the second assertion.

In System 1, let us call ${}^1i_1, \cdots, {}^1i_k$ the slots selected in the interval $[pT_M, pT_M + R_M]$, and ${}^1i_{k+1}, \cdots, {}^1i_{k+n}$ are the slots selected in the interval $]pT_M + R_M, (p+1)T_M[$.

Similarly, in System 2, ${}^2i_1, \cdots, {}^2i_k, {}^2i_{k+1}$ are the slots selected in the interval $[pT_M, pT_M + R_M]$, and ${}^2i_{k+2}, \cdots, {}^2i_{k+n+1}$ are the slots selected in the interval $]pT_M + R_M, (p+1)T_M[$.

The second assertion is

$$\begin{aligned} {}^2i_j &\leq {}^1i_j, \quad \forall 1 \leq j \leq k, & (11) \\ {}^2i_{j+1} &= {}^1i_j, \quad \forall k+1 \leq j \leq k+n. & (12) \end{aligned}$$

To prove this assertion, let us consider any slot $i \in [pT_M, pT_M + R_M]$. Using equation (9), $\lceil {}^2U_i \rceil = \lceil {}^1U_i + \sum_{j=pT_M}^{i} u_i^M \rceil + p$ and $\lceil {}^2U_{pT_M} \rceil = \lceil {}^1U_i \rceil + p$.

Therefore,

$$\sum_{j=pT_M}^{i} {}^2v_j = \lceil {}^2U_i \rceil - \lceil {}^2U_{pT_M} \rceil \geq \lceil {}^1U_i \rceil - \lceil {}^1U_{pT_M} \rceil = \sum_{j=pT_M}^{i} {}^1v_j.$$

Since this is true for all i, it shows that ${}^2i_j \leq {}^1i_1, \quad \forall 1 \leq j \leq k$.

For Equation (12) and for any slot $i \in]pT_M + R_M, (p+1)T_M[$, $\lceil {}^2U_i \rceil = \lceil {}^1U_i \rceil + p + 1$.

Therefore,

$$\sum_{j=pT_M+R_M+1}^{i} {}^2v_j = \lceil {}^2U_i \rceil - \lceil {}^2U_{pT_M+R_M+1} \rceil = \lceil {}^1U_i \rceil - \lceil {}^1U_{pT_M+R_M+1} \rceil = \sum_{j=pT_M}^{i} {}^1v_j.$$

Since this is true for all i, and considering the fact that system 2 has one more allocation in the interval $[pT_M, pT_M + R_M]$, it shows that ${}^2i_{j+1} = {}^1i_j, \quad \forall k+1 \leq j \leq k+n$.

Now comes the core of the proof which is also done by induction on p. This is the only part of the proof concerning the allocation process; all the previous points were concerning the selection process only.

We will prove the third assertion, which is two-fold. For all $p \in \mathbb{N}$,

$$\begin{aligned} {}^2P_{pT_M} &= {}^1P_{pT_M} \cup \{M\}, & (13) \\ \#\{a_j = M, pT_M \leq j \leq pT_M + R_M\} &= 1. & (14) \end{aligned}$$

First let us consider the first interval, $[0, R_M]$. Obviously, ${}^2P_0 = \mathcal{M}$ and ${}^1P_0 = \mathcal{M}/\{M\}$. One has ${}^2P_0 = {}^1P_0 \cup \{M\}$.

The system with $M-1$ messages has, say, k selection slots in the interval $[0, R_M]$ and n selection slots in the interval $]R_M, T_M[$, called ${}^1i_1, \cdots, {}^1i_k$ and ${}^1i_{k+1}, \cdots, {}^1i_{k+n}$ respectively. Equations (9) and (10) imply that the system with M messages has $k+1$ selection slots in the interval $[0, R_M]$ and n selection slots in the interval $]R_M, T_M[$, called ${}^2i_1, \cdots, {}^2i_k, {}^2i_{k+1}$ and ${}^2i_{k+2}, \cdots, {}^2i_{k+n+1}$ respectively.

In system 1, the DF allocation gives the selection slots to messages $m_1, \cdots, m_{n+k}$ with respective deadlines $R_{m_1} \leq \cdots \leq R_{m_{n+k}}$. The feasibility of system 1 says ${}^1i_j \leq R_{m_j}$.

Now, let us consider system 2. There is one more pending message, message M, with deadline R_M. Let us assume that $R_{m_h} \leq R_M \leq R_{m_{h+1}}$, for some $h < k$.

If R_M is smaller than R_{m_1} or if R_M is larger than R_{m_k}, then h is not defined, however these can be considered as particular cases of the general case by neglecting one side of the following argument.

- For $1 \leq j \leq h$, the j-th selection is assigned to message m_j: ${}^2a_{i_j} = m_j$. The feasibility for message m_j is still satisfied since by Equation (11), ${}^2i_j \leq {}^1i_j \leq D_{m_j}$.

- For selection slot $h+1$, the DF allocation rule assigns message M. The feasibility for message M is satisfied since ${}^2i_{h+1} \leq {}^2i_{k+1} \leq R_M$ by definition of h.

- For $h+2 \leq j \leq k+1$, the DF allocation rule assigns message m_{j-1}. The feasibility for message m_{j-1} is still satisfied since ${}^2i_j \leq {}^2i_{k+1} \leq R_M \leq R_{m_{j-1}}$.

- Finally, for $k+2 \leq j \leq k+n$, the DF allocation rule assigns message m_{j-1}. The feasibility for message m_{j-1} is still satisfied since by Equation (12),${}^2i_j = {}^1i_{j-1} \leq R_{m_{j-1}}$.

All these cases make system 2 allocates the same messages as system 1, plus message M, all in time, during the interval $[0, T_M[$. The initial step of the induction is finished.

Let us assume that Equations (13) and (14) are true up to slot pT_M, therefore ${}^2P_{pT_M} = {}^1P_{pT_M} \cup \{M\}$, and the feasibility properties hold up to slot pT_M. By mimicking the reasoning used for the first interval, the messages allocated in the interval $[pT_M, (p+1)T_M[$ in the full system (system 2) are the same as in system 1, plus message M, all of them being allocated before their deadline. Therefore, message M meets its deadline in the interval $[pT_M, (p+1)T_M[$ (Equation (14) and the pending sets at slot $(p+1)T_M$ will satisfy Equation (13)).

This ends the proof. ■

Result 2. *The complexity of step 2 is linear in the number of slots considered and in the number of messages.*

Proof: This is a straightforward consequence of the fact that the set of pending messages is always bounded by M. The modifications of this set when passing from one slot to the next are all linear in M and independent of the slots. ■

Remark 2. *It is sufficient to do step 2 up to $T = lcm(T_m)$, the allocation being periodic with period T.*

To end this section, note that the main feature of this procedure is that the selection process is global. This is why the global periodic load is smoother than without shaping. Look for instance at time 0 (often termed the *critical instant*) where the ASAP strategy would lead to many arrivals amounting to a big busy period while the selection procedure may distribute those arrival on a larger time interval.

The actual effect of the traffic shaping proposed here will be shown in a series of experiments presented in Section 5.

3.3 Algorithm

We now present the algorithm that implements the analysis developed in Subsection 3.1 and 3.2. The procedure *Shaping* (see Figure 3) has to be executed on each station independently and decides for each slot whether it is selected for emission. In that case, the procedure chooses which HRT message has to be transmitted. This computation can be done periodically before each time slot or by using the idle time of the station for performing large amounts of pre-computation. In the latter case, the queuing of the message (line 32 of the procedure) has to be postponed until the pre-computed instant of emission.
Parameter *Pending* is an array of booleans used to store the set of pending messages : $Pending[m] = true$ means that message m is pending and thus can be scheduled for transmission. U_i is the the sum of all densities up to the current instant and t is the current time.
Basically the algorithm can be subdivided in 3 consecutive steps :

1. Add the densities of emission brought by the next time slot (lines 5-9 on Figure 3).
2. Decide whether a message has to be sent or not (lines 10-22 on Figure 3).
3. If necessary, chose the message to transmit (lines 23-31 on Figure 3) and, if the message must be sent by the node, queue it (line 32 on Figure 3).

Before the first call of the procedure *Shaping*, t and U_i must be initialized to 0 and all values of $Pending[]$ must be equal to $false$. The *carry* variable (line 14, line 18) is needed because the sum of densities over a single time slot might be greater than one while it is not possible to allocate more than one message within one slot. Thus, we have to store the remaining messages to send them in the following consecutive slots.
As proved earlier, one can see from Figure 3 that the complexity of the procedure *Shaping* is linear in the number of HRT messages of the system.

4 Generalizations

From now on, we will relax some assumptions that were made in section 2. It is pointed out that because of its simplicity, the proposed shaping policy is easily extendible to various cases that are of practical interests.

```
proc $Shaping$(real $U_i$, integer $t$, boolean $Pending[M]$)
  real $u_i$, $U_{i-1}$, $min$;
  integer $m, a_i$;
  $u_i := 0$;
  for $m := 1$ to $M$ do
      if $(q_t^m \leq R_m)$ then $u_i := u_i + \frac{1}{T_m - \mathcal{R}_m + 1}$
                                 if $(q_t^m = 0)$ then $Pending[m] := true$;  fi
      fi
  od
  $U_{i-1} := U_i$;
  $U_i := U_i + u_i$;
  if $(\lceil U_i \rceil - \lceil U_{i-1} \rceil \geq 1)$
    then $v_i := 1$;
         $carry := carry + \lceil U_i \rceil - \lceil U_{i-1} \rceil - 1$;
    else
         if $(carry > 0)$
           then $v_i := 1$;
                $carry := carry - 1$;
           else
                $v_i := 0$;
         fi
  fi
  $min := +\infty$;
  if $(v_i = 1)$ then
              for $m := 1$ to $M$ do
                  if $(Pending[m] \wedge (min > p_i^m T_m + R_m))$
                    then $a_i := m$;
                         $min := p_i^m T_m + R_m$;
                  fi
              od
              $Pending[a_i] := false$;
              if $(sentByThisNode(a_i))$ then $queueMessage(a_i)$;  fi
  fi
```

Figure 3: Procedure *Shaping* that implements the traffic shaping policy

4.1 When deadlines are smaller than the periods

Up to now it has been assumed that the deadline of a frame m (denoted by D_m) is equal to its period which is very usual in real-time systems. This assumption can be relaxed by allowing the deadline to be smaller than the period. Deadlines bigger than periods are not considered because in this case, it can not be guaranteed that the message has been sent before the next one is queued. With $D_m \leq T_m$, Equation (3) becomes :

$$u_i^m = \begin{cases} \frac{1}{D_m - \mathcal{R}_m + 1} & \text{if } q_i^m \leq R_m, \\ 0 & \text{otherwise.} \end{cases} \tag{15}$$

All the rest of the results of Section 3 are left unchanged. The shaping procedure works in the same way, insuring feasibility under these restricted deadlines.

4.2 Desynchronized stations with known offsets

The offset ϕ_m is the difference between the starting instant of the first emission of a message m and a reference point called 0. So far, we have assumed that all offsets where equal to 0. Here we assume that all stations have the knowledge of the offsets at which all messages will be transmitted for the first time. The calculation of u_i has to be changed and Equation (3) becomes :

$$u_i^m = \begin{cases} \frac{1}{D_m - \mathcal{R}_m + 1} & \text{if } i \geq \phi_m \text{ and } q_{i-\phi_m}^m \leq R_m, \\ 0 & \text{otherwise.} \end{cases} \tag{16}$$

A more restricted set of pending messages denoted by $P_i^{'}$ must be defined as

$$P_i^{'} = \{m \text{ s.t. } \#\{a_j = m, j < i\} < p_{i-\phi_m}^m\} \backslash \{m \mid i < \phi_m\}.$$

The rest of the construction of the selection as well as the assignment is left unchanged.

4.3 Desynchronized stations with unknown offsets

If it is assumed that the offsets are not known globally, we have to add an initialization phase to synchronize all stations. This can be done by using any convenient synchronizing protocol that sets a global starting signal as soon as all stations become ready as it exists in the FIP (Factory Instrumentation Protocol, see [18]). This signal will serve as the origin for the calculation of all periods. Whatever the protocol an absolute synchronization can not be guaranteed because of the propagation delay and the finite granularity of the clocks but the procedure can be seen as a reduction of all the offsets to quantities smaller than the time granularity of our model (which is equal to 1).

Under this reasonable condition, all stations are synchronized up to one unit of time. Note however that this small inconsistency may have heavy consequences, since the selected slots computed by the stations may be inconsistent with each other and differ by one unit between two stations. In such a case, a stronger notion of schedulability of the system is needed to insure that the whole shaping procedure is feasible.

Definition 2 (k margins). *A system S has a margin of k units of time if $R_m \geq k$ for any $m \in \mathcal{M}$.*

Using this notion, the shaping procedure is still feasible as long as the system has margin 1.

Indeed, after the synchronization step, all stations proceed with their algorithm as if the system were perfectly synchronized. However, this is an imprecise model and some stations may put selection slots one slot too late, w.r.t other stations (there is no propagation of the error over time). If the system has margin 1, then the assignment to the k-th slot is exactly the same for all stations, since the pending set will not be modified by shifting a selection by one unit of time to the right.

4.4 Jitter in emission

As previously mentioned, the queuing time of a message can occur with a jitter, it means that instead of being queued exactly at $p \cdot T_m + \phi_m$ (with $\phi_m = 0$ in the synchronized case), the message m is ready to be send at times within the interval $p \cdot T_m + \phi_m \overset{+}{-} J_m$ (where the bound J_m is a multiple of the time unit). For instance, this jitter can be caused by a variable amount of computation needed by the task before it could queue the message. The proposed allocation algorithm handles the case where a message is ready before $p \cdot T_m + \phi_m$ by taking safe and pessimistic margins in the computations.

Basically, we assume that the message p arrives at time $p \cdot T_m + \phi_m + J_m$ while all the other messages combined arrive as soon as possible (this is taken into account in R_m), which leaves a shorter interval for the allocation of message m.

Let $j = i - \phi_m$, to take into account the desynchronization part. The calculation of u_i^m has to be modified as follows:

$$u_i^m = \begin{cases} \frac{1}{D_m - \mathcal{R}_m + 1} & \text{if } j \geq 0 \text{ and } J_m \leq q_j^m \leq R_m, \\ 0 & \text{otherwise.} \end{cases} \tag{17}$$

Where R_m also takes the jitter of all messages into account (see Appendix). A more restricted set of pending messages denoted by $P_i^{'}$ must be defined as

$$P_i^{'} = \{m \text{ s.t. } \#\{a_k = m, k < i\} < p_j^m \text{ and } p_j^m > J_m\} \backslash \{m \mid i < \phi_m\}.$$

The rest of the construction of the selection as well as the assignment is left unchanged.

In that case if the system has margin $J = \max_m J_m$, then the proposed policy is still feasible.

4.5 Variable characteristics of HRT messages : mode changes

Large real-time applications generally have a fixed number of predetermined operational phases. An example given in [9] is a computer system on-board an aircraft that will performs different control activities during the flight or when taxiing on the ground. A mode reflects a certain operational phase of the real-time application and the set of transmitted messages depends on the active tasks and thus on the current mode. One of the requirements defined in [9] for safe mode changes is the *consistency* : it means that all nodes change the mode at the same point in time. For instance, this can be achieved through a special message that indicates the mode-change request and that is sent with the highest priority and which is not postponed. Then all nodes, that must have the knowledge of the characteristics of the messages transmitted in all modes, simply have to switch to the new message set and re-initialize U_i, t and $Pending[]$ (see. 3.3).

4.6 Probabilistic guarantees with unreliable medium

It is a conventional belief that HRT traffic must have a 100% guarantee. In practice this cannot be achieved when the environment or the behavior of some components of the system are not fully predictable. Should transmission errors occur randomly on the bus then it is not unlikely that in the lifetime of the system, missed deadlines could happen with ASAP, DP or traffic shaping scheduling. It is therefore crucial for the designer to assess the risks and, if necessary, take the appropriate fault-tolerance strategies (like oversampling or mode change). The reader might refer to [8] and [23] for a good starting point on this topic. In this paragraph, we propose a simple mechanism to provide probabilistic guarantees to prevent HRT messages from missing their deadlines when using the proposed traffic shaping policy. The application designer has to define the required safety level for his particular application through a parameter denoted α which is defined as the upper bound on the Deadline Failure Probability (DFP) (see [13]). For instance, $\alpha = 0.001$ means that no periodic frame will, on average, miss its deadline more than once every 1000 transmissions. Basically, the idea enabling us to achieve this Quality Of Service (QOS), is to bring forward the emission time if necessary regarding α. That comes to including in the calculus of the worst-case response time the possibility of transmission errors. The traffic shaping method stays otherwise unchanged. For each frame m of the application, one has to find the smallest tolerable number of errors n_m (starting with $n_m = 0$) such that the requested QOS will be achieved :

$$P[X\left(\mathcal{R}_m(n_m)\right) > n_m] \leq \alpha \tag{18}$$

which can be rewritten as :

$$\left(1 - \sum_{n=0}^{n_m} P[X(\mathcal{R}(n_m) = n]\right) \leq \alpha \tag{19}$$

where $X(t)$ is the stochastic process giving the number of errors during time t and $\mathcal{R}_m(n)$ the worst-case response time with n errors for frame m. If less than n_m errors occur during $\mathcal{R}_m(n_m)$ with $\mathcal{R}_m(n_m) \leq D_m$, the frame m will meet its deadline, otherwise it could miss it. The calculus is performed iteratively, starting with $n_m = 0$ and incrementing this at the end of each unsuccessful step until equation 18 is satisfied or until $\mathcal{R}_m(n_m)$ is larger than the deadline. In the later case, the requested QOS cannot be achieved. For the calculus of $\mathcal{R}_m(n_m)$, the reader might refer to [12, 11], for a stochastic error model making assumptions on both error frequency and gravity (i.e single errors and "bursts" of errors) the reader should consult [13].

5 Experiments

In the rest of the paper, an experimental embedded CAN-based application proposed by PSA (Peugeot-Citröen) Automobiles Company will be considered. Six devices exchange messages : engine controller, wheel angle sensor, AGB (Automatic Gear Box), ABS (Anti-Blocking System), device y[1] and the bodywork gateway. The traffic consists of a set of periodic messages (e.g. speed and torque from the engine controller), see Figure 5, plus a stream of aperiodic frames whose interarrival times are exponentially distributed. Each periodic frame has its deadline equal to its period, the size of periodic frames is 95 bits and the size of aperiodic frames is 75 bits. The periodic part of the traffic induces a network load of 41.02%.

priority	transmitter	T_m	$\mathcal{R}_m$
1	engine controller	10	8
2	wheel angle sensor	14	11
3	engine controller	20	16
4	AGB	15	10
5	ABS	20	14
6	ABS	40	33
7	ABS	15	7
8	bodywork gateway	50	41
9	device y	20	10
10	engine controller	100	88
11	AGB	50	37
12	ABS	100	86

Figure 4: Set of periodic messages (time unit : ms)

The transmission rate of the CAN bus is set to 125kbit/s. Simulations were performed, using QNAP2 discrete event simulator [15], using ASAP, traffic shaping and dual-priority scheduling. In all cases, the average response time (see Figure 5 and 7) and the variance in response times (see Figure 6 and 8) were measured for aperiodic traffic with a total network load varying from 50% to 90% (with a step of 10%).

5.1 Synchronized case

In the synchronized case all nodes of the network start to transmit simultaneously at the start of the system.

[1]The name of this device can not be communicated because of confidentiality.

5.1.1 Average response time for SRT traffic

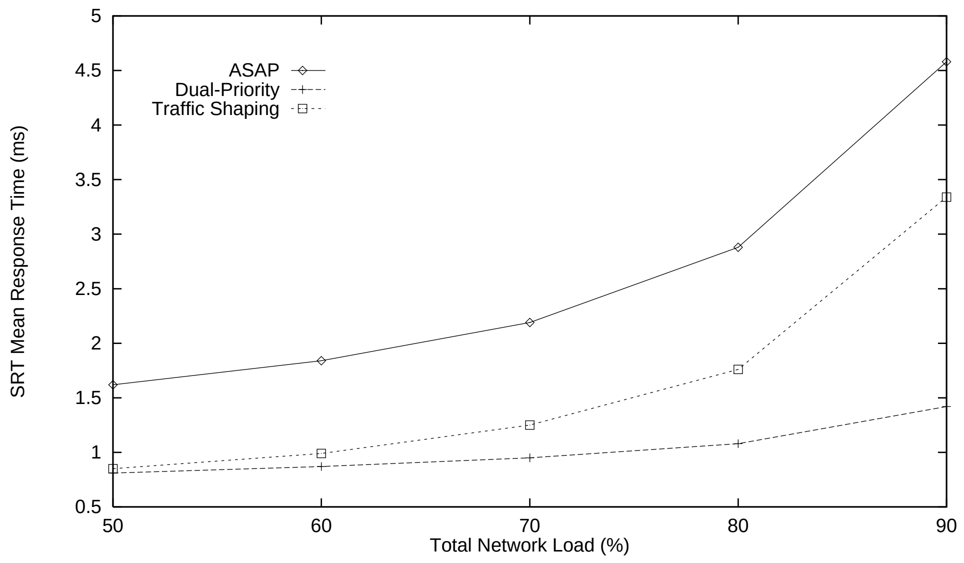

Figure 5: Average response time for SRT traffic in the synchronized case.

From Figure 5, one clearly see that the DP protocol greatly reduces the average response time of aperiodic traffic and gives the best results among the 3 policies. Nevertheless, the traffic shaping strategy performs very well up to a total network load of 80% and even for higher load it stays clearly better as ASAP. In our experiments, the observed gain of traffic shaping over ASAP ranges from a factor 1.90 for a 50% load to a factor 1.40 for a 90% load. It is also noteworthy that whatever the network load, the absolute gain with traffic shaping remains almost constant at about 0.9 ms.

5.1.2 Variance in response times for SRT traffic

The variance measures the dispersion of a sample of values relative to the mean value of the sample. In our case, the smaller the variance of response times, the more deterministic the transmission of aperiodic frames because when transmitting a frame, one can expect its response time to be near the mean value. From Figure 6, it is clear that dual-priority and traffic shaping strategies decreases the variance of aperiodic frame response times. The dual-priority performs better but traffic shaping gives close results up to a network load of 70%.

5.2 Desynchronized case

In the desynchronized case, all nodes do not start simultaneously to transmit : simulations were performed with the first transmission time randomly chosen for each message m between $[0, R_m]$.

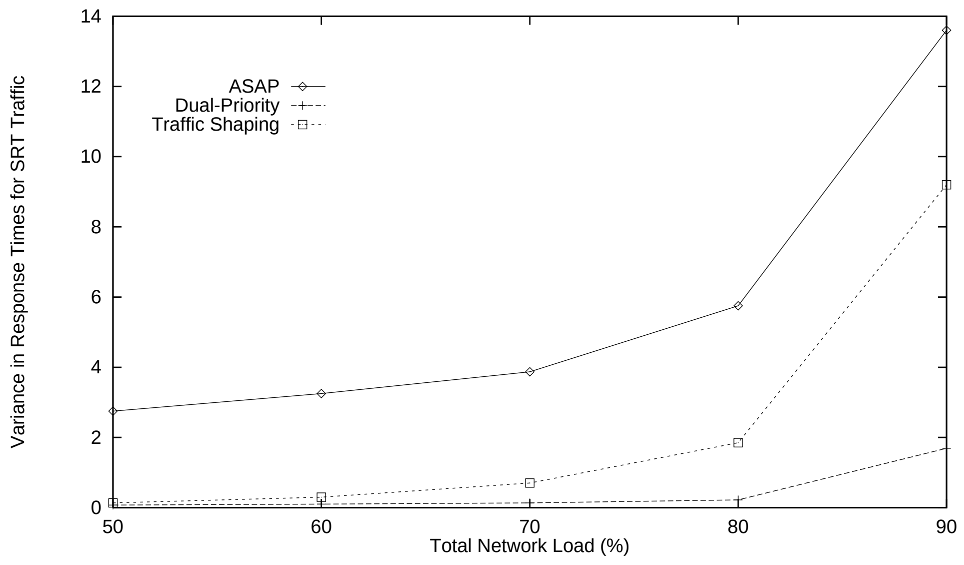

Figure 6: Variance in response times for SRT traffic in the synchronized case.

5.2.1 Average response time for SRT traffic

From Figure 5 and 7, one can note that the gain obtained with traffic shaping over ASAP diminishes when nodes are not synchronized : it now ranges from 1.28 to 1.19. Results with traffic shaping stay almost equal in the synchronized and desynchronized case while the ASAP policy performs much better in the desynchronized case and this simply because the desynchronization already shapes, even imperfectly, the traffic. However, in practice, the offsets between the messages are not random variables which are completely independent of the rest of the process. It is very likely that the gain of the shaping procedure increases when the offsets are correlated, because this brings some new kind of synchronization in the global process.

5.2.2 Variance in response times for SRT traffic

From Figure 8, one sees that the traffic shaping policy performs better than ASAP even if the gain is reduced compared to the synchronized case. The remarks made in Section 5.2.1 remains valid; in practice the desynchronization of the stations is very likely to be less "perfect" that the one given by the uniform distribution used for the simulations, increasing thus the variance in response times for the ASAP policy.

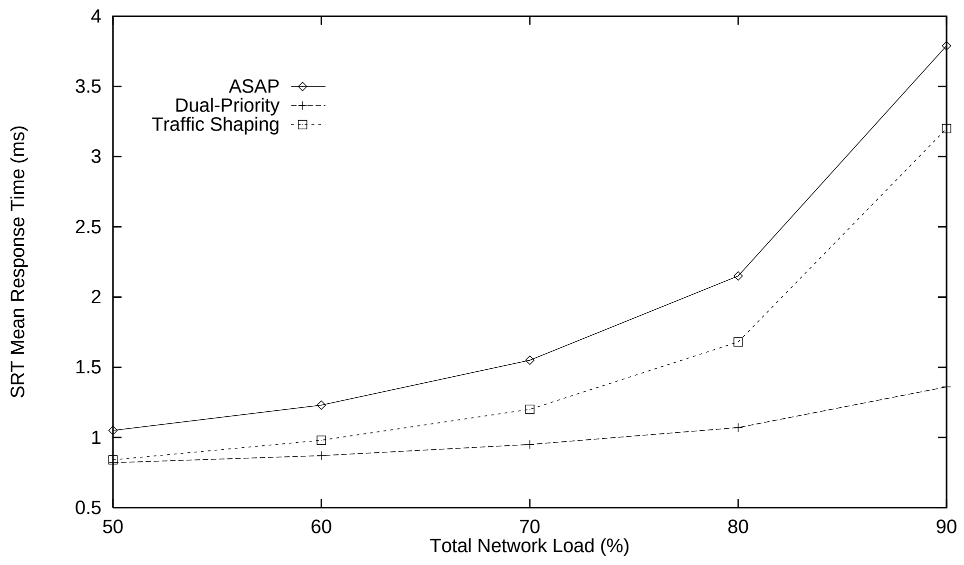

Figure 7: Average response time for SRT traffic in the desynchronized case.

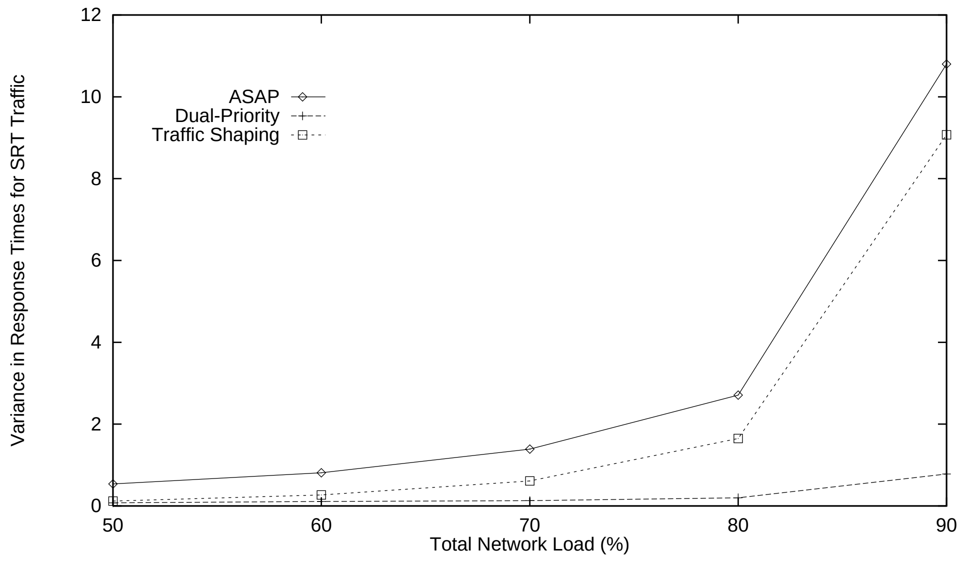

Figure 8: Variance in response times for SRT traffic in the desynchronized case.

6 Conclusions

Experiments of Section 5 shows that the proposed traffic shaping policy performs much better that the ASAP strategy whatever the network load while preserving the property of feasibility as it has been proven in 3.2. Although clearly outperformed by the dual-priority protocol under heavy traffic, the proposed policy is much more easily applicable because it can be used with "off-the-shelf" hardware at the communication controller level.

Furthermore, its complexity being linear in the number of HRT messages, it can be used on-line.

This policy has been extended to handle deadlines smaller than periods, desynchronized stations with known and unknown offsets, jitter in emission and also to provide probabilistic guarantees to prevent HRT frames from missing their deadlines under unreliable transmission. This approach could be adapted to other protocols with bounded access time to the medium such as token protocols.

References

[1] H. Chetto and M. Chetto. Some Results of the Earliest Deadline Scheduling Algorithm. *IEEE Transactions on Software Engineering*, 15(10):1261–1269, October 1989.

[2] R. Davis. Dual Priority scheduling : A means of Providing Flexibility in Hard Real-Time Systems Systems. Technical report, Department of Computer Science, Univ. of York (UK), May 1994. Technical Report YCS230.

[3] R. Davis and A.J. Wellings. Dual priority scheduling. In *Proceedings of the 16th IEEE Real-Time Systems Symposium*, pages 100–109, December 1995.

[4] Association Française de Normalisation (AFNOR). Véhicules routiers - transmission de données, 1990. document R13-708.

[5] B. Gaujal E. Altman and A. Hordijk. Optimal open-loop control of vacations, polling and service assignment. Technical Report 3261, INRIA, 1998.

[6] B. Hajek. Extremal splittings of point processes. *Mathematics of Operation Research*, 10(4):543–556, 1985.

[7] International Standard Organization ISO. *Road Vehicles - Low Speed serial data communication - Part 2: Low Speed Controller Area Network*. ISO, 1994. ISO 11519-2.

[8] H. Kopetz. *Real-Time Sytems : Design Principles for Distributed Embedded Applications*. Kluwer Academic Publishers, 1997.

[9] K Kopetz, R. Nossal, R. Hexel, A Krueger, D. Millinger, R. Pallierer, C. Temple, and M. Krug. Mode handling in the time-triggered architecture. *Control Eng. Practice*, 3(8):1163–1169, 1995.

[10] J. Lehoczky and S. Ramos-Thuel. Aperiodic task scheduling for hard-real-time systems. In *Proceedings IEEE Real-Time Systems*, pages 110–123, December 1992.

[11] N. Navet and Y.-Q. Song. Reliability improvement of the dual-priority protocol under unreliable transmission. *Control Engineering Practice*, 7(8):975–981, 1999.

[12] N. Navet and Y.-Q. Song. Une politique à changement de priorité pour l'ordonnancement de messages dans des environnements bruités. In *Colloque Francophone sur L'ingénierie des Protocoles, CFIP'99*, April 1999.

[13] N. Navet, Y.-Q. Song, and F. Simonot. Worst-case deadline failure probability in real-time applications distributed over CAN (Controller Area Network). *Journal of Systems Architecture*, 46(7):607–618, 2000.

[14] SAE Vehicle Network for Multiplexing and Data Communications Standards Committee. Class b data communications network interface - sae j1850 standard, 1996. Rev. NOV96.

[15] Simulog. Qnap2 Version 9.0 - Reference Manual, 1993.

[16] S.P. Sprunt, L. Sha, and J. Lehoczky. Aperiodic task scheduling for hard-real-time systems. *Real-Time Systems*, 1(1):27–60, 1989.

[17] M. Spuri and G. Buttazzo. Scheduling Aperiodic Tasks in Dynamic Priority Systems. *Real-Time Systems*, 10(2):179–210, 1996.

[18] J.-P. Thomesse, P. Lorenz, J.-P. Bardinet, P. Leterrier, and T. Valentin. Factory instrumentation protocol : Model, products and tools. *Control Engineering Practice*, 38(12):65–67, 1991.

[19] K. Tindell, Burns, and A.J. Wellings. Calculating Controller Area Network (CAN) Message Response Times. *Control Eng. Practice*, 3(8):1163–1169, 1995.

[20] K. Tindell and A. Burns. Guaranteed message latencies for distributed safety-critical hard real-time control networks. Technical report, Department of Computer Science, University of York (UK), May 1994. Technical Report YCS229.

[21] K. Tindell and A. Burns. Guaranteeing message latencies on controller area network (can). In *1st International CAN Conference, ICC'94*, 1994.

[22] K. Tindell and H. Hansson. Babbling idiots, the dual priority protocol, and smart can controllers. In *2nd international CAN Conference, ICC'95*, pages 7.22–7–28, september 1995.

[23] C. Ziegler, D. Powell, and P. Desroches. Dependability of On-Board Automotive Computer Systems. In *IEEE Intelligent Vehicles 1994 Symposium*, pages 568–575, October 1994.

Appendix : Worst-case response time on a priority bus

The worst-case response time $\mathcal{R}_m$, which is defined as the interval between the transmission request and the complete reception of the message to the destination processor(s), must be bounded for each frame by D_m (with $D_m \leq T_m$) otherwise the timing constraint can not be guaranteed and the set of messages of the application is said to be *non-schedulable.* To calculate $\mathcal{R}_m$, as the transmission time C_m (which may not be an integer) and the jitter J_m can be upper bounded, we just have to compute the maximum time needed by the message to gain the arbitration (termed the "interference" time). A message m can be delayed by higher priority messages and by a lower priority message that has already obtained the bus (this time denoted as B_m is the transmission time of the biggest lower priority message). Thus, from [21] and considering that $\mathcal{R}_m$ has to be multiple of the time unit, we have :

$$\mathcal{R}_m = \lceil C_m + J_m + I_m \rceil \tag{20}$$

where I_m is the interference time, i.e. the longest time that all higher priority messages can occupy the bus plus B_m :

$$I_m^n = B_m + \sum_{\forall j \in hp(m)} \left\lceil \frac{I_m^{n-1} + J_j + \tau_{bit}}{T_j} \right\rceil C_j \tag{21}$$

with τ_{bit} the bit time, $hp(m)$ the set of messages of higher priority than m and C_j the transmission time of a message j. For instance, in the context of CAN, with d_j the number of data bytes, one has :

$$C_j = \left(47 + 8d_j + \left\lfloor \frac{34 + 8d_j - 1}{4} \right\rfloor \right) \tau_{bit} \tag{22}$$

where 47 is the size of the fixed-form bit fields of the CAN frame and $\lfloor (34 + 8d_j - 1)/4 \rfloor$ is the maximum number of "stuff" bits (see [7]). I_m is computed starting with $I_m^0 = 0$ until convergence or until $\mathcal{R}_m > D_m$. In the latter case, D_m cannot be guaranteed for message m. The reader could refer to [21, 20] or [19] for more details.