Selective Real-Time Data Emission in Mobile Intelligent Transport Systems

Laurent George and Damien Masson
Université Paris-Est, LIGM (UMR 8049), CNRS, ENPC, ESIEE Paris, UPEM, F-93162, Noisy-le-Grand, France
Email: laurent.george@esiee.fr

Vincent Nelis
CISTER/INESC-TEC, ISEP, Polytechnic Institute of Porto, Portugal
Email: nelis@isep.ipp.pt

***Abstract*—Intelligent Transport Systems (ITS) gather information from their fleet vehicles to improve on their safety and quality of service. Information is sent through a wireless connection (e.g., 3G link) to a central unit responsible for controlling the activity of the vehicles. These data may include for instance the location of the vehicles, their speed, the time spent at each stop, an estimated number of passengers, the state of the doors (opened/closed), etc. In locations where the quality of the network is poor (low data rate), vehicles do not succeed in transmitting *all* the collected information and mechanisms must be implemented to select which information the network can afford to send. Assuming that the data has been labeled and classified beforehand according to its criticality, this paper identifies the exact minimum network speed to successfully send each (set of) data, defining a set of speed thresholds for each level of criticality. We assume that messages are sent according to a non-preemptive fixed priority scheduling. Based on the results of this paper, we expect to later develop algorithms to optimally send the data for arbitrary priority assignments and maximize the utility when choosing which data is discarded due to insufficient network bandwidth.**

1. Introduction

In Intelligent Transport Systems (ITS) [1] for public transports, vehicles like buses, tramways and trains are equipped with tracking devices that periodically send operational information to a central server. The server processes this information and sends feedback to the fleet of vehicles to eventually improve the security, safety, or the quality of the service in general. The tracking devices installed in each vehicle record, sample, and send a broad set of information such as the location of the vehicle (GPS coordinates), the time spent at each stop and station, the time between every two stops, the state of the doors (open/closed), the estimated number of passengers, the temperature in each car (in case of a train for example), fire-detectors information, consumption information, density of the traffic, watchdogs information or even the driver heartbeat in some cases. Note that the amount of data to be sent can be large if images must be forwarded to the central unit (from the in-vehicle security cameras or from the advanced driver-assistance systems for instance). The data is categorized according to its importance or criticality to the project objectives. For example, GPS coordinates can be seen as more important than an estimation of the air quality in the cars. Information is thus clustered and broken into categories and each category is labeled by a measure of its criticality.

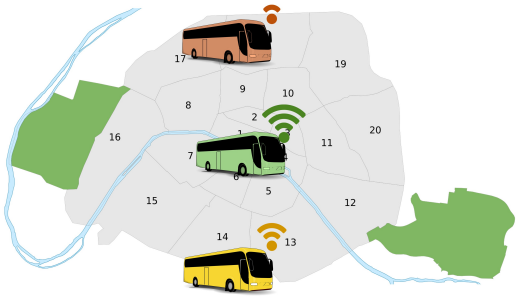

Figure 1. Example of network coverage in a city.

Every vehicle in the fleet sends its data periodically to the base central server over the city's 3G/4G network. However, many cities do not yet provide a full network coverage, and even when they do, there are always regions in which the signal strength drops and becomes insufficient to provide the network bandwidth required to send all the data at the desired frequency, as illustrated by Figure 1. In such regions, the system embedded in the vehicles has three options:

1) It can lower the emission rate by decreasing the frequency at which some of the information is sent. Note that, depending on the application constraints, this is not always feasible as some features are strongly time-dependent and become meaningless if the data is not processed at a specific rate, e.g. fire-detectors information or watchdogs information.
2) It can reduce the amount of data sent to match the capacity of the network that is currently available. That is, when the signal is too weak, the less critical data are no longer sent, which grants more bandwidth to the data that cannot alter their emission rate.
3) It can combine the first two approaches. That is, it can stop sending some of the (less critical) information *and* lowers the emission rate of some others.

The third option is most likely the most efficient technique as it combines the strengths of the first two options.

In this work though, we focus only on the second option in which the system temporarily stops sending data that are less critical to save bandwidth for the more critical ones. This difference of criticality between the data justifies the use of a fixed-priority scheduler for sending out the information. Due to the unpredictable nature of signal strength that can strongly affect the transmission time of the messages (up to the point where the transmission fails), it is arguably more appropriate to give a higher priority to the information of high criticality so that they are always sent before the less critical information. The transmission of data being a non-preemptible operation, it may happen that a data of low criticality is being sent while an information of high criticality becomes "ready-to-be-sent". This is the only case of priority inversion where the transmission of the higher critical data gets blocked by a lower critical data and this scenario can be easily factored in the schedulability analysis for non-preemptive fixed-priority arbitration.

The remainder of the paper is organized as follow. We present the model and notations in Section 2. Then the problem is formalized in Section 3. Section 4 recapitulates basics from the state of the art required to present our solution in Section 5 and to follow the proof of its correctness given in Section 6. Section 7 review existing related works and we conclude the paper in Section 8 with a discussion on future work directions.

2. Model of computation

The application is composed of a set τ of n periodic tasks denoted $\tau = \{\tau_1, \tau_2, \ldots, \tau_n\}$. Every task models the process of periodically sending one information from the bus to the central station, such as the location of the vehicle (GPS coordinates) or the time spent at the last stop. Every task τ_i is activated periodically every T_i time units. We call each activation of a task a "job" and we say that the task "releases" a job when it gets activated. We denote the jobs by $\tau_{i,j}$, where $j \in \mathbb{N}_0$ is the job index. The time of the first activation of each task is not known. That is, once the first job $\tau_{i,0}$ is released (at any time $t > 0$), its subsequent jobs $\tau_{i,j}$, $\forall j > 0$, are released exactly T_i time units apart, starting from time t, i.e. $\tau_{i,j}$ is released at time $t + jT_i$, $\forall j$.

Every job of a same task sends a unique message of fixed length over the network. The transmission time of each message depends on the network's bandwidth available at the time of sending the message and the strength of the signal varies with the location of the vehicle. In some neighborhoods, the poor network coverage results in low-speed transmissions whereas at some other locations, the signal is strong and offers high-speed flawless communications. We denote by C_i the length of every message (in Megabits) to be delivered by the jobs of τ_i. At a given reference network speed of 1 Mbit/s (Megabit per second), it takes exactly C_i seconds to deliver each message of τ_i. From there, it trivially holds that at any speed of x Mbit/s, it takes exactly $\frac{C_i}{x}$ seconds to transfer each message of τ_i. The transmission time of every job $\tau_{i,j}$ of τ_i is thus linear in the network speed. A poor signal strength yields a long transmission time and inversely, a strong signal allows for faster transmissions.

It is important to note here that what we call *bandwidth* is the available bandwidth for the application, taking into account all optimizations made by the network protocols, the congestion on the network and so on. It is not in the scope of the paper to discuss how the available bandwidth is estimated, however, amongst other solutions, one could imagine that an acknowledgement is received after each transmissions to the central station, and that the bandwidth is computed from the measured delay to receive the acknowledgement.

Every task τ_i is assigned a relative temporal deadline denoted by D_i (relative to its activation time), with the interpretation that each of its jobs $\tau_{i,j}$ must have sent its message entirely before or at its absolute deadline set D_i time units after its release. We consider the constrained-deadline task model in which $D_i \leq T_i, \forall i$, meaning that every job must have entirely sent its message before the next job (from the same task) is released. Every τ_i is also assigned a *criticality* χ_i that qualifies the "importance" of the information that it sends to the central station. For example, χ_1 could represent *very important* messages, while χ_2 could be used for *moderately important* messages, χ_3 for *non crucial* information, etc. For convenience, we denote by $\tau(k)$ the set of tasks of criticality χ_k and we assume that $k \in [1, x]$, where x is the number of different criticality in the system (criticality 1 is the highest criticality level, then 2, 3, and so on till x, the least critical one).

All the messages are transmitted sequentially over the network in a non-preemptive way. This means that once a job has started to send its message, it cannot be interrupted or paused and continues emitting until its message is entirely transmitted. Since there may be several jobs in the system that are ready to send their message at the same time, the system uses a scheduling algorithm to define the order in which the jobs access the network. The scheduler works based on fixed task-level priorities. That is, every task τ_i is assigned a constant priority that is passed on to all its jobs. We denote by $\mathrm{hp}(\tau_i)$ (resp., $\mathrm{lp}(\tau_i)$) the set of tasks of higher (resp., lower) priority than τ_i and by $\mathrm{hep}(\tau_i)$ (resp., $\mathrm{lep}(\tau_i)$) the set $\mathrm{hp}(\tau_i) \cup \{\tau_i\}$ (resp., $\mathrm{lp}(\tau_i) \cup \{\tau_i\}$).

The scheduler is designed so that at any point in time, if two jobs are ready to send their message and compete for the access to the network, it is the job coming from the task with the highest priority that will be granted the access first (since the task priorities are passed on to the jobs). This implicitly assumes that all the tasks have distinct priorities and there is at most one job per task that is ready to send its message at any point in time. The latter is guaranteed by the model itself since $D_i \leq T_i$ for all τ_i and thus if two jobs of the same task ever compete for the access to the network at the same time, then one of them has failed to meet its deadline, which is not acceptable.

The relation between the priority of the tasks and their criticality is simple. As explained earlier, the tasks of a same criticality (say χ_k) are grouped together in $\tau(k)$. All the tasks in the system have a different priority, the only constraint imposed on the system is that all the tasks of a same group $\tau(k)$ have a higher priority than all the tasks of all the groups $\tau(j)$ for $j > k$. We believe that this is a reasonable assumption in a real-world scenario: even if important tasks may have longer deadlines and periods

than less important messages – and thus, in theory, the system could potentially serve the less important tasks first – by giving a higher priority to the more important tasks we force them to be served before the less important ones. This is highly recommended as the network speed can change at any moment and we do not want to delay the transmission of important messages (and take the risk of not being able to send them on time) to benefit less critical ones. Note that we do not impose any further restriction on the priority assignment, meaning that any fixed-priority assignment can be used to define the priority of the tasks within each group.

From this point onwards, we shall assume that every task in the system has been assigned a priority such that the above restriction is respected and the entire system is schedulable (i.e. all the deadline are met) at a network speed of 1 Mbit/s. To make sure that the second assumption is always verified, the data rate unit (here, Mbit/s) can be assumed to be Gbit/s or even Tbit/s provided that all the messages lengths C_i, $\forall i$, are converted and expressed in that new data rate. In the remainder of the paper, we shall assume Mbits/s to be the reference network speed and the unit in which the C_i values are expressed.

3. Problem statement

While vehicles move from one part of the city to another, the network speed naturally varies along with the strength of the network signal. As the signal gets weaker, it may become infeasible to send all the information and still verify the timing requirement of every task. To make sure that the critical information is still sent on time, we define x thresholds $\{\mathrm{sp}_1, \mathrm{sp}_2, \ldots, \mathrm{sp}_x\}$ within the speed domain such that, each time the current speed drops below one of these thresholds, say sp_k, all the tasks from the criticality groups $\tau(j)$, with $j > k$, are immediately suspended. They will no longer send messages until the network speed raises again above the threshold sp_k. If one of these tasks with a lower criticality than χ_k was sending a message when the network speed crossed the threshold, then it cannot be stopped (since sending is not preemptible) and it must continue until its entire message has been sent. After that, the task is suspended together with all the tasks of its criticality level and lower levels. The problem is thus to find the minimum speeds sp_k, for all $k \in [1, x]$, such that the groups of tasks $\tau(1)$, $\tau(2)$, ..., $\tau(k)$ is schedulable (i.e. meets all the deadlines). These minimum speeds will then be used as the thresholds described above.

4. Prerequisite

First, let us introduce some extra notations, properties, and basic concepts that we shall use in the computation of our solution and the proof of its optimiality. For a complete state of the art on the non preemptive scheduling problem, the interested reader can refer to [2], [3], [4], [5] amongst many other publications. We just recapitulate here the main results we need.

In the analysis of the optimality of our solution, the time windows of interest are the so-called level-i busy windows. When considering the scheduling of n tasks $\tau_1, \ldots, \tau_n$ indexed by decreasing order of priority, the level-i busy window (with $i \in [1, n]$) is the *longest* time window in which (1) only the jobs from tasks $\tau_1, \ldots, \tau_i$ send their messages (except the first job as we will discuss later) and (2), there is not a single instant at which no message is being sent (the network is never idle within that window). It has been proven in Lemma 6 of [4] (Section 4.3) that the job-release scenario that leads to the longest level-i busy window is the one at which all the tasks $\tau_1, \tau_2, \ldots, \tau_i$ release a job at a same time (say, t) *and* the task $\tau_k \in \mathrm{lp}(\tau_i)$ with the longest C_k releases a job at time $t - \epsilon$. Assuming that no job is sending a message at time $t - \epsilon$, that job from τ_k starts to send his immediately upon its release. That is, the level-i busy window begins at time t with the sending of the longest lower-priority message (here, that of τ_k). This contribution of a lower priority task to the level-i busy window is called the "blocking term" B_i and is calculated as

$$B_i = \max_{k \in \mathrm{lp}(\tau_i)} \{C_k - \epsilon\} \tag{1}$$

The ϵ time units subtracted from the C_k's account for the ϵ time units during which τ_k is already sending its message before the beginning of the busy window (i.e., between the instants $t - \epsilon$ and t). Starting with an initial length $L_i = B_i$, the length of the level-i busy window can be computed by iteratively adding to L_i the contribution of τ_i and all the higher priority tasks that release jobs during these L_i time units. Formally, it has been proven in Theorem 15 of [4] that the length L_i of the level-i busy window, for $i \in [1, n]$, is the first fixed-point solution of the equation

$$L_i = B_i + \sum_{j \in \mathrm{hep}(\tau_i)} \left(1 + \left\lfloor \frac{L_i}{T_j} \right\rfloor\right) \times C_j \tag{2}$$

It is also demonstrated in the same article that the maximum *response time*[1] of the jobs of τ_i is always observed within the level-i busy window. It is thus sufficient to verify that all the deadlines are met for the jobs of τ_i within that time window to conclude on the schedulability of τ_i. We also know that the number n_i of jobs of τ_i released in the level-i busy window is given by

$$n_i = 1 + \left\lfloor \frac{L_i}{T_i} \right\rfloor \tag{3}$$

If we re-index the n_i jobs of τ_i that are released in the level-i busy window from 0 to $n_i - 1$, then the latest time at which job $\tau_{i,j}$ with $j \in [0, n_i - 1]$ starts sending its message is given by

$$w_{i,j} = B_i + j \times C_i + \sum_{k \in \mathrm{hp}(\tau_i)} \left(1 + \left\lfloor \frac{w_{i,j}}{T_k} \right\rfloor\right) \times C_k \tag{4}$$

Finally, for each such job $\tau_{i,j}$, with $j \in [0, n_i - 1]$, released in the level-i busy window, we denote by $S_{i,j}$ the set of all the time-instants between its release at time jT_i and its deadline at time $jT_i + D_i$, at which the higher priority tasks release a job. Formally, for given task

1. The response time of a job is the time between its release and its completion.

and job indexes i and j ($j \in [0, n_i - 1]$), the set $S_{i,j}$ is computed as

$$S_{i,j} = \bigcup_{k \in \mathrm{hp}(\tau_i)} \left\{ rT_k \mid r \in \mathbb{N}^+ \text{ and } jT_i < rT_k < jT_i + D_i \right\} \bigcup \left\{ jT_i, jT_i + D_i \right\} \tag{5}$$

It is important to note that the time-instants jT_i and $jT_i + D_i$ corresponding to the release and deadline of $\tau_{i,j}$ are also included in the set $S_{i,j}$ (in the second line of the equation).

5. Our solution

With the notations introduced in the previous section, we shall prove that for any set of n periodic tasks $\tau_1, \ldots, \tau_n$ (indexed in decreasing order of priority), the minimum speed sp that allows for all the deadlines to be met is given by

$$\mathrm{sp} = \max_{i \in [1,n]} \left(\max_{j \in [0, n_i - 1]} \left(\min_{t \in S_{i,j}} \left(\max \left(\frac{w_{i,j}}{t - \epsilon}, \frac{w_{i,j} + C_i}{jT_i + D_i} \right) \right) \right) \right) \tag{6}$$

The rational behind that equation is that the minimum speed is, for each job, either constrained by its nececity to start on time ($\frac{w_{i,j}}{t-\epsilon}$) or to finish before its deadline ($\frac{w_{i,j}+C_i}{jT_i+D_i}$). The more constrained job has to be search amongst all the jobs in the set described in the previous section, for all higher priority tasks, and the process must be iterate for each task. A formal proof is given in the next section (see Lemma 1).

A solution to the problem described in Section 3 can easily be derived from Equ. 6. Given a set of n tasks $\tau_1, \tau_2, \ldots, \tau_n$ grouped in x groups $\{\tau(1), \tau(2), \ldots, \tau(x)\}$ such that

1) every task of the group $\tau(\ell)$, $\ell \in [1, x]$, has criticality χ_ℓ
2) every task of the group $\tau(\ell)$, $\ell \in [1, x-1]$, has a higher priority than all the tasks of $\tau(\ell + 1)$

the minimum speed sp_k, $k \in [1, x]$, such that all the groups $\tau(1)$, $\tau(2)$, ..., $\tau(k)$ meet their deadlines is given by the speed sp computed by Equ. 6, where the n tasks considered in the computation are the tasks in all the groups $\tau(1), \tau(2), \ldots, \tau(k)$. In the next section we demonstrate the correctness and optimality of Equ. 6.

6. Proof of optimality

We now show how to compute the exact minimum speed satisfying the deadlines of a given task set in the case of non-preemptive fixed priority scheduling. This computation was proposed in the state of the art by dichotomy (see section 7), we here explicit the exact formula.

Lemma 1. *For any set τ of n periodic tasks $\tau_1, \ldots, \tau_n$ (indexed in decreasing order of priority), the exact minimum speed that allows for all the deadlines to be met is given by* sp *as defined in Equ. 6.*

Proof. In this proof, not only we demonstrate that all the deadlines are met when running all the n tasks at speed sp as defined by Equ. 6, but also that any slower speed $\mathrm{sp}^{\mathrm{low}} < \mathrm{sp}$ does *not* allow to meet all the deadlines.

Let p be any value of $i \in [1, n]$ that maximizes the outermost "max" operator of Equ. 6. Likewise, let q denote any value of j that maximizes the second outermost "max" operator. That is, Equ. 6 is maximized for $i = p$ and $j = q$ and we can rewrite it as

$$\mathrm{sp} = \min_{t \in S_{p,q}} \left(\max \left(\frac{w_{p,q}}{t - \epsilon}, \frac{w_{p,q} + C_p}{qT_p + D_p} \right) \right) \tag{7}$$

Let $t_1, t_2, \ldots, t_x$ denote all the time-instants of $S_{p,q}$ sorted in increasing order, i.e., $t_1 < t_2 < \ldots < t_x$. We know from Equ. 5 that $x \geq 2$ as $S_{p,q}$ contains at least the two time-instants $t_1 = qT_p$ and $t_x = qT_p + D_p$. Now, let t_r be any of the time-instants $\in S_{p,q}$ that minimizes the outermost "min" operator of Equ. 7, we can then further simplify it as

$$\mathrm{sp} = \max \left(\frac{w_{p,q}}{t_r - \epsilon}, \frac{w_{p,q} + C_p}{qT_p + D_p} \right) \tag{8}$$

From this simplified equation, two cases must be studied.

Case 1: $\dfrac{w_{p,q}}{t_r - \epsilon} \geq \dfrac{w_{p,q} + C_p}{qT_p + D_p}$

In this case, we get $\mathrm{sp} = \frac{w_{p,q}}{t_r - \epsilon}$ and *at that speed*, the q^{th} job $\tau_{p,q}$ of task τ_p in the level-p busy window starts sending its message at time $\frac{w_{p,q}}{\mathrm{sp}} = \frac{w_{p,q}}{\frac{w_{p,q}}{t_r - \epsilon}} = t_r - \epsilon$ and thus slightly before time t_r. At this time, $\tau_{p,q}$ starts to send its message in a non-preemptive manner until it finishes and since $\mathrm{sp} \geq \frac{w_{p,q}+C_p}{t_x}$ (from the case), we know that it will finish before its deadline at time t_x. Note that by Equ. 6, we know from the first two maximum operators that sp is higher than (or equal to) the minimum speed required to execute every job from every other task on time.

Now, we show that in this case, any speed $\mathrm{sp}_{\min}$ lower than sp does not allow all the deadlines to be met. By contradiction, suppose that $\mathrm{sp}_{\min} < \mathrm{sp}$ does allow to meet all the deadlines. Since by assumption $\mathrm{sp}_{\min} < \mathrm{sp} = \frac{w_{p,q}}{t_r - \epsilon}$, it holds that at speed $\mathrm{sp}_{\min}$ the job $\tau_{p,q}$ starts to send its message later than time $t_r - \epsilon$ but before t_x (because we assumed that all the deadlines are met at speed $\mathrm{sp}_{\min}$). Let t_{u-1} and t_u be the two consecutive time-instants in $S_{p,q}$ such that $t_r \leq t_{u-1} \leq \frac{w_{p,q}}{\mathrm{sp}_{\min}} < t_u$. We know that those two instants exist since in the worst-case scenario, we have $t_u = t_x$ and we know that $\frac{w_{p,q}}{\mathrm{sp}_{\min}} < \frac{w_{p,q}+C_p}{\mathrm{sp}_{\min}} < t_x$ (because we assumed that $\mathrm{sp}_{\min}$ allows to meet all the deadlines). Let sp_u be the speed calculated by Equ. 7 when considering $t = t_u \in S_{p,q}$ in the "min" operator, i.e. $\mathrm{sp}_u = \max \left(\frac{w_{p,q}}{t_u - \epsilon}, \frac{w_{p,q}+C_p}{t_x} \right)$. Because of that minimum, we know that the speed sp computed previously at time t_r is $\leq \mathrm{sp}_u$ and again, two cases may arise for sp_u.

Case 1.1: $\dfrac{w_{p,q}}{t_u - \epsilon} \geq \dfrac{w_{p,q} + C_p}{t_x}$

From the case, it holds that $\mathrm{sp}_u = \frac{w_{p,q}}{t_u - \epsilon}$ and thus at that speed, the job $\tau_{p,q}$ starts to send its message at time $t_u - \epsilon$. Therefore, since $\mathrm{sp}_{\min} < \mathrm{sp} \leq \mathrm{sp}_u$, $\tau_{p,q}$ finishes *after* time t_u if processed at speed $\mathrm{sp}_{\min}$. This

contradicts our definition of t_u that imposes $\frac{w_{p,q}}{\mathrm{sp}_{\min}} < t_u$.

Case 1.2: $\dfrac{w_{p,q}}{t_u - \epsilon} < \dfrac{w_{p,q} + C_p}{t_x}$

Here, $\mathrm{sp}_{\min} < \mathrm{sp}_u = \frac{w_{p,q}+C_p}{t_x}$. If $\mathrm{sp}_{\min} \leq \frac{w_{p,q}}{t_u-\epsilon}$ then we obtain the same contradiction as in Case 1.1. Therefore, it must hold that $\frac{w_{p,q}}{t_u-\epsilon} < \mathrm{sp}_{\min} < \frac{w_{p,q}+C_p}{t_x}$, which also leads to a contradiction because at that speed, (1) the job $\tau_{p,q}$ starts sending its message after time $t_u - \epsilon$ and (2), $\frac{w_{p,q}+C_p}{t_x}$ is the lowest speed at which $\tau_{p,q}$ can possibly send its message before time t_x. Therefore, all the deadlines cannot be met at speed $\mathrm{sp}_{\min} < \frac{w_{p,q}+C_p}{t_x}$.

Case 2: $\dfrac{w_{p,q}}{t_r - \epsilon} < \dfrac{w_{p,q} + C_p}{qT_p + D_p}$

In this case, we have $\frac{w_{p,q}}{t_r-\epsilon} < \mathrm{sp} = \frac{w_{p,q}+C_p}{qT_p+D_p}$. Let us first show that that speed sp allows to meet all the deadlines. At that speed, the job $\tau_{p,q}$ starts sending its message before time $t_r - \epsilon$ (since $\mathrm{sp} > \frac{w_{p,q}}{t_r-\epsilon}$). Then, since $\mathrm{sp} = \frac{w_{p,q}+C_p}{t_x}$ (from the case), we know it finishes sending its message before its deadline at time t_x. Note that by Equ. 6, we know from the two outermost "max" operators that sp is higher than (or equal to) the minimum speed required to execute every job from every other task on time.

Now, let us show that similarly to case 1, any speed $\mathrm{sp}_{\min}$ lower than sp does not allow all the deadlines to be met. By contradiction, suppose that there exists $\mathrm{sp}_{\min} < \mathrm{sp}$ that *does* allow to meet all the deadlines. Two cases may arise:

Case 2.1: $\dfrac{w_{p,q}}{t_r - \epsilon} \leq \mathrm{sp}_{\min} < \mathrm{sp} = \dfrac{w_{p,q} + C_p}{t_x}$

In this case, $\mathrm{sp}_{\min} \geq \frac{w_{p,q}}{t_r-\epsilon}$ and thus at speed $\mathrm{sp}_{\min}$ the job $\tau_{p,q}$ starts sending its message at time $t_r - \epsilon$. Since $\mathrm{sp}_{\min} < \frac{w_{p,q}+C_p}{t_x}$ (from the case), it cannot finish sending the message by its deadline at time t_x.

Case 2.2: $\mathrm{sp}_{\min} \leq \dfrac{w_{p,q}}{t_r - \epsilon} < \mathrm{sp} = \dfrac{w_{p,q} + C_p}{t_x}$

At that speed $\mathrm{sp}_{\min}$, the job $w_{p,q}$ starts to send its message after time $t_r - \epsilon$ (say at time $t^* > t_r - \epsilon$). At that time t^*, all the higher priority jobs that have arrived at every time-instant $t_u \in S_{p,q}$, with $t_r \leq t_u \leq t^*$, will have priority over $\tau_{p,q}$. However, knowing that $\mathrm{sp}_{\min} < \frac{w_{p,q}+C_p}{t_x}$, even without these extra jobs arrived within $[t_r, t^*]$ the speed $\mathrm{sp}_{\min}$ is already to slow for $\tau_{p,q}$ to finish sending its message by the deadline at time t_x.

The proof has successfully covered all possible cases and showed for each one that the speed sp as defined by Equ. 6 is the *lowest* speed at which the n tasks $\tau_1, \tau_2, \ldots, \tau_n$ can meet their deadlines.

□

7. Related Work

A common performance measurement tool for scheduling algorithms is the so called speedup factor. This speedup factor represents the minimum factor by which the processor speed should be increased in order for a given algorithm α to schedule any task-set which is schedulable by an optimal algorithm. In that sense, it gives a metric to measure the gap between α and an optimal algorithm. In order to compute the value, or bounds, of this speedup factor, one has also to compute the critical scaling factor (the maximal factor by which each tasks cost can be increase in order to keep the task-set feasible), but not for any kind of task-set, only for special-case task-sets which model the limit cases. This concept is so not directly linked to our present work, but the interested reader can refer to [6].

Previous works have investigated on the maximal slowdown factor of a processor, particularly in the context of energy-aware scheduling or in the critical scaling factor, in the context of sensitivity analysis. These problems are similar: slowdown the processor is equivalent to increase the tasks length. However, this problem is mainly studied in the context of preemptive tasks. For example, Lehoczky et al. give a threshold value to compute the critical scaling factor under Rate Monotonic analysis in [7]. For non preemptive tasksets, the slowdown factor can be compute for EDF as explained in [8], but to the best of our knowledge, works addressing this issue for fixed priority non preemptive task-sets use binary search methods to approximate the slowdown/critical-scaling factor, as in [9].

In the case of a system where tasks can experience different execution durations, Mixed Criticality (MC) was introduced to arbitrate between critical and non critical tasks when the schedulability cannot be satisfied if both critical and non critical tasks experience their maximum WCET.

Mixed Criticality (MC) was first introduced by Vestal in [10] for periodic tasks and dual criticality systems HI and LO on a uniprocessor system.

A task can be of HI or LO criticality and is characterized by: a minimum inter-arrival time T (period), a relative deadline denoted D and a Worst Case Execution Time (WCET) denoted $C(LO)$ (respectively $C(HI)$) associated to LO (respectively HI) mode representing the budget of time given to a task for its execution. In the classical MC model, $C(HI) \geq C(LO)$ for HI tasks and the system starts in LO mode where both HI and LO tasks can execute and then can switch to HI mode as soon as any HI job executes for its $C(LO)$ without completing. In HI mode, only HI tasks can be executed, LO tasks are suspended with no more guarantee. The condition that triggers a mode change was only based on a WCET overrun w.r.t. the current criticality level [10]. Several approaches have been considered for constrained deadline tasks: EDF-VD [11], AMC [12].

This classical model has been extended to other ones:

- With two Fixed Priorities for a LO-crit task in LO and HI modes. The idea is to reduce the priority of LO-crit tasks in HI mode such that LO-crit tasks do not interfere with HI-crit tasks in HI mode [13].
- By reducing the budget of some of the LO-crit tasks in HI mode [13].
- By increasing the period of LO-crit tasks in HI mode. This approach has been investigated with the elastic model [14] applied to EDF scheduled tasks in the context of MC [15]. The aim of the elastic model was to adapt the Quality–of–Service of a system or handle overloaded situations by modifying the periods of the tasks during the execution of the system. The period of a LO task

is higher when the system criticality is HI than in LO mode.

In this paper, we consider the case were the criticality of a task is such that any task τ_i of arbitrary criticality has a higher priority than any task of lower criticality than τ_i.

8. Conclusion and Future work

In this paper, we considered the problem of sending information from a mobile ITS vehicle to a central entity according their importance. We supposed a wireless link whose quality may vary according to the location of the vehicle. We characterized the exact speed at which all messages of the same or higher importance can be sent. This leads to define a set of speed thresholds where the transmisison of less important messages should be stopped. We assumed a non-preemptive fixed priority scheduling where less important messages all have a lower priority than any higher priority messages. Messages of the same importance can have distinct priorities. The speed thresholds we obtain are optimal in that context.

As a further work, we will explore optimal fixed priority assignement strategies when no restriction is imposed on the priority of messages. In [4], it is showed that Ausley's priority assignment is optimal for non-preemptive fixed priority scheduling. We would like to explore a robust priority assignement minimizing the speed of the thresholds at which less important messages should be stopped. We then would like to propose an algorithm minimizing the number of speed thresholds while satisfying the deadlines of messages.

References

[1] EU, "on the framework for the deployment of intelligent transport systems in the field of road transport and for interfaces with other modes of transport," *Official Journal of the European Union*, Jul. 2010. [Online]. Available: https://goo.gl/XwoFHX

[2] K. W. Tindell, H. Hansson, and A. J. Wellings, "Analysing real-time communications: controller area network (can)," in *1994 Proceedings Real-Time Systems Symposium*, Dec 1994, pp. 259–263.

[3] K. Tindell, A. Burns, and A. Wellings, "Analysis of hard real-time communications," *Real-Time Systems*, vol. 9, pp. 147–171, 1994.

[4] L. George, N. Rivierre, and M. Spuri, "Preemptive and Non-Preemptive Real-Time UniProcessor Scheduling," INRIA, Research Report RR-2966, 1996, projet REFLECS. [Online]. Available: https://hal.inria.fr/inria-00073732

[5] R. J. Bril, J. J. Lukkien, and W. F. Verhaegh, "Worst-case response time analysis of real-time tasks under fixed-priority scheduling with deferred preemption," *Real-Time Syst.*, vol. 42, no. 1-3, pp. 63–119, Aug. 2009. [Online]. Available: http://dx.doi.org/10.1007/s11241-009-9071-z

[6] R. I. Davis, A. Burns, S. Baruah, T. Rothvoß, L. George, and O. Gettings, "Exact comparison of fixed priority and edf scheduling based on speedup factors for both pre-emptive and non-pre-emptive paradigms," *Real-Time Systems*, vol. 51, no. 5, pp. 566–601, Sep 2015. [Online]. Available: https://doi.org/10.1007/s11241-015-9233-0

[7] J. Lehoczky, L. Sha, and Y. Ding, *Rate monotonic scheduling algorithm: Exact characterization and average case behavior*. Publ by IEEE, 1989, pp. 166–171.

[8] R. Jejurikar and R. Gupta, "Energy aware non-preemptive scheduling for hard real-time systems," in *17th Euromicro Conference on Real-Time Systems (ECRTS'05)*, July 2005, pp. 21–30.

[9] S. Punnekkat, R. Davis, and A. Burns, *Sensitivity analysis of real-time task sets*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1997, pp. 72–82.

[10] S. Vestal, "Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance," in *Real-Time Systems Symposium*, Dec 2007, pp. 239–243.

[11] S. Baruah, V. Bonifaci, G. D'Angelo, H. Li, A. Marchetti-Spaccamela, S. van der Ster, and L. Stougie, "The preemptive uniprocessor scheduling of mixed-criticality implicit-deadline sporadic task systems," in *Euromicro Conference on Real-Time Systems*, July 2012, pp. 145–154.

[12] S. K. Baruah, A. Burns, and R. I. Davis, "Response-time analysis for mixed criticality systems," in *Real-Time Systems Symposium*, Nov 2011, pp. 34–43.

[13] A. Burns and S. Baruah, "Towards a more practical model for mixed criticality systems," in *Workshop on Mixed Criticality Systems, Real-Time Systems Symposium*, december 2013, pp. 1–6.

[14] G. C. Buttazzo, G. Lipari, and L. Abeni, "Elastic task model for adaptive rate control," in *Real-Time Systems Symposium*, Dec 1998, pp. 286–295.

[15] H. Su and D. Zhu, "An elastic mixed-criticality task model and its scheduling algorithm," in *Conference on Design, Automation Test in Europe*, March 2013, pp. 147–152.